Manual del programador

INDICE – Manual del programador

6 Manual del Programador

6.1 Introducción

En esta fase se describirán las clases que se han identificado en la fase de diseño. Se tratará de realizar una descripción, ya vista en el diseño, pero ahora adaptándola a la tecnología Java, ya que en el diseño se describe independientemente del leguaje elegido para la implementación.

Para la descripción de las distintas clases, nos vamos a ayudar de la documentación generada por el programa javadoc incluido con el entorno de desarrollo de Java (J2SDK).

Distribuiremos las clases según pertenezcan al Modelo o Controlador del Patrón MVC.

6.2 Clases del Modelo

tablas

Class ArticuloBean

```
java.lang.Object
  |
  +--tablas.ArticuloBean
```

All Implemented Interfaces:

java.io.Serializable

---

public class **ArticuloBean**
extends java.lang.Object
implements java.io.Serializable

Clase que sirve para acceder a la base de datos y que gestiona los Articulos que hay en el sistema, permite albergar las búsquedas en páginas (búsqueda PRS).

See Also:

Serialized Form

Constructor Summary
ArticuloBean() Realiza la conexión con la base de datos

Method Summary	
void	**buscar**(int _idSecc) Se buscan los artículos que están contenidas en una determinada Sección (Busqueda PRS)
void	**buscar**(int _idSecc, java.lang.String _palabra, boolean _titulo, boolean _descripcion, boolean _palabrasClave, int _numResultados) Se buscan los artículos que cumplan los parámetros pasados (Busqueda PRS), si no se introduce parámetro se ignora que el artículo cumpla ese parámetro

void	**buscarArticulo**(int _idArti) Busca un artículo y se situa sobre el para coger sus atributos los métodos get.
void	**buscarArticulosNuevos**(java.lang.String _eMail, int _idRevi) Busca los artículos que está revisando un revisor en una determinada revista.
void	**buscarArticulosNuevosSinInteres**(java.lang.String _eMail, int _idRevi) Busca un artículo que no tiene un revisor asignado pero que además NO coincide con las palabras clave del revisor, además deve estar contenido en una determinada revista
void	**buscarArticulosNuevosSinRevisor**(java.lang.String _eMail, int _idRevi) Busca los artículos que no tienen revisor asociado pero que sus palabras clave tienen alguna coincidencia con las palabras clave del revisor y que estén asociados con una determinada revista.
void	**buscarMasVisitados**(int _idRevi) Se buscan los artículo más visitados de una determinada revista
void	**buscarMasVotados**(int _idRevi) Se buscan los artículo más votados de una determinada revista
void	**buscarTopArticulos**(java.lang.String _palabra, boolean _tituloArti, boolean _descripArti, boolean _palabrasArti) Se realiza una búsqueda y se almacenan todas los artículos en páginas (Busqueda PRS).
void	**buscarUltimos**(int _idRevi) Se buscan los últimos artículo publicados en una determinada revista
void	**desconectar**() Se desconecta la base de datos.
boolean	**eliminarArticuloNuevo**(int _id) eliminar un articulo
boolean	**eliminarPalabrasClaveArticulo**(int _id) Elimina las palabras claves asociadas con un artículo
protected void	**finalize**() Cuando se finaliza el objeto se deberá de desconectar de la base de datos si aun sigue conectado el objeto
boolean	**firstPage**() Se va a la primera página de la búsqueda (PRS) y nos situamos sobre el primer elemento.
java.lang.String	**getAutor**() Se devuelve el Autor del autor del del artículo, previamente hay que hacer una búsqueda de artículos
java.lang.String	**getComentario**() Se devuelve el comentario de un artículo, previamente hay que hacer una búsqueda de artículos
int	**getCountComentariosArticulo**(int _id) Para saber cuantos cometarios tiene un artículo.
java.lang.String	**getDescripcion**() Se devuelve la descripcion del Artículo, previamente hay que hacer una búsqueda de artículos

java.lang.String	**getElements**() Devuelve el número de resultados de la búsqueda PRS
java.lang.String	**getElementsInPage**() Nos devuelve el número de elementos que conforman una página en una búsqueda PRS
java.lang.String	**getEmail**() Se devuelve el correo electrónico del autor de un artículo, previamente hay que hacer una búsqueda de artículos
java.lang.String	**getExtension**() Se devuelve la extensión del artículo, previamente hay que hacer una búsqueda de artículos
java.lang.String	**getExtension**(int _id) Se devuelve la extensión del archivo que contiene el artículo
java.lang.String	**getFecha**() Se devuelve la fecha en la cual se creo el artículo, previamente hay que hacer una búsqueda de artículos o la fecha de la publicación de un artículo busqueda de los artículos publicados
java.lang.String	**getFechaPublicacion**(int _id) Nos devuelve la fecha en la que fue publicado un artículo
int	**getIdArticulo**() Se devuelve el Identificador del Artículo
int	**getIdArticulo**(int _idRevi, java.lang.String _eMail, java.lang.String _titulo, java.lang.String _descripcion, java.lang.String _autor, java.sql.Date _fecha, java.lang.String _extension, boolean _imagen)
int	**getIdRevista**() Se devuelve el Identificador de la revista en la cual está asociado el artículo, previamenta hay que hacer una búsqueda de artículos
int	**getIdSeccion**() Se devuelve el identificador de la sección, previamente hay que hacer una búsqueda de donde están publicados los artículos
int	**getIdSeccPublicado**(int _id) Nos devuelve el identificador de la sección en la cual un artículo está publicado
boolean	**getNext**() Se pasa al siguiente elemento de una búsqueda normal (si es que hay más elementos),
java.lang.String	**getPage**() Para saber sobre que página estamos situados en una búsqueda PRS
java.lang.String	**getPages**() Para saber cuantas páginas ha devuelto la búsqueda PRS
java.lang.String	**getPalabraClave**() Se devuelve el nombre de la palabra clave, se ha de buscar palabras clave
java.lang.String	**getPrsAutor**() Para coseguir el nombre del autor del artículo sobre el cual estamos en una busqueda PRS
java.lan	**getPrsDescripcion**()

g.String	Para coseguir la descripción del artículo sobre el cual estamos en una busqueda PRS
java.lang.String	**getPrsEmail**() Para coseguir el correo electrónico del autor del artículo sobre el cual estamos en una busqueda PRS
java.lang.String	**getPrsExtension**() Para coseguir la extensión en la cual se alverga el artículo correspondiente al artículo sobre el cual estamos en una busqueda PRS
java.lang.String	**getPrsFechaEnvio**() Para coseguir la fecha en la que se envio el artículo sobre el cual estamos en una busqueda PRS
int	**getPrsIDArticulo**() Para coseguir el ID del artículo sobre el cual estamos en una busqueda PRS
int	**getPrsIDRevista**() Para coseguir el ID de la revista en la cual fue subido el artículo sobre el cual estamos en una busqueda PRS
int	**getPrsSumaVotaciones**() Para coseguir la Suma de las votaciones que ha tenido el artículo sobre el cual estamos en una busqueda PRS
java.lang.String	**getPrsTitulo**() Para coseguir el Título que tiene el artículo sobre el cual estamos en una busqueda PRS
int	**getPrsVisitas**() Para coseguir las visitas que ha tenido el artículo sobre el cual estamos en una busqueda PRS
int	**getPrsVotaciones**() Para coseguir el número de votaciones que ha tenido el artículo sobre el cual estamos en una busqueda PRS
int	**getSumaVotaciones**() Se devuelve la Suma de Votaciones que ha tenido el artículo, previamente hay que hacer una búsqueda de artículos
java.lang.String	**getTitulo**() Se devuelve el Titulo del artículo, previamente hay que hacer una búsqueda de artículos
int	**getVisitas**() Se devuelve el numero de visitas que ha tenido este artículo, previamente hay que hacer una búsqueda de artículos

int	**getVotaciones**() Se devuelve el numero de votaciones que ha tenido un artículo, previamente hay que hacer una búsqueda de artículos
boolean	**insertarPalabraClave**(int _id, java.lang.String _palabra) Inserta una palabra clave asociada con un artículo
boolean	**irPagina**(int _pagina) Para ir en una búsqueda PRS a una determinada página
boolean	**isArticuloRelacionadoSeccion**(int _idArti, int _idSecc) Comprueba si la sección y el artículo tienen alguna palabra clave en común
boolean	**isImagen**() Devuelve si el artículo tiene imagen asociada o no, previamente hay que hacer una búsqueda de artículos
boolean	**isPaginaAnterior**() Para saber si hay alguna página antes sobre la cual estamos situados en busqueda PRS
boolean	**isPaginaSiguiente**() Para saber si hay alguna página despues sobre la cual estamos situados en una búsqueda PRS
boolean	**isPrsTieneGrafico**() Para sabes si el artículo tiene una imagen asociada, sobre el cual estamos en una busqueda PRS
boolean	**lastPage**() Se va a la última página de la búsqueda (PRS) y nos situamos sobre su primer elemento.
boolean	**moverArticulo**(int _idArti, int _idSeccDestino) Mover el articulo a otra sección
boolean	**moverArticulos**(int _idSecc, int _idSeccNueva) Mover los artículos de una sección a otra
boolean	**newArticulo**(int _idRevi, java.lang.String _eMail, java.lang.String _titulo, java.lang.String _descripcion, java.lang.String _autor, java.sql.Date _fecha, java.lang.String _extension, boolean _imagen) Inserta una nuevo artículo en el sistema
boolean	**newComentario**(java.lang.String _comentario, int _id, java.lang.String _autor) Insertamos un comentario asociado con un artículo
boolean	**nextInPage**() Se va al siguiente elemento de una búsqueda PRS.
boolean	**nextPage**() Se va a la siguiente página de la búsqueda PRS y nos situamos sobre el primer elemento.
boolean	**previousPage**() Se va a la página anterior en una busqueda PRS
boolean	**revisa**(java.lang.String _eMail, int _id) Este método asocia un artículo con usuario revisor que se encargará posteriormene de revisarlo para su publicación

boolean	**setArticuloEnSeccion**(int _idArti, int _idSecc) Marca el artículo como publicado y lo asocia con la sección en la cual está contenido
void	**setBuscarComentarios**(int _id) Busca los comentarios que se han hecho a un artículo
void	**setBuscarPalabrasClave**(int _id) Busca las palabras clave asociadas a un determinado artículo
void	**setNoNext**() Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda (PRS) al realizar el siguiente Next o alguna de sus variantes.
void	**setVisitaArticulo**(int _idArti) Sumamos una visita al artículo
boolean	**topPage**() Nos ponemos sobre el primer elemento de la página sobre la que estamos situados en una busqueda PRS
boolean	**updateArticulo**(int _idArti, java.lang.String _titulo, java.lang.String _descripcion) Actualizar los datos sobre un artículo
void	**votoArticulo**(int _idArti, int _voto) Sumamos un voto al artículo, actualizando el numero de votaciones que tiene y la suma total de votos

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ArticuloBean

```
public ArticuloBean()
```

Realiza la conexión con la base de datos

Method Detail

revisa

```
public boolean revisa(java.lang.String _eMail,
                      int _id)
```

Este método asocia un artículo con usuario revisor que se encargará posteriormene de revisarlo para su publicación

Parameters:

_eMail - Es el correo electrónico del revisor

_id - Identificador del artículo

Returns:

true -> Si ha sido asociado correctamente False -> Pues no.

---

buscarArticulo

```
public void buscarArticulo(int _idArti)
```

Busca un artículo y se situa sobre el para coger sus atributos los métodos get.

Parameters:

_idArti - Es el identificador del artículo.

---

buscarArticulosNuevos

```
public void buscarArticulosNuevos(java.lang.String _eMail,
```

```
                                        int _idRevi)
```
Busca los artículos que está revisando un revisor en una determinada revista. Posteriormente mediante los metodos get cogeremos la información de estos.
Parameters:
`_eMail` - Es el correo electrónico del revisor.
`_idRevi` - Es el identificador de la revista.

---

buscarArticulosNuevosSinRevisor
```
public void buscarArticulosNuevosSinRevisor(java.lang.String _eMail,
                                            int _idRevi)
```
Busca los artículos que no tienen revisor asociado pero que sus palabras clave tienen alguna coincidencia con las palabras clave del revisor y que estén asociados con una determinada revista.
Parameters:
`_eMail` - Es el correo electrónico del revisor.
`_idRevi` - Es el identificador de la revista.

---

buscarArticulosNuevosSinInteres
```
public void buscarArticulosNuevosSinInteres(java.lang.String _eMail,
                                            int _idRevi)
```
Busca un artículo que no tiene un revisor asignado pero que además NO coincide con las palabras clave del revisor, además deve estar contenido en una determinada revista
Parameters:
`_eMail` - Es el correo electrónico del revisor.
`_idRevi` - Es el identificador de la revista.

---

getIdArticulo
```
public int getIdArticulo()
```
Se devuelve el Identificador del Artículo
Returns:
Un Entero 0 -> Error

---

getIdRevista
```
public int getIdRevista()
```
Se devuelve el Identificador de la revista en la cual está asociado el artículo, previamenta hay que hacer una búsqueda de artículos
Returns:
Un Entero 0 -> Error

---

getExtension
```
public java.lang.String getExtension(int _id)
```
Se devuelve la extensión del archivo que contiene el artículo
Parameters:
`_id` - Es el identificador del artículo
Returns:
La extension

---

getExtension
```
public java.lang.String getExtension()
```
Se devuelve la extensión del artículo, previamente hay que hacer una búsqueda de artículos
Returns:
La extension

---

getComentario
```
public java.lang.String getComentario()
```
Se devuelve el comentario de un artículo, previamente hay que hacer una búsqueda de artículos
Returns:
El comentario

getTitulo
`public java.lang.String` **`getTitulo`**`()`
Se devuelve el Titulo del artículo, previamente hay que hacer una búsqueda de artículos
Returns:
El nombre del título

getIdSeccion
`public int` **`getIdSeccion`**`()`
Se devuelve el identificador de la sección, previamente hay que hacer una búsqueda de donde están publicados los artículos
Returns:
El nombre del título

getAutor
`public java.lang.String` **`getAutor`**`()`
Se devuelve el Autor del autor del del artículo, previamente hay que hacer una búsqueda de artículos
Returns:
El nombre del título

getDescripcion
`public java.lang.String` **`getDescripcion`**`()`
Se devuelve la descripcion del Artículo, previamente hay que hacer una búsqueda de artículos
Returns:
la descripción

getEmail
`public java.lang.String` **`getEmail`**`()`
Se devuelve el correo electrónico del autor de un artículo, previamente hay que hacer una búsqueda de artículos
Returns:
el correo electrónico del autor

isImagen
`public boolean` **`isImagen`**`()`
Devuelve si el artículo tiene imagen asociada o no, previamente hay que hacer una búsqueda de artículos
Returns:
True -> tiene imagen asociada False -> No la tiene

getFecha
`public java.lang.String` **`getFecha`**`()`
Se devuelve la fecha en la cual se creo el artículo, previamente hay que hacer una búsqueda de artículos o la fecha de la publicación de un artículo busqueda de los artículos publicados
Returns:
Una cadena que es la fecha

getVisitas
`public int` **`getVisitas`**`()`
Se devuelve el numero de visitas que ha tenido este artículo, previamente hay que hacer una búsqueda de artículos
Returns:
el número de visitas

getVotaciones
`public int` **`getVotaciones`**`()`
Se devuelve el numero de votaciones que ha tenido un artículo, previamente hay que hacer una búsqueda de artículos

Returns:
el número de visitas

---

getSumaVotaciones

```
public int getSumaVotaciones()
```

Se devuelve la Suma de Votaciones que ha tenido el artículo, previamente hay que hacer una búsqueda de artículos
Returns:
el número de visitas

---

getNext

```
public boolean getNext()
```

Se pasa al siguiente elemento de una búsqueda normal (si es que hay más elementos),
Returns:
True -> Se ha pasado a la siguiente tupla de la búsqueda False -> Estamos al final y no hay más tuplas o no se ha efectuado la operación

---

eliminarArticuloNuevo

```
public boolean eliminarArticuloNuevo(int _id)
```

eliminar un articulo
Parameters:
`_id` - Es el identificador del artículo para eliminar
Returns:
True -> Se ha eliminado correctamente False -> No se ha podido eliminar

---

desconectar

```
public void desconectar()
```

Se desconecta la base de datos.

---

newComentario

```
public boolean newComentario(java.lang.String _comentario,
                             int _id,
                             java.lang.String _autor)
```

Insertamos un comentario asociado con un artículo
Parameters:
`_comentario` - Es el comentario que vamos a insertar
`_id` - Es el identificador del artículo
`_autor` - Es el nombre del autor del comentario
Returns:
True -> Se ha insertado correctamente False -> No se ha insertado correctamente

---

newArticulo

```
public boolean newArticulo(int _idRevi,
                           java.lang.String _eMail,
                           java.lang.String _titulo,
                           java.lang.String _descripcion,
                           java.lang.String _autor,
                           java.sql.Date _fecha,
                           java.lang.String _extension,
                           boolean _imagen)
```

Inserta una nuevo artículo en el sistema
Parameters:
`_idRevi` - Es el identificador de la revista en la cual se quiere publicar el artículo
`_eMail` - Es la dirección de correo electrónico del autor
`_titulo` - Es el titulo que tiene el artículo
`_descripcion` - Es una breve descripción del artículo
`_autor` - Es el nombre del autor del artículo

_fecha - Es la fecha asociada con la insercción de este artículo
_extension - Es la estensión que tiene el artículo (Doc o PDF)
_imagen - Si tiene imagen asociada o no el artículo
return - True -> Se ha insertado False -> No se ha podido insertar

---

votoArticulo

```
public void votoArticulo(int _idArti,
                         int _voto)
```

Sumamos un voto al artículo, actualizando el numero de votaciones que tiene y la suma total de votos
Parameters:
_idArti - Es el identificado del artículo al cual se le ha hecho un voto
_voto - Es la puntuación que se ha dado al boto

---

setVisitaArticulo

```
public void setVisitaArticulo(int _idArti)
```

Sumamos una visita al artículo
Parameters:
_idArti - Es el identificado del artículo al cual se le va a aumentar el numéro de visitas

---

updateArticulo

```
public boolean updateArticulo(int _idArti,
                              java.lang.String _titulo,
                              java.lang.String _descripcion)
```

Actualizar los datos sobre un artículo
Parameters:
_idArti - Es el identificador del artículo
_titulo - Es el titulo que tiene el artículo
_descripcion - Es una breve descripción del artículo
return - True -> Se ha modificado False -> No se ha podido modificar

---

getIdArticulo

```
public int getIdArticulo(int _idRevi,
                         java.lang.String _eMail,
                         java.lang.String _titulo,
                         java.lang.String _descripcion,
                         java.lang.String _autor,
                         java.sql.Date _fecha,
                         java.lang.String _extension,
                         boolean _imagen)
```

---

insertarPalabraClave

```
public boolean insertarPalabraClave(int _id,
                                    java.lang.String _palabra)
```

Inserta una palabra clave asociada con un artículo
Parameters:
_id - Es el identificado del artículo
_palabra - Es la palabra clave que se inserta
Returns:
True -> Se ha insertado correctamente False -> Se ha producido algún error

---

setBuscarPalabrasClave

```
public void setBuscarPalabrasClave(int _id)
```

Busca las palabras clave asociadas a un determinado artículo
Parameters:
_id - Identificador de las palabras clave

---

setBuscarComentarios

```
public void setBuscarComentarios(int _id)
```
Busca los comentarios que se han hecho a un artículo
Parameters:
`_id` - Identificador del articulo

---

getPalabraClave
```
public java.lang.String getPalabraClave()
```
Se devuelve el nombre de la palabra clave, se ha de buscar palabras clave
Returns:
Una cadena que es su nombre

---

eliminarPalabrasClaveArticulo
```
public boolean eliminarPalabrasClaveArticulo(int _id)
```
Elimina las palabras claves asociadas con un artículo
Parameters:
`_id` - Es el identificador del artículo
Returns:
True -> Se ha eliminado correctamente False -> No se ha podido eliminar

---

isArticuloRelacionadoSeccion
```
public boolean isArticuloRelacionadoSeccion(int _idArti,
                                            int _idSecc)
```
Comprueba si la sección y el artículo tienen alguna palabra clave en común
Parameters:
`_idArti` - Es el identificador del artículo
`_idSecc` - Es el identificador de la sección
Returns:
True -> Si hay coincidencias False -> No las hay

---

setArticuloEnSeccion
```
public boolean setArticuloEnSeccion(int _idArti,
                                    int _idSecc)
```
Marca el artículo como publicado y lo asocia con la sección en la cual está contenido
Parameters:
`_idArti` - identificador del artículo
`_idSecc` - identificador de la sección
Returns:
true Se ha efectuado con éxito False pues no se ha efectuado con éxito

---

buscarTopArticulos
```
public void buscarTopArticulos(java.lang.String _palabra,
                               boolean _tituloArti,
                               boolean _descripArti,
                               boolean _palabrasArti)
```
Se realiza una búsqueda y se almacenan todas los artículos en páginas (Busqueda PRS). La búsqueda se realiza mediante los parametros introducidos, si no se introduce se ignora la búsqueda por es campo.
Parameters:
`_palabra` - Palabra de la búsqueda
`_tituloArti` - Se buscan los artículos que contenga la palabra en su titulo esta palabra
`_descripArti` - Se buscan los artículos que contengan la palabra en su descripción
`_palabrasArti` - Se buscan los artículos que contengan la palabra en sus palabras clave

---

buscar
```
public void buscar(int _idSecc)
```
Se buscan los artículos que están contenidas en una determinada Sección (Busqueda PRS)
Parameters:
`_idSecc` - Es el identificador de la sección

buscarUltimos
```
public void buscarUltimos(int _idRevi)
```
Se buscan los últimos artículo publicados en una determinada revista
Parameters:
`_idRevi` - Es el identificador de la revista

buscarMasVisitados
```
public void buscarMasVisitados(int _idRevi)
```
Se buscan los artículo más visitados de una determinada revista
Parameters:
`_idRevi` - Es el identificador de la revista

buscarMasVotados
```
public void buscarMasVotados(int _idRevi)
```
Se buscan los artículo más votados de una determinada revista
Parameters:
`_idRevi` - Es el identificador de la revista

buscar
```
public void buscar(int _idSecc,
                   java.lang.String _palabra,
                   boolean _titulo,
                   boolean _descripcion,
                   boolean _palabrasClave,
                   int _numResultados)
```
Se buscan los artículos que cumplan los parámetros pasados (Busqueda PRS), si no se introduce parámetro se ignora que el artículo cumpla ese parámetro
Parameters:
`_palabra` - Es la palabra que debe contener el artículo en su titulo, descripción o sus palabras clave
`_titulo` - Si se quiere realizar por el título la búsqueda
`_descripción` - Si se quiere realizar por la descripción la búsqueda
`_palabrasClave` - Si se quiere realizar por sus palabras clave la búsqueda
`_numResultados` - Es el número de elementos que se mostrarán en cada página de la búsqueda

getPage
```
public java.lang.String getPage()
```
Para saber sobre que página estamos situados en una búsqueda PRS
Returns:
Nos devuelve un String indicándonos el número de la página sobre la que estamos

getPages
```
public java.lang.String getPages()
```
Para saber cuantas páginas ha devuelto la búsqueda PRS
Returns:
No devuelve el número de páginas en un String

getElementsInPage
```
public java.lang.String getElementsInPage()
```
Nos devuelve el número de elementos que conforman una página en una búsqueda PRS

getElements
```
public java.lang.String getElements()
```
Devuelve el número de resultados de la búsqueda PRS

topPage
```
public boolean topPage()
```

Nos ponemos sobre el primer elemento de la página sobre la que estamos situados en una busqueda PRS

---

setNoNext

`public void` **`setNoNext`**`()`

Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda (PRS) al realizar el siguiente Next o alguna de sus variantes. Muy util si se cambia de página y no queremos que en la primera iteración del Next realice un avance. Asi podemos tratar a todos los elementos de la misma forma en bucle while. Ya que al realizar el bucle se saltaria al hacer el next el elemento sobre el cual estamos situados (Al pasar de página nunca podemos estar antes del primer elemento,siempre nos dejará en el primero).

nextInPage

`public boolean` **`nextInPage`**`()`

Se va al siguiente elemento de una búsqueda PRS.

Returns:

True -> Hay elemento posterior y nos hemos situados sobre el False -> No hay elemento posterior

---

getPrsIDArticulo

`public int` **`getPrsIDArticulo`**`()`

Para coseguir el ID del artículo sobre el cual estamos en una busqueda PRS

Returns:

Nos devulve el Identificador del artículo 0 --> Error

---

getPrsIDRevista

`public int` **`getPrsIDRevista`**`()`

Para coseguir el ID de la revista en la cual fue subido el artículo sobre el cual estamos en una busqueda PRS

Returns:

Nos devulve el Identificador de la revista 0 --> Error

---

getPrsTitulo

`public java.lang.String` **`getPrsTitulo`**`()`

Para coseguir el Título que tiene el artículo sobre el cual estamos en una busqueda PRS

Returns:

Nos devulve el titulo

---

getPrsDescripcion

`public java.lang.String` **`getPrsDescripcion`**`()`

Para coseguir la descripción del artículo sobre el cual estamos en una busqueda PRS

Returns:

Nos devulve la descripción

---

getPrsEmail

`public java.lang.String` **`getPrsEmail`**`()`

Para coseguir el correo electrónico del autor del artículo sobre el cual estamos en una busqueda PRS

Returns:

Nos devulve la dirección de correo electrónico

---

getPrsAutor

`public java.lang.String` **`getPrsAutor`**`()`

Para coseguir el nombre del autor del artículo sobre el cual estamos en una busqueda PRS

Returns:

Nos devulve el nombre del autor

---

getPrsFechaEnvio

`public java.lang.String` **`getPrsFechaEnvio`**`()`

Para coseguir la fecha en la que se envio el artículo sobre el cual estamos en una busqueda PRS

Returns:

Nos devulve la fecha

---

getPrsExtension

`public java.lang.String` **`getPrsExtension`**`()`

Para coseguir la extensión en la cual se alverga el artículo correspondiente al artículo sobre el cual estamos en una busqueda PRS

Returns:

Nos devulve la extension

---

getPrsVisitas

`public int` **`getPrsVisitas`**`()`

Para coseguir las visitas que ha tenido el artículo sobre el cual estamos en una busqueda PRS

Returns:

Nos devulve el número de visitas

---

getPrsVotaciones

`public int` **`getPrsVotaciones`**`()`

Para coseguir el número de votaciones que ha tenido el artículo sobre el cual estamos en una busqueda PRS

Returns:

Nos devulve el número de votaciones

---

getPrsSumaVotaciones

`public int` **`getPrsSumaVotaciones`**`()`

Para coseguir la Suma de las votaciones que ha tenido el artículo sobre el cual estamos en una busqueda PRS

Returns:

Nos devulve la Suma de las votaciones

---

isPrsTieneGrafico

`public boolean` **`isPrsTieneGrafico`**`()`

Para sabes si el artículo tiene una imagen asociada, sobre el cual estamos en una busqueda PRS

Returns:

True -> Tiene imagen False -> no la tiene

---

getFechaPublicacion

`public java.lang.String` **`getFechaPublicacion`**`(int _id)`

Nos devuelve la fecha en la que fue publicado un artículo

Parameters:

`_id` - Es el identificador del artículo

Returns:

Nos devuelve una cadena con la fecha

---

getIdSeccPublicado

`public int` **`getIdSeccPublicado`**`(int _id)`

Nos devuelve el identificador de la sección en la cual un artículo está publicado

Parameters:

`_id` - Es el identificador del artículo

Returns:

Nos devuelve el identificador de la sección

---

isPaginaAnterior

`public boolean` **`isPaginaAnterior`**`()`

Para saber si hay alguna página antes sobre la cual estamos situados en busqueda PRS

Returns:

True -> Hay una página anterior False -> No hay ninguna página anterior

---

isPaginaSiguiente

```
public boolean isPaginaSiguiente()
```

Para saber si hay alguna página despues sobre la cual estamos situados en una búsqueda PRS

Returns:

True -> Hay una página False -> No la hay

---

previousPage

```
public boolean previousPage()
```

Se va a la página anterior en una busqueda PRS

Returns:

True -> Hay página anterior y nos hemos situados sobre el primer elemento False -> No hay página anterior

---

nextPage

```
public boolean nextPage()
```

Se va a la siguiente página de la búsqueda PRS y nos situamos sobre el primer elemento.

Returns:

True -> Hay siguiente página y nos hemos situado en su primer elemento False -> No hay siguiente página

---

irPagina

```
public boolean irPagina(int _pagina)
```

Para ir en una búsqueda PRS a una determinada página

Parameters:

`_pagina` - Es la página a la cual queremos ir

Returns:

True -> Se ha ido correctamente False -> Caso contrario

---

getCountComentariosArticulo

```
public int getCountComentariosArticulo(int _id)
```

Para saber cuantos cometarios tiene un artículo.

Parameters:

`_id` - Es el identificador del artículo

Returns:

El número de comentarios que tiene

---

moverArticulos

```
public boolean moverArticulos(int _idSecc,
                              int _idSeccNueva)
```

Mover los artículos de una sección a otra

Parameters:

`_idSecc` - Es el identificador de la sección de la cual se cogen todos sus artículos para moverlos

`_idSeccNueva` - Es el identificador de la sección destino de todos los artículos

Returns:

True -> Se han movido correctamente False -> No se ha podido mover correctamente

---

moverArticulo

```
public boolean moverArticulo(int _idArti,
                             int _idSeccDestino)
```

Mover el articulo a otra sección

Parameters:

`_idArti` - Es el identificador del artículo

`_idSeccDestino` - Es el identificador de la sección destino del artículo

Returns:

True -> Se han movido correctamente False -> No se ha podido mover correctamente

---

firstPage
`public boolean firstPage()`
Se va a la primera página de la búsqueda (PRS) y nos situamos sobre el primer elemento.
Returns:
True -> Hay primer página y nos hemos situado en su primer elemento False -> No hay ninguna página

---

lastPage
`public boolean lastPage()`
Se va a la última página de la búsqueda (PRS) y nos situamos sobre su primer elemento.
Returns:
True -> Nos hemos situado en la última página y en su primer elemento False -> No hay ninguna página

---

finalize
```
protected void finalize()
                throws java.lang.Throwable
```
Cuando se finaliza el objeto se deberá de desconectar de la base de datos si aun sigue conectado el objeto
Overrides:
`finalize` in class `java.lang.Object`

tablas

Class EstudiosBean

```
java.lang.Object
  |
  +--tablas.EstudiosBean
```

---

public class **EstudiosBean**
extends java.lang.Object
Clase que sirve para acceder a la base de datos y que gestiona la tabla Estudios

---

Constructor Summary
EstudiosBean() Realiza la conexión con la base de datos

Method Summary	
void	**cerrar**() Se Cierra cualquier consulta que halla quedado abierta con la base de datos.
void	**desconectar**() Se Cierra cualquier consulta que halla almacenado resultados en este objeto y la conexión a la base de datos
int	**getIdEstudio**() Se devuelve el Identificador del estudio sobre el cual estamos situados dentro de una búsqueda
int	**getIdEstudio**(java.lang.String _estudio) Se obtiene el ID del estudio asociado con el que le pasamos no afecta a las búsquedas
boolean	**getNext**() Se pasa a la siguiente tupla de la búsqueda (si es que la hay) si se llega al final se cierra la consulta con la base de datos (la conexión permanecerá abierta).
java.lang.String	**getNombreEstudio**() Se Devuelve el nombre del estudio sobre el cual estamos situados en la búsqueda

java.lang.String	**getNombreEstudio**(int _id) Se obtiene el nombre del estudio asociado al ID
boolean	**Insertar**(java.lang.String _estudio) Se inserta un estudio en la tabla
boolean	**setBuscar**(int _id) Se hace una busqueda sobre los estudios que estén asociadas al ID pasado como parámetro
boolean	**setBuscarTodos**() Se realiza una búsqueda en la cual estan todas las tuplas de Estudios.
boolean	**setEliminar**(int _id) Se elimina la tupla de la tabla que este asociada con el ID que se le pase

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

EstudiosBean

public **EstudiosBean**()

Realiza la conexión con la base de datos

Method Detail

Insertar

public boolean **Insertar**(java.lang.String _estudio)

Se inserta un estudio en la tabla

Parameters:

_estudio - El estudio que se va a insertar

Returns:

True -> El estudio se ha insertado en la tabla False -> El estudio no se ha insertado en la tabla

---

getNext

public boolean **getNext**()

Se pasa a la siguiente tupla de la búsqueda (si es que la hay) si se llega al final se cierra la consulta con la base de datos (la conexión permanecerá abierta).

Returns:

True -> Se ha pasado a la siguiente tupla de la búsqueda False -> Estamos al final y no hay más tuplas o no se ha efectuado la operación

---

getIdEstudio

public int **getIdEstudio**()

Se devuelve el Identificador del estudio sobre el cual estamos situados dentro de una búsqueda

Returns:

Devuelve el número el ID del estudio.

---

getIdEstudio

public int **getIdEstudio**(java.lang.String _estudio)

Se obtiene el ID del estudio asociado con el que le pasamos no afecta a las búsquedas

Parameters:

_estudio - Se pasa el estudio del cual queremos obtener su ID

Returns:

Devuelve el ID asociado, 0 si no esta contenido en la tabla

---

getNombreEstudio
`public java.lang.String` **`getNombreEstudio`**`(int _id)`
Se obtiene el nombre del estudio asociado al ID
Parameters:
`_id` - Es el ID del estudio del cual queremos obtener su nombre
Returns:
Devuelve el nombre contenido en la tabla asociado con el ID=_id, si no hay ninguno asociado devuelve la cadena vacia

---

getNombreEstudio
`public java.lang.String` **`getNombreEstudio`**`()`
Se Devuelve el nombre del estudio sobre el cual estamos situados en la búsqueda
Returns:
Nos devuelve su nombre, si hay algún error y no se puediera obtener devolvemos la cadena vacía.

---

setEliminar
`public boolean` **`setEliminar`**`(int _id)`
Se elimina la tupla de la tabla que este asociada con el ID que se le pase
Parameters:
`_id` - Es el ID de la tupla que se quiere borrar de la tabla
Returns:
True -> Se ha efectuado el borrado False -> No se ha efectuado el borrado

---

setBuscar
`public boolean` **`setBuscar`**`(int _id)`
Se hace una busqueda sobre los estudios que estén asociadas al ID pasado como parámetro
Parameters:
`_id` - Se corresponde con el ID de la tabla de la cual el resultado de la búsqueda esta asociado
Returns:
True -> Se ha realizado la búsqueda False -> No se ha realizado correctamente la búsqueda

---

setBuscarTodos
`public boolean` **`setBuscarTodos`**`()`
Se realiza una búsqueda en la cual estan todas las tuplas de Estudios.
Returns:
True -> Se ha realizado la búsqueda con éxito False -> No se ha podido llevar a cabo la búsqueda

---

cerrar
`public void` **`cerrar`**`()`
Se Cierra cualquier consulta que halla quedado abierta con la base de datos.

---

desconectar
`public void` **`desconectar`**`()`
Se Cierra cualquier consulta que halla almacenado resultados en este objeto y la conexión a la base de datos

tablas

Class ForoBean

```
java.lang.Object
  |
  +--tablas.ForoBean
```

All Implemented Interfaces:
java.io.Serializable

---

public class **ForoBean**
extends java.lang.Object

implements java.io.Serializable
Clase que implementa un Foro en el cual se pueden insertar preguntas y respuestas. Se podrán hacer búsquedas PRS, en las cuales el resultado de la búsqueda se almancena en Páginas.
See Also:
Serialized Form

Constructor Summary
ForoBean() Realiza las conexiones oportunas a la base de datos

Method Summary	
boolean	**borrar**(boolean _registrado, boolean _cliente, boolean _administrador, boolean _revisor, java.sql.Date _desde, java.sql.Date _hasta) Borra las preguntas del foro que se corresponda con los parámetros introducidos, si un parámetro no es introducido se ignorará a la hora de realizar el borrado
void	**buscar**(java.lang.String _palabra, int _orden, int _usuario, int _orden2) Se realiza una búqueda de preguntas del Foro y se almacena en páginas (Busqueda PRS)
void	**buscarAvanzada**(java.lang.String _palabra, java.lang.String _nick, java.sql.Date desde, java.sql.Date hasta, int _orden, int _usuario, int _orden2) Se realiza una búqueda avanza de preguntas del foro y se almacena en páginas (Búsqueda PRS)
void	**desconectar**() Cierra las consultas y se desconecta de la base de datos
boolean	**eliminarFila**(int _fila) Para eliminar un fila de una búsqueda PRS y de la base de datos
protected void	**finalize**() Finaliza el objeto, cierra las sesiones y desconecta el objeto de la base de datos si aun sigue conectado y con las consultas abiertas
int	**getCountForoPreguntas**() Para saber cuantas preguntas hay en el sistema.
java.lang.String	**getCuerpo**() Para saber el Cuerpo del elemento sobre el que estamos en una busqueda
java.sql.Date	**getFecha**() Para conocer la fecha que hay en el registro actual de la fecha
int	**getId**() Para saber el identificador del elemento sobre el que estamos en una busqueda
int	**getIdPregunta**() Para saber el identificador de la pregunta de la cual depende una respuesta
java.lang.String	**getNick**() Para conocer el nick del elemento sobre el que estamos en una busqueda
int	**getNumeroFila**() Devuelve el número de fila que corresponde al registro sobre el cual estamos situados de la búsqueda PRS.

int	**getNumResp**(int _id) Para saber cuantas respuestas estan asociadas con la pregunta que se identifica en el parámetro de entrada
int	**getNumResp2**(int _id) Para saber cuantas respuestas estan asociadas con una Respuesta que se identifica en el parámetro de entrada
java.lang.String	**getPage**() Para saber sobre que página estamos situados en una búsqueda PRS
java.lang.String	**getPages**() Para saber cuantas páginas ha devuelto la búsqueda PRS
java.lang.String	**getPrsCuerpo**() Para coseguir el Cuerpo sobre la tupla en la que estamos situados en una búsqueda PRS.
java.lang.String	**getPrsFecha**() Para coseguir la Fecha sobre la tupla en la que estamos situados en una búsqueda PRS.
java.lang.String	**getPrsId**() Para coseguir el Identificador de una pregunta sobre la la que estamos situados en una búsqueda PRS.
java.lang.String	**getPrsNick**() Para coseguir el Nick sobre la tupla en la que estamos situados en una búsqueda PRS.
java.lang.String	**getPrsTitulo**() Para coseguir el Titulo sobre la pregunta sobre la que estamos situados en una búsqueda PRS.
java.lang.String	**getTitulo**() Para saber el Titulo del elemento sobre el que estamos en una busqueda
boolean	**insertar**(java.lang.String _titulo, java.lang.String _cuerpo, java.lang.String _nick, java.sql.Date _fecha, int _dirigido) Se inserta en el Foro una Pregunta
boolean	**insertarRespuesta**(int _id, int _id2, java.lang.String _titulo, java.lang.String _cuerpo, java.lang.String _nick, java.sql.Date _fecha) Se inserta en el Foro una respuesta asociada con una pregunta y una respuesta.
boolean	**insertarRespuesta**(int _id, java.lang.String _titulo, java.lang.String _cuerpo, java.lang.String _nick, java.sql.Date _fecha) Se inserta en el Foro una Respuesta asociada a una pregunta.
boolean	**isPaginaAnterior**() Para saber si hay alguna página antes sobre la cual estamos situados en una búsqueda PRS
boolean	**isPaginaSiguiente**() Para saber si hay alguna página despues sobre la cual estamos situados en una búsqueda PRS
boolean	**isVacio**() Para saber si se ha devuelto algún resultado de alguna búsqueda (PRS)
boolean	**next**()

	Pasamos al siguiente elemento de una búsqueda
void	**prsCerrar**() Cerrar la consulta devuelta en páginas
boolean	**prsFirst**() Se va al primer elemento del resultado de una búsqueda PRS.
boolean	**prsFirstPage**() Se va a la primera página de la búsqueda PRS y nos situamos sobre el primer elemento.
boolean	**prsLastPage**() Se va a la última página de la búsqueda PRS y nos situamos sobre su primer elemento.
boolean	**prsNextInPage**() Se va al siguiente elemento de una búsqueda PRS dentro de una página.
boolean	**prsNextPage**() Se va a la siguiente página de la búsqueda PRS y nos situamos sobre el primer elemento.
boolean	**prsPreviousPage**() Se va al anterior elemento de una búsqueda PRS dentro de una página.
void	**setBuscar**(int _id) Se realiza una búsqueda de una pregunta del Foro
void	**setBuscarRespuesta**(int _id) Se busca una respuesta que se corresponda con su identificador
void	**setBuscarRespuestas**(int _id, boolean _dependePregunta) Se busca una respuestas que depende de otra respuesta ó de una pregunta.
void	**setPrsNoNext**() Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda PRS al realizar el siguiente Next o alguna de sus variantes.
boolean	**topPrsPage**() Nos ponemos sobre el primer elemento de la página sobre la que estamos situados en la búsqueda PRS

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ForoBean
public **ForoBean**()
Realiza las conexiones oportunas a la base de datos

Method Detail

isVacio
public boolean **isVacio**()
Para saber si se ha devuelto algún resultado de alguna búsqueda (PRS)
Returns:
True -> No esta vacio False -> Está vacío

---

getCountForoPreguntas
public int **getCountForoPreguntas**()

Para saber cuantas preguntas hay en el sistema.
Returns:
El número de preguntas que hay insertadas en el foro.

---

borrar

```
public boolean borrar(boolean _registrado,
                      boolean _cliente,
                      boolean _administrador,
                      boolean _revisor,
                      java.sql.Date _desde,
                      java.sql.Date _hasta)
```

Borra las preguntas del foro que se corresponda con los parámetros introducidos, si un parámetro no es introducido se ignorará a la hora de realizar el borrado
Parameters:
`_desde` - Fecha desde la cual tienen que estar publicada la pregunta del foro para eliminarla
`_hasta` - Fecha desde la cual tienen que estar publicada la pregunta del foro para eliminarla
Returns:
True -> Se ha realizado la operación correctamente False -> No se ha realizado correctamente

---

eliminarFila

```
public boolean eliminarFila(int _fila)
```

Para eliminar un fila de una búsqueda PRS y de la base de datos
Parameters:
`_fila` - La fila que se elimina
Returns:
true -> Se ha eliminado False -> No se ha podido eliminar

---

getNumeroFila

```
public int getNumeroFila()
```

Devuelve el número de fila que corresponde al registro sobre el cual estamos situados de la búsqueda PRS.
Returns:
El número de la fila. -1 indica que se ha comentido algún error

---

setPrsNoNext

```
public void setPrsNoNext()
```

Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda PRS al realizar el siguiente Next o alguna de sus variantes. Muy util si se cambia de página y no queremos que en la primera iteración del Next realice un avance. Asi podemos tratar a todos los elementos de la misma forma en bucle while. Ya que al realizar el bucle se saltaria al hacer el next el elemento sobre el cual estamos situados (Al pasar de página nunca podemos estar antes del primer elemento,siempre nos dejará en el primero).

---

topPrsPage

```
public boolean topPrsPage()
```

Nos ponemos sobre el primer elemento de la página sobre la que estamos situados en la búsqueda PRS

---

prsCerrar

```
public void prsCerrar()
```

Cerrar la consulta devuelta en páginas

---

desconectar

```
public void desconectar()
```

Cierra las consultas y se desconecta de la base de datos

---

insertar

```
public boolean insertar(java.lang.String _titulo,
                        java.lang.String _cuerpo,
```

```
                        java.lang.String _nick,
                        java.sql.Date _fecha,
                        int _dirigido)
```

Se inserta en el Foro una Pregunta

Parameters:

`_titulo` - Es el Título de la tupla que se va a insertar

`_cuerpo` - Es el Cuerpo de la tupla que se va a insertar

`_nick` - Es el Nick de la tupla que se va a insertar

`_fecha` - Es la Fecha de la tupla

`_dirigido` - Es el Dirigido de la tupla

Returns:

True -> Se ha insertado False -> No se ha podido insertar

---

insertarRespuesta

```
public boolean insertarRespuesta(int _id,
                                 java.lang.String _titulo,
                                 java.lang.String _cuerpo,
                                 java.lang.String _nick,
                                 java.sql.Date _fecha)
```

Se inserta en el Foro una Respuesta asociada a una pregunta.

Parameters:

`_id` - Es el IDPregunta es decir es un entero que se asocia con la pregunta de la que depende ForoBean, por tanto este número tiene que estar antes asociado en una tupla de la tabla ForoPreguntas

`_titulo` - Es el Título de la tupla que se va a insertar en la tabla

`_cuerpo` - Es el Cuerpo de la tupla que se va a insertar en la tabla

`Date` - Es la Fecha que se va a insertar en la tupla que se va a insertar en la tabla

Returns:

True -> Se ha insertado con éxito la tupla False -> La tupla no se ha insertado con éxito

---

insertarRespuesta

```
public boolean insertarRespuesta(int _id,
                                 int _id2,
                                 java.lang.String _titulo,
                                 java.lang.String _cuerpo,
                                 java.lang.String _nick,
                                 java.sql.Date _fecha)
```

Se inserta en el Foro una respuesta asociada con una pregunta y una respuesta.

Parameters:

`_id` - Es el IDPregunta es decir es un entero que se asocia con la pregunta de la que depende ForoRespuestas, por tanto este número tiene que estar asociado en una tupla de la tabla ForoPreguntas

`_id2` - Es el IDRespuesta es decir es un entero que se asocia con otra respuesta de la que depende ForoRespuestas, por este número tiene que estar asociado a una tupla de la tabla ForoRespuestas

`_titulo` - Es el Título de la tupla que se va a insertar en la tabla

`_cuerpo` - Es el Cuerpo de la tupla que se va a insertar en la tabla

`Date` - Es la Fecha que se va a insertar en la tupla que se va a insertar en la tabla

Returns:

True -> Se ha insertado con éxito la tupla False -> La tupla no se ha insertado con éxito

---

getNumResp

```
public int getNumResp(int _id)
```

Para saber cuantas respuestas estan asociadas con la pregunta que se identifica en el parámetro de entrada

Parameters:

`_id` - Entero que identifica a la tupla de ForoPregunta de la cual queremos saber cuantas tuplas tenemos asociadas con ella

Returns:

El número de tuplas asociadas que contiene la tabla.

---

getNumResp2

```
public int getNumResp2(int _id)
```

Para saber cuantas respuestas estan asociadas con una Respuesta que se identifica en el parámetro de entrada

Parameters:

`_id` - Entero que identifica a la tupla de ForoRespuestas de la cual queremos saber cuantas tuplas tenemos asociadas con ella

Returns:

El número de tuplas asociadas que contiene la tabla.

---

setBuscar

```
public void setBuscar(int _id)
```

Se realiza una búsqueda de una pregunta del Foro

Parameters:

`_id` - Se corresponde al identificador de la pregunta

---

buscar

```
public void buscar(java.lang.String _palabra,
                   int _orden,
                   int _usuario,
                   int _orden2)
```

Se realiza una búqueda de preguntas del Foro y se almacena en páginas (Busqueda PRS)

Parameters:

`_palabra` - Se realiza la búsqueda de todas las tuplas que contengan esa palabra en su Titulo o Cuerpo. Si tiene null 0 cadena Vacia se busca ignorando que contenga Titulo y Cuerpo

`_orden` - 1 -> Se ordena por la Fecha 2 -> Se ordena por el Titulo 3 -> Se ordena por el nick (otro valor se ignora el orden)

`_usuario` - Se buscan solo las tuplas que tengan como Usuario el que le indicamos en _usuario

`_orden2` - 1 -> Orden ascendente 2 -> Orden descendente

---

buscarAvanzada

```
public void buscarAvanzada(java.lang.String _palabra,
                           java.lang.String _nick,
                           java.sql.Date desde,
                           java.sql.Date hasta,
                           int _orden,
                           int _usuario,
                           int _orden2)
```

Se realiza una búqueda avanza de preguntas del foro y se almacena en páginas (Búsqueda PRS)

Parameters:

`_palabra` - Se realiza la búsqueda de todas las tuplas que contengan esa palabra en su Titulo o Cuerpo. Si tiene null 0 cadena Vacia se busca ignorando que contenga Titulo y Cuerpo

`_nick` - Se realiza la búsqueda de todas las tuplas con Nick igual a _nick. Si _nick es null o cadena vacia se ignora este campo en la búsqueda

`_desde` - Se buscarán las tuplas con Fecha mayor a la fecha contenida en _desde, si es null se ignora este campo de búsqueda

`_hasta` - Se buscarán las tuplas con Fecha menor a la fecha indicada en _hasta, si es null se ignora este campo de búsqueda

`_orden` - 1 -> Se ordena por la Fecha 2 -> Se ordena por el Titulo 3 -> Se ordena por el nick (otro valor se ignora el orden)

`_usuario` - Se buscan solo las tuplas que tengan como Usuario el que le indicamos en _usuario

`_orden2` - 1 -> Orden ascendente 2 -> Orden descendente

---

setBuscarRespuesta

```
public void setBuscarRespuesta(int _id)
```

Se busca una respuesta que se corresponda con su identificador

Parameters:

_id - Es el identificador de la respuesta

---

setBuscarRespuestas

```
public void setBuscarRespuestas(int _id,
                                 boolean _dependePregunta)
```

Se busca una respuestas que depende de otra respuesta ó de una pregunta.

Parameters:

_id - Es el identificador de la respuesta o pregunta de la cual tienen que depender

_dependePregunta - Si la respuestas buscadas dependen de una pregunta true o de otra respuesta false

---

prsFirst

```
public boolean prsFirst()
```

Se va al primer elemento del resultado de una búsqueda PRS.

Returns:

True -> Se ha podido ir al primer elemento False -> No se ha podido ir al primer elemento

---

prsNextInPage

```
public boolean prsNextInPage()
```

Se va al siguiente elemento de una búsqueda PRS dentro de una página.

Returns:

True -> Hay elemento posterior y nos hemos situados sobre el False -> No hay elemento posterior

---

prsPreviousPage

```
public boolean prsPreviousPage()
```

Se va al anterior elemento de una búsqueda PRS dentro de una página.

Returns:

True -> Hay elemento anterior y nos hemos situados sobre el False -> No hay elemento anterior

---

prsNextPage

```
public boolean prsNextPage()
```

Se va a la siguiente página de la búsqueda PRS y nos situamos sobre el primer elemento.

Returns:

True -> Hay siguiente página y nos hemos situado en su primer elemento False -> No hay siguiente página

---

prsFirstPage

```
public boolean prsFirstPage()
```

Se va a la primera página de la búsqueda PRS y nos situamos sobre el primer elemento.

Returns:

True -> Hay primer página y nos hemos situado en su primer elemento False -> No hay ninguna página

---

prsLastPage

```
public boolean prsLastPage()
```

Se va a la última página de la búsqueda PRS y nos situamos sobre su primer elemento.

Returns:

True -> Nos hemos situado en la última página y en su primer elemento False -> No hay ninguna página

---

next

```
public boolean next()
```

Pasamos al siguiente elemento de una búsqueda

Returns:

True -> Hay siguiente elemento y nos hemos situado sobre el False -> No hemos podido ir al siguiente elemento

---

getPages

```
public java.lang.String getPages()
```

Para saber cuantas páginas ha devuelto la búsqueda PRS
Returns:
No devuelve el número de páginas en un String

---

isPaginaAnterior
`public boolean isPaginaAnterior()`
Para saber si hay alguna página antes sobre la cual estamos situados en una búsqueda PRS
Returns:
True -> Hay una página anterior False -> No hay ninguna página anterior

---

isPaginaSiguiente
`public boolean isPaginaSiguiente()`
Para saber si hay alguna página despues sobre la cual estamos situados en una búsqueda PRS
Returns:
True -> Hay una página False -> No la hay

---

getPage
`public java.lang.String getPage()`
Para saber sobre que página estamos situados en una búsqueda PRS
Returns:
Nos devuelve un String indicándonos el número de la página sobre la que estamos

---

getPrsId
`public java.lang.String getPrsId()`
Para coseguir el Identificador de una pregunta sobre la la que estamos situados en una búsqueda PRS.
Returns:
Nos devulve un entero con el atributo ID de la pregunta

---

getPrsTitulo
`public java.lang.String getPrsTitulo()`
Para coseguir el Titulo sobre la pregunta sobre la que estamos situados en una búsqueda PRS.
Returns:
Nos devulve el Titulo

---

getTitulo
`public java.lang.String getTitulo()`
Para saber el Titulo del elemento sobre el que estamos en una busqueda
Returns:
Nos devulve el Titulo de la tupla

---

getCuerpo
`public java.lang.String getCuerpo()`
Para saber el Cuerpo del elemento sobre el que estamos en una busqueda
Returns:
Nos devulve el Cuerpo de la tupla

---

getNick
`public java.lang.String getNick()`
Para conocer el nick del elemento sobre el que estamos en una busqueda
Returns:
Nos devuelve el nick

---

getId
`public int getId()`
Para saber el identificador del elemento sobre el que estamos en una busqueda
Returns:
Nos devuelve el identificador

getIdPregunta
`public int` **`getIdPregunta`**`()`
Para saber el identificador de la pregunta de la cual depende una respuesta
Returns:
Nos devuelve el identificador

getFecha
`public java.sql.Date` **`getFecha`**`()`
Para conocer la fecha que hay en el registro actual de la fecha
Returns:
Nos devuelve la fecha

getPrsCuerpo
`public java.lang.String` **`getPrsCuerpo`**`()`
Para coseguir el Cuerpo sobre la tupla en la que estamos situados en una búsqueda PRS.
Returns:
Nos devulve el Cuerpo

getPrsNick
`public java.lang.String` **`getPrsNick`**`()`
Para coseguir el Nick sobre la tupla en la que estamos situados en una búsqueda PRS.
Returns:
Nos devulve el Nick

getPrsFecha
`public java.lang.String` **`getPrsFecha`**`()`
Para coseguir la Fecha sobre la tupla en la que estamos situados en una búsqueda PRS.
Returns:
Nos devulve la Fecha

finalize
`protected void` **`finalize`**`()`
`throws java.lang.Throwable`
Finaliza el objeto, cierra las sesiones y desconecta el objeto de la base de datos si aun sigue conectado y con las consultas abiertas
Overrides:
`finalize` in class `java.lang.Object`

tablas

Class GustosBean

```
java.lang.Object
  |
  +--tablas.GustosBean
```

public class **GustosBean**
extends java.lang.Object
Clase que sirve para acceder a la base de datos y que gestiona la tabla Gustos. Cuando se realice una búsqueda habrá que cerrar las consultas.

Constructor Summary
GustosBean() Realiza la conexión con la base de datos

Method Summary

void	**cerrar**() Se Cierra la consulta con la base de datos si esta sigue aun abierta
void	**desconectar**() Se desconecta de la base de datos y se cierran las consultas.
int	**getIdGusto**() Se devuelve el identificador del gusto sobre el cual estamos situados después de una búsqueda.
int	**getIdGusto**(java.lang.String _gusto) Se devuelve el identificador respecto al gusto pasado como parámetro
boolean	**getNext**() Se pasa a la siguiente tupla de la búsqueda si es que existe.
java.lang.String	**getNombreGusto**() Se obtiene el nombre del gusto sobre el que estamos situados después de realizar una búsqueda
java.lang.String	**getNombreGusto**(int _id) Se obtiene el nombre del gusto pasando su identificador como parámetro
boolean	**Insertar**(java.lang.String _gusto) Inserta un Gusto en la tabla de la base de datos
boolean	**setBuscar**(int _id) Se busca el gusto que tenga como identificador el pasado como parámetro de datos queda abierta y deberá de ser cerrada
boolean	**setBuscar**(java.lang.String _gusto) Se busca el gusto que se corresponde con el nombre pasado como parámetro
void	**setBuscarTodos**() Se buscan todas los gustos contenidos en la base de datos
boolean	**setEliminar**(int _id) Se elimina el gusto que se corersponde con el parámetro pasado

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

GustosBean

```
public GustosBean()
```

Realiza la conexión con la base de datos

Method Detail

Insertar

```
public boolean Insertar(java.lang.String _gusto)
```

Inserta un Gusto en la tabla de la base de datos

Parameters:

_gusto - Es el contenido del campo Gusto que se va insertar de la tupla

Returns:

True -> El gusto se ha insertado correctamente False -> No se ha insertado correctamente

---

getNext

```
public boolean getNext()
```

Se pasa a la siguiente tupla de la búsqueda si es que existe. Además cuando se llegue al final se cierra la consulta con la base de datos.
Returns:
True -> Se ha podido pasar a la siguiente tupla False -> Pues no se ha podido pasar a la siguiente tupla

---

getIdGusto
`public int` **`getIdGusto`**`()`
Se devuelve el identificador del gusto sobre el cual estamos situados después de una búsqueda.
Returns:
Un entero que es el identificador de la tupla

---

getIdGusto
`public int` **`getIdGusto`**`(java.lang.String _gusto)`
Se devuelve el identificador respecto al gusto pasado como parámetro
Parameters:
`_gusto` - Es el valor del campo Gusto.
Returns:
Es un entero que identifica a la tupla

---

getNombreGusto
`public java.lang.String` **`getNombreGusto`**`(int _id)`
Se obtiene el nombre del gusto pasando su identificador como parámetro
Parameters:
`_id` - Se corresponde con el valor del campo ID
Returns:
Devuelve el contenido de la tupla asociada con el valor pasado

---

getNombreGusto
`public java.lang.String` **`getNombreGusto`**`()`
Se obtiene el nombre del gusto sobre el que estamos situados después de realizar una búsqueda
Returns:
El contenido del campo Gusto de la tupla sobre la cual estamos situados.

---

setEliminar
`public boolean` **`setEliminar`**`(int _id)`
Se elimina el gusto que se corersponde con el parámetro pasado
Parameters:
`_id` - Se corresponde con el valor ID del identificador del gusto
Returns:
True -> Se ha borrado False -> No se ha podido borrar

---

setBuscar
`public boolean` **`setBuscar`**`(int _id)`
Se busca el gusto que tenga como identificador el pasado como parámetro de datos queda abierta y deberá de ser cerrada
Parameters:
`_id` - Es el identificador de la tupla
Returns:
True -> Se ha efectuado la búsqueda False -> No se ha podido llevar a cabo

---

setBuscarTodos
`public void` **`setBuscarTodos`**`()`
Se buscan todas los gustos contenidos en la base de datos

---

setBuscar
`public boolean` **`setBuscar`**`(java.lang.String _gusto)`

Se busca el gusto que se corresponde con el nombre pasado como parámetro
Parameters:
`_gusto` - Se corresponde con el valor contenido en el campo Gusto.
Returns:
True -> Se ha efectuado la búsqueda correctamente False -> No se ha efectuado correctamente

---

cerrar
`public void` **`cerrar`**`()`
Se Cierra la consulta con la base de datos si esta sigue aun abierta

---

desconectar
`public void` **`desconectar`**`()`
Se desconecta de la base de datos y se cierran las consultas.

tablas

Class InfoPortalBean

```
java.lang.Object
  |
  +--tablas.InfoPortalBean
```

All Implemented Interfaces:
java.io.Serializable

---

public class **InfoPortalBean**
extends java.lang.Object
implements java.io.Serializable
Clase que sirve para acceder a la base de datos y que gestiona la tabla InfoPortal
See Also:
Serialized Form

Constructor Summary
`InfoPortalBean``()` Realiza la conexión con la base de datos

Method Summary	
`void`	**`desconectar`**`()` Realiza la desconexión con la base de datos
`java.lang.String`	**`getCuerpo`**`()` Para obtener el campo Cuerpo del resultado de la búsqueda
`java.lang.String`	**`getTitulo`**`()` Para obtener el campo Titulo del resultado de la búsqueda
`void`	**`setBuscar`**`(java.lang.String _id)` Se realiza una búsqueda en la tabla de la tupla que tenga por ID el pasado por parámetro.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

InfoPortalBean

`public` **`InfoPortalBean`**`()`
Realiza la conexión con la base de datos

Method Detail

desconectar
`public void` **`desconectar`**`()`
Realiza la desconexión con la base de datos

setBuscar
`public void` **`setBuscar`**`(java.lang.String _id)`
Se realiza una búsqueda en la tabla de la tupla que tenga por ID el pasado por parámetro. Almacenamos en el objeto el resultado de la búsqueda.
Parameters:
`_id` - Es el entero que queremos que tenga el resultado de la búsqueda en su campo ID

getTitulo
`public java.lang.String` **`getTitulo`**`()`
Para obtener el campo Titulo del resultado de la búsqueda
Returns:
El valor contenido en el campo Titulo

getCuerpo
`public java.lang.String` **`getCuerpo`**`()`
Para obtener el campo Cuerpo del resultado de la búsqueda
Returns:
El valor contenido en el campo Cuerpo

tablas

Class InfoVariosBean

```
java.lang.Object
  |
  +--tablas.InfoVariosBean
```

All Implemented Interfaces:
java.io.Serializable

public class **InfoVariosBean**
extends java.lang.Object
implements java.io.Serializable
Clase que sirve para acceder a la base de datos y que gestiona la tabla InfoVarios.
See Also:
Serialized Form

Constructor Summary
`InfoVariosBean``()` Realiza la conexión con la base de datos

Method Summary	
`void`	**`desconectar`**`()` Realiza la desconexión con la base de datos
`java.lang.String`	**`getCuerpo`**`()` Para obtener el campo Cuerpo del resultado de la búsqueda
`java.lang.`	**`getTitulo`**`()`

String	Para obtener el campo Titulo del resultado de la búsqueda
void	**setBuscar**(java.lang.String _id) Se realiza una búsqueda en la tabla de la tupla que tenga por ID el pasado por parámetro.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

InfoVariosBean
public **InfoVariosBean**()
Realiza la conexión con la base de datos

Method Detail

desconectar
public void **desconectar**()
Realiza la desconexión con la base de datos

---

setBuscar
public void **setBuscar**(java.lang.String _id)
Se realiza una búsqueda en la tabla de la tupla que tenga por ID el pasado por parámetro. Almacenamos en el objeto el resultado de la búsqueda.
Parameters:
_id - Es el entero que queremos que tenga el resultado de la búsqueda en su campo ID

---

getTitulo
public java.lang.String **getTitulo**()
Para obtener el campo Titulo del resultado de la búsqueda
Returns:
El valor contenido en el campo Titulo

---

getCuerpo
public java.lang.String **getCuerpo**()
Para obtener el campo Cuerpo del resultado de la búsqueda
Returns:
El valor contenido en el campo Cuerpo

tablas

Class JobbiesBean

java.lang.Object
  |
  +--**tablas.JobbiesBean**

---

public class **JobbiesBean**
extends java.lang.Object
Clase que sirve para acceder a la base de datos y que gestiona la tabla Jobbies

---

Constructor Summary
JobbiesBean() Realiza la conexión con la base de datos

Method Summary	
void	**desconectar**() Se desconecta de la base de datos y se cierran todas las consultas.
int	**getJobbie**() Pasamos si hay a la siguiente tupla (Resultado de alguna búsqueda) y devovemos el valor del campo Jobbie.
boolean	**Insertar**(java.lang.String _eMail, int _jobbie) Inserta una tupla en la tabla Jobbies
boolean	**isJobbie**(java.lang.String _eMail, int _jobbie) Comprobamos si hay una tupla con los campos Jobbie y eMail iguales a los pasados por parámetro.
boolean	**setBuscar**(java.lang.String _eMail) Se buscan todas las tuplas que esten asociadas con el valor del parámetro en su campo eMail.
boolean	**setEliminar**(java.lang.String _eMail) Eliminamos todas las tuplas que en su campo eMail contenga el valor pasado como parámetro

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

JobbiesBean

```
public JobbiesBean()
```

Realiza la conexión con la base de datos

Method Detail

Insertar

```
public boolean Insertar(java.lang.String _eMail,
                        int _jobbie)
```

Inserta una tupla en la tabla Jobbies

Parameters:

_eMail - Se corresponde al campo eMail y tiene que estar asociado a la existencia de alguna tupla con este mismo eMail en la tabla Usuario

_jobbie - entero que esta asociado con la existencia de una tupla en Gustos con el mismo valor en IDGusto

Returns:

True -> Se ha insertado correctamente False -> No se ha insertado correctamente

---

isJobbie

```
public boolean isJobbie(java.lang.String _eMail,
                        int _jobbie)
```

Comprobamos si hay una tupla con los campos Jobbie y eMail iguales a los pasados por parámetro. Nos sirve para saber si un Usuario esta relacionado con un gusto determinado.

Parameters:

_eMail - Se corresponde al eMail de algún usuario

_jobbie - Es el identificador de un Gusto

Returns:

True -> Existe la tupla con esos dos valores False -> No existe

---

getJobbie
`public int` **`getJobbie`**`()`
Pasamos si hay a la siguiente tupla (Resultado de alguna búsqueda) y devovemos el valor del campo Jobbie.
Returns:
Es el valor contenido del campo Jobbie, nos devuelve 0 si ha ocurrido algún error o no hay más tuplas.

---

setEliminar
`public boolean` **`setEliminar`**`(java.lang.String _eMail)`
Eliminamos todas las tuplas que en su campo eMail contenga el valor pasado como parámetro
Parameters:
`_eMail` - Contiene el valor del campo eMail de las tuplas de las cuales se quieren borrar.
Returns:
True -> Se ha borrado satisfactoriamente False -> No se ha borrado correctamente

---

setBuscar
`public boolean` **`setBuscar`**`(java.lang.String _eMail)`
Se buscan todas las tuplas que esten asociadas con el valor del parámetro en su campo eMail. La consulta a la base de datos no se cierra, deberá de ser cerrada.
Parameters:
`_eMail` - Valor del cual se quieren buscar todas las tuplas donde su campo eMail sea igual a este.
Returns:
True -> La operación se efectuado correctamente False -> No se ha efectuado correctamente

---

desconectar
`public void` **`desconectar`**`()`
Se desconecta de la base de datos y se cierran todas las consultas.

tablas

Class NotasBean

```
java.lang.Object
  |
  +--tablas.NotasBean
```

All Implemented Interfaces:
java.io.Serializable

---

public class **NotasBean**
extends java.lang.Object
implements java.io.Serializable
Clase que sirve para acceder a la base de datos y que gestiona las tablas Notas y NotasAnonimas. En definitiva gestiona los datos almacenados en la base de datos sobre el envío de notas de la aplicación. Los tipos de usuarios que pueden enviar notas 1 -> Administrador 2-> cliente 3-> Revisor 4 -> Usuario No registrado 5 -> Cliente que envia a todos los administradores 6 -> Revisor que envia a todos los administradores 7 -> Usuario Registrado que envía a todos los administradores que envía a todos los usuarios , esta clase se puede utilizar como un JavaBean
See Also:
Serialized Form

Constructor Summary
`NotasBean``()` Realiza la conexión con la base de datos

Method Summary

void	**desconectar**() Se desconecta la base de datos, se deberá de utilizar este método cada vez que creemos un objeto
boolean	**eliminar**(int _id, int _usuario) Elimina una nota (tupla) que se corresponda con los parámetros introducidos
int	**getCountSinContestar**(java.lang.String _eMail, int _usuario) Para saber cuantas tuplas hay en notas y notasAdministradores sin contestar.
int	**getCountSinLeer**(java.lang.String _eMail, int _usuario) Para saber cuantas tuplas hay en notas y notasAdministradores sin leer por un administrador.
java.lang.String	**getCuerpo**() Devuelve el campo Cuerpo de la nota (tupla) sobre la cual estamos situados de la búsqueda
java.lang.String	**getEMail**(int _id, int _usuario) Para conseguir el Correo electrónico de quien envio una nota
java.lang.String	**getEmailRemitente**() Devuelve el contenido del campo eMailRemitente de una búsqueda
java.lang.String	**getFecha**() Devuelve una cadena que representa el campo Fecha contenido en la tupla sobre la cual estamos situados
int	**getId**() Devuelve el identificador ID de la tupla sobre la cual estamos situados de la búsqueda y que nos identifica la nota
boolean	**getNext**() Pasa a la siguiente tupla de una búsqueda.
java.lang.String	**getNombreUsuarioRemitente**(int _usuarioRemitente) Nos devuelve el nombre del tipo de usuario que se introduce como parámetro y se corresponde con su rol
Java.lang.String	**getTitulo**() Devuelve el Titulo de la nota (tupla) sobre la cual estamos situados de la búsqueda
int	**getUsuarioRemitente**() Devuelve el valor del campo UsuarioRemitente contenido en la tupla sobre la que estamos de la búsqueda y nos identifica el rol del usuario que realizó la nota
boolean	**insertar**(java.lang.String _remitente, int _usuarioDestinatario, java.lang.String _titulo, java.lang.String _cuerpo, java.sql.Date _fecha) Se inserta una nota anónima en el sistema, estas van dirigidas a los Administradores del sistema.
boolean	**insertar**(java.lang.String _eMailDestinatario, java.lang.String _eMailRemitente, int _usuarioDestinatario, int _usuarioRemitente, java.lang.String _titulo, java.lang.String _cuerpo, java.sql.Date _fecha) Se inserta una nota en el sistema
boolean	**isContestado**() Devuelve el valor del campo Contestado contenido en la tupla sobre la cual estamos situados después de una búsqueda.

boolean	**isLeido**() Devuelve el valor del campo Leido contenido en la tupla sobre la cual estamos situados, despues de una búsqueda.
boolean	**setBuscar**(int _id, int _usuario) Busca una nota (tupla) que se corresponda con el ID y el usuarioRemitente que la realizó
void	**setBuscarAdminin**(java.lang.String _eMail) Busca las tuplas de la tabla NotasAdministrador y Notas.
void	**setBuscarNotas**(java.lang.String _eMail, int _usuario) Busca las tuplas de la tabla Notas que están dirigidas a un usuario con un determinado tipo de rol y las dirigidas específicamente al Administrador que se introduce por parámetro
void	**setContestada**(int _id, int _usuario) Marca la nota como Contestada

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

NotasBean

```
public NotasBean()
```

Realiza la conexión con la base de datos

Method Detail

insertar

```
public boolean insertar(java.lang.String _eMailDestinatario,
                        java.lang.String _eMailRemitente,
                        int _usuarioDestinatario,
                        int _usuarioRemitente,
                        java.lang.String _titulo,
                        java.lang.String _cuerpo,
                        java.sql.Date _fecha)
```

Se inserta una nota en el sistema

Parameters:

_eMail - Correo electrónico del usuario que realiza una nota, por tanto debe de existir un usuario en la tabla usuario con el mismo eMail

_titulo - Es el titulo correspondiente a la sugerencia

_cuerpo - Es el cuerpo de la sugerencia

_fecha - Es la fecha en la que se hizo la sugerencia

Returns:

True -> Se ha insertado en la tabla False -> No se ha insertado en la tabla

---

insertar

```
public boolean insertar(java.lang.String _remitente,
                        int _usuarioDestinatario,
                        java.lang.String _titulo,
                        java.lang.String _cuerpo,
                        java.sql.Date _fecha)
```

Se inserta una nota anónima en el sistema, estas van dirigidas a los Administradores del sistema.

Parameters:

_remitente - Correo electrónico del usuario que realiza la sugerencia como es anonima no hace falta que este insertado previamente en la tabla usuarios.

_titulo - Es el titulo correspondiente a la sugerencia
_cuerpo - Es el cuerpo de la sugerencia
_fecha - Es la fecha en la que se hizo la sugerencia
_usuarioDestinatario - Es el tipo de usuario al que va dirigido (Ya que un usuario puede desempeñar varios roles)
Returns:
True -> Se ha insertado en la tabla False -> No se ha insertado en la tabla

---

setBuscarAdminin

```
public void setBuscarAdminin(java.lang.String _eMail)
```

Busca las tuplas de la tabla NotasAdministrador y Notas. Están dirigidas a los administradores (para todos) o las dirigidas específicamente al Administrador que se introduce por parámetro
Parameters:
_eMail - Es el eMail del administrador

---

setBuscarNotas

```
public void setBuscarNotas(java.lang.String _eMail,
                           int _usuario)
```

Busca las tuplas de la tabla Notas que están dirigidas a un usuario con un determinado tipo de rol y las dirigidas específicamente al Administrador que se introduce por parámetro
Parameters:
_eMail - Es el eMail del usuario
_usuario - Es un entero que representa el rol del usuario 1 -> Administrador 2-> cliente 3-> Revisor 4 -> Usuario No registrado 5 -> Cliente que envia a todos los administradores 6 -> Revisor que envia a todos los administradores 7 -> Usuario Registrado que envía a todos los administradores

---

getEMail

```
public java.lang.String getEMail(int _id,
                                 int _usuario)
```

Para conseguir el Correo electrónico de quien envio una nota
Parameters:
_id - identificador de la nota
_usuario - Es el rol del usuario que hizo la nota
Returns:
el correo electrónico

---

setContestada

```
public void setContestada(int _id,
                          int _usuario)
```

Marca la nota como Contestada
Parameters:
_id - identificador de la nota
_usuario - Es el rol del usuario que hizo la nota

---

setBuscar

```
public boolean setBuscar(int _id,
                         int _usuario)
```

Busca una nota (tupla) que se corresponda con el ID y el usuarioRemitente que la realizó
Parameters:
_id - identificador de la tupla
_usuario - El rol del usuario que hizo la nota
Returns:
True-> Se ha encontrado una tupla False -> no se ha encontrado

---

eliminar

```
public boolean eliminar(int _id,
```

```
                                int _usuario)
```
Elimina una nota (tupla) que se corresponda con los parámetros introducidos
Parameters:
`_id` - identificador de la tupla
`_usuario` - El rol del usuario que hizo la nota
Returns:
True-> Se ha eliminado False -> no se ha eliminado

---

getNext
`public boolean` **`getNext`**`()`
Pasa a la siguiente tupla de una búsqueda.
Returns:
True -> Hay siguiente elemento y se ha situado sobre el False -> No se ha podido ir al siugiente elemento o no lo hay

---

getEmailRemitente
`public java.lang.String` **`getEmailRemitente`**`()`
Devuelve el contenido del campo eMailRemitente de una búsqueda
Returns:
El email

---

getFecha
`public java.lang.String` **`getFecha`**`()`
Devuelve una cadena que representa el campo Fecha contenido en la tupla sobre la cual estamos situados
Returns:
la fecha en el formato dd/mm/año

---

isLeido
`public boolean` **`isLeido`**`()`
Devuelve el valor del campo Leido contenido en la tupla sobre la cual estamos situados, despues de una búsqueda.
Returns:
El valor contenido en el campo Leido

---

isContestado
`public boolean` **`isContestado`**`()`
Devuelve el valor del campo Contestado contenido en la tupla sobre la cual estamos situados después de una búsqueda.
Returns:
El valor contenido en el campo Contestado

---

getUsuarioRemitente
`public int` **`getUsuarioRemitente`**`()`
Devuelve el valor del campo UsuarioRemitente contenido en la tupla sobre la que estamos de la búsqueda y nos identifica el rol del usuario que realizó la nota
Returns:
El contenido del UsuarioRemitente 0 --> Error

---

getNombreUsuarioRemitente
`public java.lang.String` **`getNombreUsuarioRemitente`**`(int _usuarioRemitente)`
Nos devuelve el nombre del tipo de usuario que se introduce como parámetro y se corresponde con su rol
Returns:
El nombre del tipo de usuario que pertenecía cuando realizó la nota

---

getId
`public int` **`getId`**`()`

Devuelve el identificador ID de la tupla sobre la cual estamos situados de la búsqueda y que nos identifica la nota
Returns:
El identificador 0 -> Error

---

getTitulo
`public java.lang.String` **`getTitulo`**`()`
Devuelve el Titulo de la nota (tupla) sobre la cual estamos situados de la búsqueda
Returns:
El Campo Titulo almacenado en la tupla

---

getCuerpo
`public java.lang.String` **`getCuerpo`**`()`
Devuelve el campo Cuerpo de la nota (tupla) sobre la cual estamos situados de la búsqueda
Returns:
El Campo Cuerpo almacenado en la tupla

---

getCountSinLeer
```
public int getCountSinLeer(java.lang.String _eMail,
                           int _usuario)
```
Para saber cuantas tuplas hay en notas y notasAdministradores sin leer por un administrador.
Parameters:
`_eMail` - Es el identificador del Administrador
`_usuario` - Para ver las notas del usuario que despeña el rol: ADMINISTRADOR, CLIENTE, REVISOR
Returns:
El número de tuplas que contiene las 2 tablas y han sido contestadas.

---

getCountSinContestar
```
public int getCountSinContestar(java.lang.String _eMail,
                                int _usuario)
```
Para saber cuantas tuplas hay en notas y notasAdministradores sin contestar.
Parameters:
`_usuario` - Para ver las notas del usuario que despeña el rol: ADMINISTRADOR, CLIENTE, REVISOR
Returns:
El número de tuplas que contiene las 2 tablas y no han sido contestadas.

---

desconectar
`public void` **`desconectar`**`()`
Se desconecta la base de datos, se deberá de utilizar este método cada vez que creemos un objeto

tablas

Class NoticiasBean

```
java.lang.Object
  |
  +--tablas.NoticiasBean
```
All Implemented Interfaces:
java.io.Serializable

---

public class **NoticiasBean**
extends java.lang.Object
implements java.io.Serializable
Clase que implementa el acceso a la Tabla Noticias y por tanto gestiona las noticias que se inserten en el sistema. Tiene búsquedas PRS que permiten albergar los datos de la búsqueda en páginas.

See Also:
Serialized Form

Constructor Summary
NoticiasBean() Realiza las conexiones oportunas a la base de datos

Method Summary	
boolean	**borrar**(boolean _noRegistrado, boolean _registrado, boolean _cliente, boolean _administrador, boolean _revisor, java.sql.Date _desde, java.sql.Date _hasta) Borra noticias según los parámetros introducidos, si uno de ellos no es introducido se ignorará a la hora de hacer el borrado.
void	**buscar**(java.lang.String _palabra, int _orden, int _orden2, boolean _cliente, boolean _administrador, boolean _revisor) Se realiza una búqueda PRS de noticias y se almacenan todas las tuplas
void	**buscarAvanzada**(java.lang.String _palabra, java.sql.Date desde, java.sql.Date hasta, int _orden, int _orden2, boolean _noRegistrado, boolean _registrado, boolean _administrador, boolean _cliente, boolean _revisor) Se realiza una búqueda PRS de noticias y se almacenan todas las tuplas
void	**desconectar**() Se desconecta de la base de datos y cierra las consultas
boolean	**eliminarFila**(int _fila) Para eliminar un fila de una búsqueda PRS, también elimina la tupla de la base de datos
protected void	**finalize**() Cuanod se elimina el objeto nos aseguramos de que se desconecte de la base de datos y que cierre las consultas
boolean	**first**() Se va al primer elemento del resultado de una búsqueda PRS.
boolean	**firstPage**() Se va a la primera página de la búsqueda PRS y nos situamos sobre el primer elemento.
int	**getCountNoticias**() Para saber cuantas Noticias hay en el sistema.
java.lang.String	**getCuerpo**() Para coseguir el Cuerpo sobre la noticia en la que estamos situados, en una búsqueda PRS.
java.lang.String	**getCuerpoNoticia**() Recogemos el valor del campo Cuerpo sobre la noticia que estamos situados.
java.lang.String	**getDestinatario**() Para obtener una cadena que forma mediante los campos booleanos y agregando el campo que esta a true en la cadena.
java.lang.	**getDestinatarioNoticia**()

String	Nos devuelve en una cadena los distintos usuarios al que va dirigiada la noticia, es decir mira los campos booleanos y confeciona la cadena dependiendo si está o no activados estos campos.
Java.lang.String	**getFecha**() Para coseguir la Fecha sobre la noticia en la que estamos situados, en una búsqueda PRS.
java.lang.String	**getFechaNoticia**() Recogemos el valor del campo Fecha sobre la noticia que estamos situados.
int	**getNumeroFila**() Devuelve el número de fila que corresponde al registro sobre el cual estamos situados (Busquedas PRS).
java.lang.String	**getPage**() Para saber sobre que páginas estamos situados en una búsqueda PRS
java.lang.String	**getPages**() Para saber cuantas páginas ha devuelto la búsqueda PRS
java.lang.String	**getTextoCuerpo**() Para saber el Cuerpo de una noticia, en una búsqueda
java.lang.String	**getTextoTitulo**() Para coseguir el Titulo sobre la noticia en la que estamos situados, en una búsqueda PRS.
java.lang.String	**getTitulo**() Para coseguir el ID sobre la noticia en la que estamos situados, en una búsqueda PRS.
java.lang.String	**getTituloNoticia**() Recogemos el valor del campo Titulo sobre la noticia que estamos situados.
boolean	**insertar**(java.lang.String _titulo, java.lang.String _cuerpo, java.sql.Date _fecha, boolean _noRegistrado, boolean _registrado, boolean _cliente, boolean _administrador, boolean _revisor) Se inserta una Noticia en el sistema
boolean	**isPaginaAnterior**() Para saber si hay alguna página antes sobre la cual estamos situados en una búsqueda PRS
boolean	**isPaginaSiguiente**() Para saber si hay alguna página despues sobre la cual estamos situados en una búsqueda PRS
boolean	**isVacio**() Para saber si hay alguna noticia devuelta de una búsqueda (PRS)
boolean	**lastPage**() Se va a la última página de la búsqueda PRS y nos situamos sobre su primer elemento.
boolean	**next**() Pasamos al siguiente elemento de la búsqueda PRS sin importarnos en que página se encuentra la siguiente noticia
boolean	**nextInPage**() Se va al siguiente elemento de una búsqueda PRS dentro de una página.

boolean	**nextNoticia**() Pasamos al siguiente elemento de la búsqueda.
boolean	**nextPage**() Se va a la siguiente página de la búsqueda PRS y nos situamos sobre el primer elemento.
boolean	**previousPage**() Se va al anterior elemento de una búsqueda PRS dentro de una página.
void	**setBuscarNoticias**() Se buscan todas las noticias que estén dirigidas a usuario NoRegistrados y se ordena por Fecha descendentemente Con esta búsqueda funcionan los métodos getNextNoticia, getTituloNoticia, getCuerpoNoticia, getDestinatarioNoticia y getFechaNoticia.
void	**setBuscarNoticias2**(boolean cliente, boolean administrador, boolean revisor) Se realiza una búsqueda y se almacena los resultados en el propio objeto.
void	**setNoNext**() Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda (PRS) al realizar el siguiente Next o alguna de sus variantes.
boolean	**topPage**() Nos ponemos sobre el primer elemento de la página sobre la que estamos situados (Búsquedas PRS)

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

NoticiasBean

```
public NoticiasBean()
```

Realiza las conexiones oportunas a la base de datos

Method Detail

borrar

```
public boolean borrar(boolean _noRegistrado,
                      boolean _registrado,
                      boolean _cliente,
                      boolean _administrador,
                      boolean _revisor,
                      java.sql.Date _desde,
                      java.sql.Date _hasta)
```

Borra noticias según los parámetros introducidos, si uno de ellos no es introducido se ignorará a la hora de hacer el borrado.

Parameters:

_noRegistrado - La noticia si esta dirigida a un usuario no registrado podrá ser borrada
_cliente - La noticia si esta dirigida a un cliente podrá ser borrada
_administrador - La noticia si esta dirigida a un administrador podrá ser borrada
_desde - Fecha desde la cual tienen que estar publicada la noticia para eliminarla
_hasta - Fecha desde la cual tienen que estar publicada la noticia para eliminarla

Returns:

True -> Se ha realizado la operación correctamente False -> No se ha realizado correctamente

---

getCountNoticias

`public int` **`getCountNoticias`**`()`
Para saber cuantas Noticias hay en el sistema.
Returns:
El número de tuplas que contiene la tabla.

---

isVacio
`public boolean` **`isVacio`**`()`
Para saber si hay alguna noticia devuelta de una búsqueda (PRS)
Returns:
True -> No esta vacio False -> Está vacío

---

eliminarFila
`public boolean` **`eliminarFila`**`(int _fila)`
Para eliminar un fila de una búsqueda PRS, también elimina la tupla de la base de datos
Parameters:
`_fila` - La fila que se elimina
Returns:
true -> Se ha eliminado False -> No se ha podido eliminar

---

getNumeroFila
`public int` **`getNumeroFila`**`()`
Devuelve el número de fila que corresponde al registro sobre el cual estamos situados (Busquedas PRS).
Returns:
El número de la fila. -1 indica que se ha comentido algún error

---

setNoNext
`public void` **`setNoNext`**`()`
Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda (PRS) al realizar el siguiente Next o alguna de sus variantes. Muy util si se cambia de página y no queremos que en la primera iteración del Next realice un avance. Asi podemos tratar a todos los elementos de la misma forma en bucle while. Ya que al realizar el bucle se saltaria al hacer el next el elemento sobre el cual estamos situados (Al pasar de página nunca podemos estar antes del primer elemento,siempre nos dejará en el primero).

---

topPage
`public boolean` **`topPage`**`()`
Nos ponemos sobre el primer elemento de la página sobre la que estamos situados (Búsquedas PRS)

---

desconectar
`public void` **`desconectar`**`()`
Se desconecta de la base de datos y cierra las consultas

---

insertar
```
public boolean insertar(java.lang.String _titulo,
                        java.lang.String _cuerpo,
                        java.sql.Date _fecha,
                        boolean _noRegistrado,
                        boolean _registrado,
                        boolean _cliente,
                        boolean _administrador,
                        boolean _revisor)
```
Se inserta una Noticia en el sistema
Parameters:
`_titulo` - Es el valor del campo Título de la tupla que se va a insertar
`_cuerpo` - Es el valor del campo Cuerpo de la tupla que se va a insertar
`_fecha` - Es el valor del campo Fecha de la tupla que se va a insertar
`_noRegistrado` - Es el valor del campo noRegistrado de la tupla que se va a insertar
`_Registrado` - Es el valor del campo Registrado de la tupla que se va a insertar

_cliente - Es el valor del campo cliente de la tupla que se va a insertar
_Administrador - Es el valor del campo Administrador de la tupla que se va a insertar
_revisor - Es el valor del campo revisor de la tupla que se va a insertar
Returns:
True -> Se ha insertado en la tabla False -> No se ha podido insertar en la tabla

---

setBuscarNoticias2

```
public void setBuscarNoticias2(boolean cliente,
                               boolean administrador,
                               boolean revisor)
```

Se realiza una búsqueda y se almacena los resultados en el propio objeto. Se buscan las tuplas que cumplan alguno de los parámetros introducidos, estos parámetros se corresponden al valor de distintos campos de la tabla. Se ordenan las tuplas por el campo Fecha descendentemente. Además realiza una consulta a la base de datos que deberá ser cerrada. Con esta búsqueda funcionan los métodos getNextNoticia, getTituloNoticia, getCuerpoNoticia, getDestinatarioNoticia y getFechaNoticia. No se machacan los datos con la busqueda normal método buscar.
Parameters:
cliente - Se buscan todas las tuplas que tenga el campo cliente al valor introducido.
administrador - Se buscan todas las tuplas que tenga el campo Administrador al valor introducido.
revisor - Se buscan todas las tuplas que tengan el campo revisor al valor introducido.

---

setBuscarNoticias

```
public void setBuscarNoticias()
```

Se buscan todas las noticias que estén dirigidas a usuario NoRegistrados y se ordena por Fecha descendentemente Con esta búsqueda funcionan los métodos getNextNoticia, getTituloNoticia, getCuerpoNoticia, getDestinatarioNoticia y getFechaNoticia. No se machacan los datos con la busqueda normal método buscar. Habrá que cerrar la consulta

---

nextNoticia

```
public boolean nextNoticia()
```

Pasamos al siguiente elemento de la búsqueda. Se utiliza con los métodos setBuscarNoticias y setBuscarNoticias2. Si no hay más elementos cierra la consulta con la base de datos.
Returns:
True -> Hay siguiente elemento y nos hemos situado sobre el False -> No hemos podido ir al siguiente elemento

---

getTituloNoticia

```
public java.lang.String getTituloNoticia()
```

Recogemos el valor del campo Titulo sobre la noticia que estamos situados. Se utiliza con los métodos setBuscarNoticias y setBuscarNoticias2.
Returns:
El contenido del campo, cadena vacía si ha habido algún error

---

getCuerpoNoticia

```
public java.lang.String getCuerpoNoticia()
```

Recogemos el valor del campo Cuerpo sobre la noticia que estamos situados. Se utiliza con los métodos setBuscarNoticias y setBuscarNoticias2.
Returns:
El valor contenido en el campo. Devolvemos la cadena vacía si ha habido algún error.

---

getDestinatarioNoticia

```
public java.lang.String getDestinatarioNoticia()
```

Nos devuelve en una cadena los distintos usuarios al que va dirigiada la noticia, es decir mira los campos booleanos y confeciona la cadena dependiendo si está o no activados estos campos. Se utiliza con los métodos setBuscarNoticias y setBuscarNoticias2.
Returns:

True -> Hay siguiente elemento y nos hemos situado sobre el False -> No hemos podido ir al siguiente elemento

---

getDestinatario

```
public java.lang.String getDestinatario()
```

Para obtener una cadena que forma mediante los campos booleanos y agregando el campo que esta a true en la cadena. Esta es para las búsquedas PRS

Returns:

una cadena compuesta de los distitons usuario al que va dirigido la tupla

---

getFechaNoticia

```
public java.lang.String getFechaNoticia()
```

Recogemos el valor del campo Fecha sobre la noticia que estamos situados. Se utiliza con los métodos setBuscarNoticias y setBuscarNoticias2.

Returns:

Nos devuelve una cadena con la fecha contenida en el campo Fecha

---

buscar

```
public void buscar(java.lang.String _palabra,
                   int _orden,
                   int _orden2,
                   boolean _cliente,
                   boolean _administrador,
                   boolean _revisor)
```

Se realiza una búqueda PRS de noticias y se almacenan todas las tuplas

Parameters:

`_palabra` - Se realiza la búsqueda de todas las noticias que contengan esa palabra en su Titulo o Cuerpo. Si tiene null ó cadena Vacia se busca ignorando que contenga Titulo y Cuerpo

`_orden` - 1 -> Se ordena por la Fecha 2 -> Se ordena por el Titulo (otro valor se ignora el orden)

`_orden2` - 1 -> Orden ascendente 2 -> Orden descendente

`cliente` - Tendrá tener el campo Cliente al valor pasado

`administrador` - Tendrá que tener el campo Administrador al valor pasado

`Revisor` - Tendrá que tener el campo revisor al valor pasado

---

buscarAvanzada

```
public void buscarAvanzada(java.lang.String _palabra,
                           java.sql.Date desde,
                           java.sql.Date hasta,
                           int _orden,
                           int _orden2,
                           boolean _noRegistrado,
                           boolean _registrado,
                           boolean _administrador,
                           boolean _cliente,
                           boolean _revisor)
```

Se realiza una búqueda PRS de noticias y se almacenan todas las tuplas

Parameters:

`_palabra` - Se realiza la búsqueda de todas las noticias que contengan esa palabra en su Titulo o Cuerpo. Si tiene null ó cadena Vacia se busca ignorando que contenga Titulo y Cuerpo

`_desde` - Se buscarán las tuplas con Fecha mayor a la fecha contenida en _desde, si es null se ignora este campo de búsqueda

`_hasta` - Se buscarán las tuplas con Fecha menor a la fecha indicada en _hasta, si es null se ignora este campo de búsqueda

`_orden` - 1 -> Se ordena por la Fecha 2 -> Se ordena por el Titulo (otro valor se ignora el orden)

`_orden2` - 1 -> Orden ascendente 2 -> Orden descendente

`_noRegistrado` - Tendrá que tener el campo noRegistrado al valor pasado

`_registrado` - Tendrá que tener el campo registrado al valor pasado

`cliente` - Tendrá tener el campo Cliente al valor pasado
`administrador` - Tendrá que tener el campo Administrador al valor pasado
`revisor` - Tendrá que tener el campo revisor al valor pasado

---

first
`public boolean` **`first`**`()`
Se va al primer elemento del resultado de una búsqueda PRS.
Returns:
True -> Se ha podido ir al primer elemento False -> No se ha podido ir al primer elemento

---

nextInPage
`public boolean` **`nextInPage`**`()`
Se va al siguiente elemento de una búsqueda PRS dentro de una página.
Returns:
True -> Hay elemento posterior y nos hemos situados sobre el False -> No hay elemento posterior

---

previousPage
`public boolean` **`previousPage`**`()`
Se va al anterior elemento de una búsqueda PRS dentro de una página.
Returns:
True -> Hay elemento anterior y nos hemos situados sobre el False -> No hay elemento anterior

---

nextPage
`public boolean` **`nextPage`**`()`
Se va a la siguiente página de la búsqueda PRS y nos situamos sobre el primer elemento.
Returns:
True -> Hay siguiente página y nos hemos situado en su primer elemento False -> No hay siguiente página

---

firstPage
`public boolean` **`firstPage`**`()`
Se va a la primera página de la búsqueda PRS y nos situamos sobre el primer elemento.
Returns:
True -> Hay primer página y nos hemos situado en su primer elemento False -> No hay ninguna página

---

lastPage
`public boolean` **`lastPage`**`()`
Se va a la última página de la búsqueda PRS y nos situamos sobre su primer elemento.
Returns:
True -> Nos hemos situado en la última página y en su primer elemento False -> No hay ninguna página

---

next
`public boolean` **`next`**`()`
Pasamos al siguiente elemento de la búsqueda PRS sin importarnos en que página se encuentra la siguiente noticia
Returns:
True -> Hay siguiente elemento y nos hemos situado sobre el False -> No hemos podido ir al siguiente elemento

---

getPages
`public java.lang.String` **`getPages`**`()`
Para saber cuantas páginas ha devuelto la búsqueda PRS
Returns:
No devuelve el número de páginas en un String

---

isPaginaAnterior
`public boolean` **`isPaginaAnterior`**`()`

Para saber si hay alguna página antes sobre la cual estamos situados en una búsqueda PRS
Returns:
True -> Hay una página anterior False -> No hay ninguna página anterior

---

isPaginaSiguiente
`public boolean` **`isPaginaSiguiente`**`()`
Para saber si hay alguna página despues sobre la cual estamos situados en una búsqueda PRS
Returns:
True -> Hay una página False -> No la hay

---

getPage
`public java.lang.String` **`getPage`**`()`
Para saber sobre que páginas estamos situados en una búsqueda PRS
Returns:
Nos devuelve un String indicándonos el número de la página sobre la que estamos

---

getTitulo
`public java.lang.String` **`getTitulo`**`()`
Para coseguir el ID sobre la noticia en la que estamos situados, en una búsqueda PRS.
Returns:
Nos devulve un entero con el contenido de ID de la tupla

---

getTextoTitulo
`public java.lang.String` **`getTextoTitulo`**`()`
Para coseguir el Titulo sobre la noticia en la que estamos situados, en una búsqueda PRS.
Returns:
Nos devulve el Titulo

---

getTextoCuerpo
`public java.lang.String` **`getTextoCuerpo`**`()`
Para saber el Cuerpo de una noticia, en una búsqueda
Returns:
Nos devulve el Cuerpo de la tupla

---

getCuerpo
`public java.lang.String` **`getCuerpo`**`()`
Para coseguir el Cuerpo sobre la noticia en la que estamos situados, en una búsqueda PRS.
Returns:
Nos devulve el Cuerpo

---

getFecha
`public java.lang.String` **`getFecha`**`()`
Para coseguir la Fecha sobre la noticia en la que estamos situados, en una búsqueda PRS.
Returns:
Nos devulve la Fecha

---

finalize
`protected void` **`finalize`**`()`
`throws java.lang.Throwable`
Cuanod se elimina el objeto nos aseguramos de que se desconecte de la base de datos y que cierre las consultas
Overrides:
`finalize` in class `java.lang.Object`

tablas

Class PagesRowSet

```
java.lang.Object
  |
  +--sun.jdbc.rowset.BaseRowSet
        |
        +--sun.jdbc.rowset.CachedRowSet
              |
              +--tablas.PagesRowSet
```

All Implemented Interfaces:
java.lang.Cloneable, java.sql.ResultSet, javax.sql.RowSet, javax.sql.RowSetInternal, java.io.Serializable

---

public class **PagesRowSet**
extends sun.jdbc.rowset.CachedRowSet
implements java.io.Serializable
Clase que implementa una iteración integral con la base de datos. Permite su conexión y desconexión y realizar busquedas que iran almancenan en el propio objeto el resultado de la búsqueda. Permitimos ir a cualquier tupla en cualquier orden, además gestionamos las distintas tuplas en páginas, para realizar una visualización más fácil en forma de distintas páginas que muestran la información. Se puede comportar como un JavaBean
See Also:
Serialized Form

Fields inherited from class sun.jdbc.rowset.BaseRowSet
ASCII_STREAM_PARAM, asciiStream, BINARY_STREAM_PARAM, binaryStream, charStream, UNICODE_STREAM_PARAM, unicodeStream

Fields inherited from interface java.sql.ResultSet
CONCUR_READ_ONLY, CONCUR_UPDATABLE, FETCH_FORWARD, FETCH_REVERSE, FETCH_UNKNOWN, TYPE_FORWARD_ONLY, TYPE_SCROLL_INSENSITIVE, TYPE_SCROLL_SENSITIVE

Constructor Summary	
PagesRowSet()	

Method Summary	
boolean	**absolutePage**(int _page) Se va a la página indicada en el parámetro y nos situamos en su primer elemento.
void	**close**() Cierra la conexión con la base de datos y inicializa el objeto.
void	**deleteRow**() Elimina la celda sobre la que estamos situados dentro de la búsqueda
void	**execute**() Sirve para ejecutar una setencia sql que se ha introducido previamente
boolean	**first**() Se va al primer elemento del resultado de una búsqueda.
boolean	**firstInPage**() Se va al primer elemento de la página sobre la cual estamos
boolean	**firstPage**()

	Se va a la primera página de la búsqueda y nos situamos sobre el primer elemento.
int	**getActualPage**() Nos dice el número de la página sobre la cual estamos situados
int	**getNumberElementsInPage**() Nos dice el número de elementos que forma cada página
int	**getTotalPages**() Nos dice el número de páginas que tenemos (del resultado de la última búsqueda)
boolean	**isNextPage**() Para saber si hay alguna página despues sobre la cual estamos situados
boolean	**isPreviusPage**() Para saber si hay alguna página antes sobre la cual estamos situados
boolean	**isVacio**() Para saber si hay alguna tupla contenida en el objeto (Se supone que se ha realizado alguna búsqueda)
boolean	**last**() Se va al último elemento del resultado de una búsqueda.
boolean	**lastInPage**() Se va al último elemento de la página sobre la cual estamos situados.
boolean	**lastPage**() Se va a la última página de la búsqueda y nos situamos sobre su primer elemento.
boolean	**next**() Pasamos al siguiente elemento sin importarnos en que página se encuentra
boolean	**nextInPage**() Se va al siguiente elemento dentro de una página.
boolean	**nextPage**() Se va a la siguiente página de la búsqueda y nos situamos sobre el primer elemento.
boolean	**previous**() Pasamos al anterior elemento sin importarnos en que página se encuentra
boolean	**previousInPage**() Se va al anterior elemento dentro de una página.
boolean	**previousPage**() Se va al anterior elemento de una búsqueda dentro de una página.
void	**setNoNext**() Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda al realizar el siguiente Next o alguna de sus variantes.
void	**setNumberElementsInPage**(int _number) Para que nos divida los resultados en páginas con igual número de elementos en cada una, como el número introducido
boolean	**topPage**() Nos ponemos sobre el primer elemento de la página sobre la que estamos situados

Methods inherited from class sun.jdbc.rowset.CachedRowSet
absolute, acceptChanges, acceptChanges, afterLast, beforeFirst, cancelRowDelete, cancelRowInsert, cancelRowUpdates, clearWarnings, clone, columnUpdated, createCopy, createShared, execute, findColumn, getArray, getArray, getAsciiStream, getAsciiStream, getBigDecimal,

getBigDecimal, getBigDecimal, getBigDecimal, getBinaryStream, getBinaryStream, getBlob, getBlob, getBoolean, getBoolean, getByte, getByte, getBytes, getBytes, getCharacterStream, getCharacterStream, getClob, getClob, getConnection, getCurrentRow, getCursorName, getDate, getDate, getDate, getDate, getDouble, getDouble, getFloat, getFloat, getInt, getInt, getKeyColumns, getLong, getLong, getMetaData, getObject, getObject, getObject, getObject, getOriginal, getOriginalRow, getReader, getRef, getRef, getRow, getShort, getShort, getStatement, getString, getString, getTableName, getTime, getTime, getTime, getTime, getTimestamp, getTimestamp, getTimestamp, getTimestamp, getUnicodeStream, getUnicodeStream, getWarnings, getWriter, insertRow, internalFirst, internalLast, internalNext, internalPrevious, isAfterLast, isBeforeFirst, isFirst, isLast, moveToCurrentRow, moveToInsertRow, populate, refreshRow, relative, release, removeCurrentRow, restoreOriginal, rowDeleted, rowInserted, rowUpdated, setCommand, setKeyColumns, setMetaData, setOriginal, setOriginalRow, setReader, setTableName, setWriter, size, toCollection, toCollection, updateAsciiStream, updateAsciiStream, updateBigDecimal, updateBigDecimal, updateBinaryStream, updateBinaryStream, updateBoolean, updateBoolean, updateByte, updateByte, updateBytes, updateBytes, updateCharacterStream, updateCharacterStream, updateDate, updateDate, updateDouble, updateDouble, updateFloat, updateFloat, updateInt, updateInt, updateLong, updateLong, updateNull, updateNull, updateObject, updateObject, updateObject, updateObject, updateRow, updateShort, updateShort, updateString, updateString, updateTime, updateTime, updateTimestamp, updateTimestamp, wasNull

Methods inherited from class sun.jdbc.rowset.BaseRowSet

addRowSetListener, clearParameters, getCommand, getConcurrency, getDataSourceName, getEscapeProcessing, getFetchDirection, getFetchSize, getMaxFieldSize, getMaxRows, getParams, getPassword, getQueryTimeout, getShowDeleted, getTransactionIsolation, getType, getTypeMap, getUrl, getUsername, initParams, isReadOnly, notifyCursorMoved, notifyRowChanged, notifyRowSetChanged, removeRowSetListener, setArray, setAsciiStream, setBigDecimal, setBinaryStream, setBlob, setBoolean, setByte, setBytes, setCharacterStream, setClob, setConcurrency, setDataSourceName, setDate, setDate, setDouble, setEscapeProcessing, setFetchDirection, setFetchSize, setFloat, setInt, setLong, setMaxFieldSize, setMaxRows, setNull, setNull, setObject, setObject, setObject, setPassword, setQueryTimeout, setReadOnly, setRef, setShort, setShowDeleted, setString, setTime, setTime, setTimestamp, setTimestamp, setTransactionIsolation, setType, setTypeMap, setUnicodeStream, setUrl, setUsername

Methods inherited from class java.lang.Object

equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface javax.sql.RowSet

addRowSetListener, clearParameters, getCommand, getDataSourceName, getEscapeProcessing, getMaxFieldSize, getMaxRows, getPassword,

getQueryTimeout, getTransactionIsolation, getTypeMap, getUrl, getUsername, isReadOnly, removeRowSetListener, setArray, setAsciiStream, setBigDecimal, setBinaryStream, setBlob, setBoolean, setByte, setBytes, setCharacterStream, setClob, setConcurrency, setDataSourceName, setDate, setDate, setDouble, setEscapeProcessing, setFloat, setInt, setLong, setMaxFieldSize, setMaxRows, setNull, setNull, setObject, setObject, setObject, setPassword, setQueryTimeout, setReadOnly, setRef, setShort, setString, setTime, setTime, setTimestamp, setTimestamp, setTransactionIsolation, setType, setTypeMap, setUrl, setUsername

Methods inherited from interface java.sql.ResultSet
getConcurrency, getFetchDirection, getFetchSize, getType, setFetchDirection, setFetchSize

Methods inherited from interface javax.sql.RowSetInternal
getParams

Constructor Detail

PagesRowSet

```
public PagesRowSet()
            throws java.sql.SQLException
```

Method Detail

deleteRow

```
public void deleteRow()
               throws java.sql.SQLException
```

Elimina la celda sobre la que estamos situados dentro de la búsqueda

Overrides:

deleteRow in class sun.jdbc.rowset.CachedRowSet

Throws:

java.sql.SQLException -

---

isVacio

```
public boolean isVacio()
```

Para saber si hay alguna tupla contenida en el objeto (Se supone que se ha realizado alguna búsqueda)

Returns:

True -> No esta vacio False -> Está vacío

---

close

```
public void close()
            throws java.sql.SQLException
```

Cierra la conexión con la base de datos y inicializa el objeto.

Overrides:

close in class sun.jdbc.rowset.CachedRowSet

Throws:

java.sql.SQLException -

---

isPreviusPage

```
public boolean isPreviusPage()
```

Para saber si hay alguna página antes sobre la cual estamos situados

Returns:

True -> Hay una página anterior False -> No hay ninguna página anterior

---

isNextPage
`public boolean isNextPage()`
Para saber si hay alguna página despues sobre la cual estamos situados
Returns:
True -> Hay una página False -> No la hay

---

setNumberElementsInPage
`public void setNumberElementsInPage(int _number)`
Para que nos divida los resultados en páginas con igual número de elementos en cada una, como el número introducido
Parameters:
`_number` - Es el número de elementos que queremos que disponga cada una de las páginas

---

setNoNext
`public void setNoNext()`
`throws java.sql.SQLException`
Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda al realizar el siguiente Next o alguna de sus variantes. Muy util si se cambia de página y no queremos que en la primera iteración del Next realice un avance. Asi podemos tratar a todos los elementos de la misma forma en bucle while. Ya que al realizar el bucle se saltaria al hacer el next el elemento sobre el cual estamos situados (Al pasar de página nunca podemos estar antes del primer elemento,siempre nos dejará en el primero).
Throws:
`java.sql.SQLException` -

---

getTotalPages
`public int getTotalPages()`
Nos dice el número de páginas que tenemos (del resultado de la última búsqueda)

---

getActualPage
`public int getActualPage()`
Nos dice el número de la página sobre la cual estamos situados

---

getNumberElementsInPage
`public int getNumberElementsInPage()`
Nos dice el número de elementos que forma cada página

---

execute
`public void execute()`
`throws java.sql.SQLException`
Sirve para ejecutar una setencia sql que se ha introducido previamente
Overrides:
`execute` in class `sun.jdbc.rowset.CachedRowSet`
Throws:
`java.sql.SQLException` -

---

nextPage
`public boolean nextPage()`
`throws java.sql.SQLException`
Se va a la siguiente página de la búsqueda y nos situamos sobre el primer elemento.
Returns:
True -> Hay siguiente página y nos hemos situado en su primer elemento False -> No hay siguiente página
Throws:
`java.sql.SQLException` -

---

topPage
`public boolean topPage()`

```
                throws java.sql.SQLException
```
Nos ponemos sobre el primer elemento de la página sobre la que estamos situados
Returns:
true -> Se ha efecutado correctamente la operacion false -> caso contrario
Throws:
`java.sql.SQLException` -

---

previousPage
```
public boolean previousPage()
                     throws java.sql.SQLException
```
Se va al anterior elemento de una búsqueda dentro de una página.
Returns:
True -> Hay elemento anterior y nos hemos situados sobre el False -> No hay elemento anterior
Throws:
`java.sql.SQLException` -

---

absolutePage
```
public boolean absolutePage(int _page)
                     throws java.sql.SQLException
```
Se va a la página indicada en el parámetro y nos situamos en su primer elemento.
Parameters:
`_page` - Indica la página sobre la cual queremos situarnos
Returns:
True -> Hay elemento anterior y nos hemos situados sobre el False -> No hay elemento anterior
Throws:
`java.sql.SQLException` -

---

firstInPage
```
public boolean firstInPage()
                    throws java.sql.SQLException
```
Se va al primer elemento de la página sobre la cual estamos
Returns:
True -> Hay elemento anterior y nos hemos situados sobre el False -> No hay elemento
Throws:
`java.sql.SQLException` -

---

lastInPage
```
public boolean lastInPage()
                   throws java.sql.SQLException
```
Se va al último elemento de la página sobre la cual estamos situados.
Returns:
True -> Hay elemento anterior y nos hemos situados sobre el False -> No hay elemento
Throws:
`java.sql.SQLException` -

---

firstPage
```
public boolean firstPage()
                  throws java.sql.SQLException
```
Se va a la primera página de la búsqueda y nos situamos sobre el primer elemento.
Returns:
True -> Hay primer página y nos hemos situado en su primer elemento False -> No hay ninguna página
Throws:
`java.sql.SQLException` -

---

lastPage
```
public boolean lastPage()
                throws java.sql.SQLException
```

Se va a la última página de la búsqueda y nos situamos sobre su primer elemento.
Returns:
True -> Nos hemos situado en la última página y en su primer elemento False -> No hay ninguna página
Throws:
`java.sql.SQLException` -

---

nextInPage

```
public boolean nextInPage()
                    throws java.sql.SQLException
```

Se va al siguiente elemento dentro de una página.
Returns:
True -> Hay elemento posterior y nos hemos situados sobre el False -> No hay elemento posterior
Throws:
`java.sql.SQLException` -

---

previousInPage

```
public boolean previousInPage()
                        throws java.sql.SQLException
```

Se va al anterior elemento dentro de una página.
Returns:
True -> Hay elemento anterior y nos hemos situados sobre el False -> No hay elemento anterior
Throws:
`java.sql.SQLException` -

---

next

```
public boolean next()
             throws java.sql.SQLException
```

Pasamos al siguiente elemento sin importarnos en que página se encuentra
Overrides:
`next` in class `sun.jdbc.rowset.CachedRowSet`
Returns:
True -> Hay siguiente elemento y nos hemos situado sobre el False -> No hemos podido ir al siguiente elemento
Throws:
`java.sql.SQLException` -

---

previous

```
public boolean previous()
                throws java.sql.SQLException
```

Pasamos al anterior elemento sin importarnos en que página se encuentra
Overrides:
`previous` in class `sun.jdbc.rowset.CachedRowSet`
Returns:
True -> Hay anterior elemento y nos hemos situado sobre el False -> No hemos podido ir al siguiente elemento
Throws:
`java.sql.SQLException` -

---

first

```
public boolean first()
              throws java.sql.SQLException
```

Se va al primer elemento del resultado de una búsqueda.
Overrides:
`first` in class `sun.jdbc.rowset.CachedRowSet`
Returns:
True -> Se ha podido ir al primer elemento False -> No se ha podido ir al primer elemento

---

last

```
public boolean last()
            throws java.sql.SQLException
```

Se va al último elemento del resultado de una búsqueda.

Overrides:

`last` in class `sun.jdbc.rowset.CachedRowSet`

Returns:

True -> Se ha podido ir al último elemento False -> No se ha podido ir al último elemento

tablas

Class PaisesBean

```
java.lang.Object
  |
  +--tablas.PaisesBean
```

---

public class **PaisesBean**
extends java.lang.Object
Clase que sirve para acceder a la base de datos y que gestiona la tabla Paises.

Constructor Summary
PaisesBean() Realiza la conexión con la base de datos

Method Summary	
void	**cerrar**() Cierra la consulta que pueda estar abierta con la base de datos
void	**desconectar**() Desconecta de la base de datos.
int	**getIdPais**() Se devuelve el Identificador del pais sobre el cual estamos situados dentro de la tabla
int	**getIdPais**(java.lang.String _pais) Se devuelve el Identificador del pais del cual hemos introducido su nombre como parámetro
boolean	**getNext**() Se pasa a la siguiente tupla de la búsqueda (si es que la hay), si no hay además cierra las consultas a la base de datos.
java.lang.String	**getNombrePais**() Se devuelve el nombre dle pais sobre el cual estamos situados.
java.lang.String	**getNombrePais**(int _id) Se devuelve el Nombre del país del cual hemos su identificador ID como parámetro
boolean	**Insertar**(java.lang.String _pais) Se inserta un pais en la tabla
void	**setBuscarConProvincias**() Se buscan todos los paises que tienen almenos una provincia asociada de la tabla provincias.

void	**setBuscarSinProvincias**() Se buscan todos los paises que no tienen ninguna provincia asociada de la tabla provincias.
void	**setBuscarTodos**() Se buscan todos los paises.
boolean	**setEliminar**(int _id) Se elimina el páis que tiene como identificador ID el pasado como parámetro

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

PaisesBean

public **PaisesBean**()

Realiza la conexión con la base de datos

Method Detail

Insertar

public boolean **Insertar**(java.lang.String _pais)

Se inserta un pais en la tabla

Parameters:

_pais - El pais que se va a insertar

Returns:

True -> El pais se ha insertado en la tabla False -> El pais no se ha insertado en la tabla

---

getNext

public boolean **getNext**()

Se pasa a la siguiente tupla de la búsqueda (si es que la hay), si no hay además cierra las consultas a la base de datos.

Returns:

True -> Se ha pasado a la siguiente tupla de la búsqueda False -> Estamos al final y no hay más tuplas o no se ha efectuado la operación

---

getIdPais

public int **getIdPais**()

Se devuelve el Identificador del pais sobre el cual estamos situados dentro de la tabla

Returns:

Devuelve el número el ID del pais.

---

getIdPais

public int **getIdPais**(java.lang.String _pais)

Se devuelve el Identificador del pais del cual hemos introducido su nombre como parámetro

Parameters:

_pais - Nombre del pais del cual queremos saber su identificador.

Returns:

Devuelve el número ID del pais.

---

getNombrePais

public java.lang.String **getNombrePais**(int _id)

Se devuelve el Nombre del país del cual hemos su identificador ID como parámetro

Parameters:

_id - Identificador del país.

Returns:
Devuelve el nombre del pais. Cadena vacía si no se ha encontrado ninguno con el identificador introducido.

---

getNombrePais
`public java.lang.String` **`getNombrePais`**`()`
Se devuelve el nombre dle pais sobre el cual estamos situados. La consulta sigue abierta una vez ejecutada este método por tanto deberá de ser cerrada.
Returns:
Devuelve el nombre del pais.

---

setEliminar
`public boolean` **`setEliminar`**`(int _id)`
Se elimina el páis que tiene como identificador ID el pasado como parámetro
Parameters:
`_id` - Identificador del pais.
Returns:
True -> Se ha efectuado correctamente la operación False -> Pues no.

---

setBuscarConProvincias
`public void` **`setBuscarConProvincias`**`()`
Se buscan todos los paises que tienen almenos una provincia asociada de la tabla provincias. La consulta a la base de datos queda abierta y portanto deberá de cerrarse.

---

setBuscarSinProvincias
`public void` **`setBuscarSinProvincias`**`()`
Se buscan todos los paises que no tienen ninguna provincia asociada de la tabla provincias. La consulta a la base de datos queda abierta y portanto deberá de cerrarse.

---

setBuscarTodos
`public void` **`setBuscarTodos`**`()`
Se buscan todos los paises. La consulta a la base de datos queda abierta y portanto deberá de cerrarse.

---

cerrar
`public void` **`cerrar`**`()`
Cierra la consulta que pueda estar abierta con la base de datos

---

desconectar
`public void` **`desconectar`**`()`
Desconecta de la base de datos. Tambien cierra las consultas que puedan etar abiertas.

tablas

Class ProfesionesBean

```
java.lang.Object
  |
  +--tablas.ProfesionesBean
```

---

public class **ProfesionesBean**
extends java.lang.Object
Clase que sirve para acceder a la base de datos y que gestiona la tabla Profesiones, esta clase se puede utilizar como un JavaBean

---

Constructor Summary

ProfesionesBean() Realiza la conexión con la base de datos

Method Summary	
void	**cerrar**() Se Cierra cualquier consulta que halla quedado abierta con la base de datos.
void	**desconectar**() Se Cierra cualquier consulta y la conexión a la base de datos
int	**getIdProfesion**() Se obtiene el ID de la profesión sobre la cual estamos situados.
int	**getIdProfesion**(java.lang.String _profesion) Se obtiene el ID de la profesion asociada con la que le pasamos no afecta a las búsquedas
boolean	**getNext**() Se pasa a la siguiente tupla de la búsqueda (si es que la hay), si se llega al final se cierra la consulta con la base de datos.
java.lang.String	**getNombreProfesion**() Se obtiene el Nombre de la profesión sobre la cual estamos situados.
java.lang.String	**getNombreProfesion**(int _id) Se obtiene el nombre de la profesion asociada al ID
void	**Insertar**(java.lang.String _profesion) Se inserta una profesión en la tabla.
void	**setBuscarTodos**() Se realiza una búsqueda en la cual estan todas las tuplas de Profesiones, la consulta con la base de datos no se cierra por tanto deberá de ser cerrado con posterioridad.
boolean	**setEliminar**(int _id) Se elimina una profesión asociada con el ID pasado como parámetro.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ProfesionesBean

```
public ProfesionesBean()
```

Realiza la conexión con la base de datos

Method Detail

Insertar

```
public void Insertar(java.lang.String _profesion)
              throws java.sql.SQLException
```

Se inserta una profesión en la tabla.

Parameters:

_profesion - La profesión que se va a insertar

Returns:

True -> Se ha insertado en la tabla False -> No se ha insertado en la tabla

---

getNext

`public boolean` **`getNext`**`()`
Se pasa a la siguiente tupla de la búsqueda (si es que la hay), si se llega al final se cierra la consulta con la base de datos.
Returns:
True -> Se ha pasado a la siguiente tupla de la búsqueda False -> Estamos al final y no hay más tuplas o no se ha efectuado la operación

---

getIdProfesion
`public int` **`getIdProfesion`**`()`
Se obtiene el ID de la profesión sobre la cual estamos situados. La consulta sigue quedando abierta, tendrá que ser cerrada.
Returns:
Devuelve el ID asociado, 0 si ha habido algún tipo de error

---

getIdProfesion
`public int` **`getIdProfesion`**`(java.lang.String _profesion)`
Se obtiene el ID de la profesion asociada con la que le pasamos no afecta a las búsquedas
Parameters:
`_profesion` - Se pasa la profesion de la cual queremos obtener su ID
Returns:
Devuelve el ID asociado, 0 si no esta contenido en la tabla

---

getNombreProfesion
`public java.lang.String` **`getNombreProfesion`**`(int _id)`
Se obtiene el nombre de la profesion asociada al ID
Parameters:
`_id` - Es el ID de la profesion de la cual queremos obtener su nombre
Returns:
Devuelve el nombre contenido en la tabla asociado con el ID=_id, si no hay ninguno asociado devuelve la cadena vacia

---

getNombreProfesion
`public java.lang.String` **`getNombreProfesion`**`()`
Se obtiene el Nombre de la profesión sobre la cual estamos situados. La consulta seguirá quedando abierta, tendrá que ser cerrada.
Returns:
Devuelve el ID asociado, 0 si ha habido algún tipo de error

---

setEliminar
`public boolean` **`setEliminar`**`(int _id)`
Se elimina una profesión asociada con el ID pasado como parámetro. No interfiere con las búsquedas.
Parameters:
`_id` - Es el identificador (ID) de la profesión que se quiere borrar.
Returns:
True -> Se ha borrado False -> No se ha podido borrar

---

setBuscarTodos
`public void` **`setBuscarTodos`**`()`
Se realiza una búsqueda en la cual estan todas las tuplas de Profesiones, la consulta con la base de datos no se cierra por tanto deberá de ser cerrado con posterioridad.
Returns:
True -> Se ha realizado la búsqueda con éxito False -> No se ha podido llevar a cabo la búsqueda

---

cerrar
`public void` **`cerrar`**`()`
Se Cierra cualquier consulta que halla quedado abierta con la base de datos.

---

desconectar
`public void` **`desconectar`**`()`
Se Cierra cualquier consulta y la conexión a la base de datos

tablas

Class ProvinciasBean

```
java.lang.Object
  |
  +--tablas.ProvinciasBean
```

public class **ProvinciasBean**
extends java.lang.Object
Clase que sirve para acceder a la base de datos y que gestiona la tabla Provincias

Constructor Summary
`ProvinciasBean``()` Realiza la conexión con la base de datos

Method Summary	
`void`	**`cerrar`**`()` Cierra las consultas que puedan estar abiertas con la base de datos.
`void`	**`desconectar`**`()` Desconecta la base de datos y cierra las consultas.
`int`	**`getIdProvincia`**`(int _idPais, java.lang.String _provincia)` Se devuelve el Identificador de la provincia que se corresponde con el nombre y está asociada a un determinado país.
`boolean`	**`getNext`**`()` Se pasa a la siguiente tupla de la búsqueda (si es que la hay).
`java.lang.String`	**`getNombreProvincia`**`(int _id)` Se devuelve el nombre de la provincia que se corresponde con el identificador.
`int`	**`getNumeroPaisesConProvincias`**`()` Para saber el número de paises que tienen provincias.
`int`	**`getNumeroProvincias`**`(int _idPais)` Para saber el número de provincias que estan asociadas a un pais,
`java.lang.String`	**`getProvincia`**`()` Se devuelve el nombre de la provincia sobre la cual estamos situados.
`boolean`	**`Insertar`**`(int _idPais, java.lang.String _provincia)` Se inserta una provincia en la tabla
`boolean`	**`setBuscar`**`(int _idPais)` Se buscan todas las provincias asociadas a un pais.
`boolean`	**`setEliminar`**`(int _id)` Se elimina la provincia que tiene como identificador ID el pasado como parámetro

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

ProvinciasBean

```
public ProvinciasBean()
```

Realiza la conexión con la base de datos

Method Detail

Insertar

```
public boolean Insertar(int _idPais,
                        java.lang.String _provincia)
```

Se inserta una provincia en la tabla

Parameters:

`_provincia` - El nombre de la provincia

`_idPais` - Es el identificador del pais con la cual la provincia está asociada.

Returns:

True -> Se ha insertado en la tabla False -> No se ha insertado en la tabla

---

getNext

```
public boolean getNext()
```

Se pasa a la siguiente tupla de la búsqueda (si es que la hay). Si se llega al final se cierra la consulta con la base de datos.

Returns:

True -> Se ha pasado a la siguiente tupla de la búsqueda False -> Estamos al final y no hay más tuplas o no se ha efectuado la operación

---

getIdProvincia

```
public int getIdProvincia(int _idPais,
                          java.lang.String _provincia)
```

Se devuelve el Identificador de la provincia que se corresponde con el nombre y está asociada a un determinado país.

Parameters:

`_idPais` - Es el identificador del pais con la cual esta asocida la provincia

`_provincia` - Es el nombre que tiene la provincia.

Returns:

Devuelve el número el ID del pais.

---

getNombreProvincia

```
public java.lang.String getNombreProvincia(int _id)
```

Se devuelve el nombre de la provincia que se corresponde con el identificador.

Parameters:

`_id` - Es el identificador de la provincia

Returns:

Devuelve el nombre de la provincia.

---

getProvincia

```
public java.lang.String getProvincia()
```

Se devuelve el nombre de la provincia sobre la cual estamos situados. La consulta sigue habierta una vez ejecutada este método por tanto deberá de ser cerrada.

Returns:

Devuelve el nombre de la provincia.

---

setEliminar

```
public boolean setEliminar(int _id)
```

Se elimina la provincia que tiene como identificador ID el pasado como parámetro

Parameters:

`_id` - Identificador de la provincia.

Returns:
True -> Se ha efectuado correctamente la operación False -> Pues no.

---

setBuscar
`public boolean` **`setBuscar`**`(int _idPais)`
Se buscan todas las provincias asociadas a un pais. La consulta con la base de datos queda abierta y deberá de ser cerrada.
Returns:
True -> Se ha efectuado la búsqueda correctamente False -> Pues no.

---

getNumeroProvincias
`public int` **`getNumeroProvincias`**`(int _idPais)`
Para saber el número de provincias que estan asociadas a un pais,
Parameters:
`_idPais` - Identificador del pais.
Returns:
Devuelve el número de provincias asociadas a un pais.

---

getNumeroPaisesConProvincias
`public int` **`getNumeroPaisesConProvincias`**`()`
Para saber el número de paises que tienen provincias.
Returns:
El número de paises que tienen provincias.

---

cerrar
`public void` **`cerrar`**`()`
Cierra las consultas que puedan estar abiertas con la base de datos.

---

desconectar
`public void` **`desconectar`**`()`
Desconecta la base de datos y cierra las consultas.

tablas

Class RevistaBean

```
java.lang.Object
  |
  +--tablas.RevistaBean
```

All Implemented Interfaces:
java.io.Serializable

---

public class **RevistaBean**
extends java.lang.Object
implements java.io.Serializable
Clase que gestiona las Revistas del sistema y permite búsquedas que devuelve los resultados en páginas, búsquedas PRS.
See Also:
Serialized Form

Constructor Summary
`RevistaBean``()` Realiza la conexión con la base de datos

Method Summary

void	**Actualizar**() Actualiza la búsqueda PRS, si se produce algún cambio y se quieren actualizar los cambios producidos
void	**beforeFirst**() Nos posicionamos antes del primer elemento de la búsqueda PRS
void	**buscar**(java.lang.String _nombre, java.lang.String _nick, int _ordenar, int _orden) Se realiza una búqueda PRS sólo se realiza la búsqueda si se introduce un parámetro Si no este se ignora al realizar la búsqueda.
void	**buscarSeccion**(int _id) Busca una seccion .
void	**buscarSecciones**(int _idRevi) Busca una serie de secciones que dependen directamente de una revista.
void	**buscarSeccionesDependientes**(int _idSecc) Busca una serie de secciones que dependen de otra seccion.
void	**buscarSolicitud**(java.lang.String _eMail, int _idRevi) Busca una solicitud para ser revisor y nos situa sobre el único resultado y nos situamos sobre ella Atributos de las tuplas devueltas en la búsqueda: IDRevi, eMail, Exposicion, Fecha
void	**buscarSolicitudes**(java.lang.String _eMail) Busca las solicitudes que hacen los usuarios para ser revisores de alguna de las revistas de un determinado usuario cliente Atributos de las tuplas devueltas en la búsqueda: IDRevi, eMail, Exposicion, Fecha
void	**buscarTopRevistas**(java.lang.String _palabra, boolean _tituloRevi, boolean _descripRevi, boolean _palabrasRevi, boolean _tituloSecc, boolean _descripSecc, boolean _palabrasSecc) Se realiza una búqueda PRS sólo se realiza la búsqueda si se introduce un parámetro si no este se ignora al realizar la búsqueda.
boolean	**cambiarDatos**(int _id, java.lang.String _nombre, java.lang.String _descripcion, int _estilo, boolean _promocionada) Cambia los datos que tenemos de una revista
boolean	**cambiarDatosSeccion**(int _idSecc, java.lang.String _nombre, java.lang.String _descripcion) Cambia los datos sobre una sección de la revista
void	**desconectar**() Se desconecta y se cierran las consultas con la base de datos
boolean	**eliminar**(int _id) Se elimina la revista
boolean	**eliminarPalabrasClave**(int _id) Elimina las palabras claves asociadas a una revista
boolean	**eliminarPalabrasClaveSeccion**(int _id) Elimina las palabras claves asociadas a una sección
boolean	**eliminarSeccion**(int _idSecc, boolean _descendientes) Se elimina una sección de una revista puediendo optar por borrar también todas las que dependan de ella
boolean	**eliminarSolicitud**(java.lang.String _eMail, int _idRevi)

	Elimina una solicitud para ser revisor de una de las revistas
protected void	**finalize**() Cuando se elimina el objeto hay que cerrar las consultas y la conexión a la base de datos, si aun siguen activadas
boolean	**firstPage**() Se va a la primera página de la búsqueda PRS y nos situamos sobre el primer elemento.
int	**getContador**(int _id) Devuelve el número de visitas de una determinada revista
int	**getCountArticulos**() Este metodo permite saber el número de artículos que hay albergados en el sistema
int	**getCountArticulos**(int _id) Este metodo permite saber el número de artículos que tiene una determinada revista
int	**getCountArticulos**(java.lang.String _eMail) Este método permite saber el número de artículos que tienen todas las revistas de un determinado cliente
int	**getCountArticulosFromSeccion**(int _id) Este metodo permite saber el número de artículos que tiene una sección determinada
int	**getCountArticulosQueRevisas**(java.lang.String _eMail, int _idRevi) Este metodo permite saber el número de artículos que esta revisando un determinado revisor de una revista dada
int	**getCountArticulosSinRevisor**(int _idRevi) Este metodo permite saber el número de artículos que No esta revisando un determinado revisor de una revista dada.
int	**getCountArticulosSinRevisorConInteres**(java.lang.String _eMail, int _idRevi) Este metodo permite saber el número de artículos de una revista que no estan siendo revisados y que le puede interesera al revisor.
int	**getCountRevisores**(int _id) Para saber cuantos revisores tiene una revista.
int	**getCountRevistas**() Para saber cuantas revistas hay en el sistema.
int	**getCountRevistas**(java.lang.String _eMail, int _usuario) Para saber cuantas Revistas tiene un cliente o gestiona un Revisor-
int	**getCountSecciones**(int _id) Este metodo permite saber el número de seccines que tiene una determinada revista
int	**getCountSecciones**(int _idRevi, int _idSec) Este método permite saber el número de secciones que tiene una determinada sección de una revista.
int	**getCountSeccionesDependientes**(int _id) Este metodo permite saber el número de seccines que depende de otra sección

int	**getCountSeccionesPrincipales**(int _id) Este metodo permite saber el número de seccines principales que tiene una determinada revista
java.lang.String	**getDescripcion**() Se devuelve el atributo Descripcion sobre la tupla en la que estamos dentro de una búsqueda
java.lang.String	**getEmail**() Devuelve el atributo Email de una búsqueda
int	**getEstilo**() Se devuelve el Atributo Estilo de la búsqueda
int	**getEstilo**(int _id) Se devuelve el identificador del Estilo que tiene la revista que se pasa como parámetro
java.lang.String	**getExposicion**() Devuelve el campo o atributo exposicion de una búsqueda
java.lang.String	**getFecha**() Se devuelve el atributo Fecha de una búsqueda
java.lang.String	**getFileEstilo**() Se devuelve el atributo File de la búsqueda
java.lang.String	**getFileEstilo**(int _id) Se devuelve el nombre del archivo donde está ubicado el estilo que tiene como identificador el pasado
int	**getId**() Se devuelve el ID de la revista sobre la cual estamos dentro de una búsqueda
int	**getId**(java.lang.String _nombre) Dado el nombre de la revista nos devuelve el número identificatorio de esta.
int	**getIdRevista**() Se devuelve el ID de la revista
int	**getIdSeccion**() Se devuelve el Atributo IDSeccion sobre la tupla que estamos
int	**getIdSeccion**(java.lang.String _nombre) Dado el nombre de la sección nos devuelve el número identificatorio de esta
int	**getMaxIdSecc**(int _idRevi) Nos devuelve la sección con identificado mayor de una determinada Revista, nos servirá para saber el identificador de la última sección introducida, ya que es la que tiene el identificador mayor.
boolean	**getNext**() Se pasa a la siguiente tupla de la búsqueda (si es que la hay), si no hay además cierra las consultas a la base de datos.
java.lang.String	**getNombre**() Se devuelve el atributo nombre de la tupla sobre la cual estamos situados en una búsqueda
java.lang.String	**getNombre**(int _id) Se devuelve el nombre de una revista pasando su identificador
java.lang.String	**getNombreEstilo**(int _id) Se devuelve el Nombre del estilo que se pasa como parámetro

java.lang.String	**getNombreSeccion**(int _id) Pasado el identificador de una sección devolvemos su nombre.
java.lang.String	**getPage**() Para saber sobre que páginas estamos situados en una búsqueda PRS
java.lang.String	**getPages**() Para saber cuantas páginas ha devuelto la búsqueda PRS
java.lang.String	**getPalabraClave**() Se devuelve el nombre del atributo Palabra sobre el cual estamos situados
int	**getPrsContador**() Para coseguir el número de visitas que tiene la revista sobre la cual estamos en una búsqueda PRS
java.lang.String	**getPrsDescripcion**() Para coseguir la descripcion de la revista sobre la que estamos situados de la búsqueda PRS.
java.lang.String	**getPrsEmail**() Para coseguir el Email del creador de la revista sobre la que estamos situados de la búsqueda PRS.
int	**getPrsEstilo**() Para Saber el identificador del Estilo que usa la revista sobre la que estamos situados dentro de una búsqueda PRS.
java.lang.String	**getPrsFecha**() Para coseguir la Fecha de creación de la revista sobre la cual estamos situados en la búsqueda PRS.
int	**getPrsID**() Para coseguir el ID de la tupla en la que estamos situados de la búsqueda PRS
boolean	**getPrsLogo**() Para saber si la revista sobre la que estamos situados tiene o no Logo dentro de una búsqueda PRS.
java.lang.String	**getPrsNombre**() Para coseguir el Nombre de la trevista en la que estamos situados de la búsqueda PRS
boolean	**getPrsPromocinada**() Para coseguir si esta o no promocionada la revista sobre la cual estamos situados de la búsqueda PRS.
int	**getSeccionPadre**(int _id) Devuelve el identificador de la sección padre de una sección determinada
boolean	**insertarPalabraClave**(int _id, java.lang.String _palabra) Inserta una palabra clave asociada con una revista (las palabras clave son palabras que describen de que trata la revista
boolean	**insertarPalabraClaveSeccion**(int _id, java.lang.String _palabra) Inserta una palabra clave asociada con una sección de una revista (las palabras clave son palabras que describen de que trata la sección)
boolean	**isLogo**() Se devuelve si el atributo Logo es true o false de la búsqueda
boolean	**isLogo**(int _id)

	Se devuelve si tiene o no logo la revista
boolean	**isPaginaAnterior**() Para saber si hay alguna página anterior sobre la cual estamos situados en una búsqueda PRS
boolean	**isPaginaSiguiente**() Para saber si hay alguna página posterior sobre la cual estamos situados en una búsqueda PRS
boolean	**isPromocionada**() Para saber si esta a true o false el atributo Promocionada sobre la tupla en la que estamos situados en la búsqueda
boolean	**isPrsVacio**() Para saber si hay alguna tupla devuelta por una búsqueda PRS
boolean	**isSeccionFinal**(int _id) Nos indica si la sección que le indicamos, es una sección que tiene o no secciones dependientes
boolean	**lastPage**() Se va a la última página de la búsqueda PRS y nos situamos sobre su primer elemento.
boolean	**NewRevista**(java.lang.String _eMail, java.lang.String _nombre, java.lang.String _descripcion, int _estilo, boolean _promocionada, java.sql.Date _fecha) Inserta una nueva revista en el sistema
boolean	**next**() Pasa al siguiente elemento de la búsqueda PRS sin importarnos si el elmento siguiente está en la misma página o en otra distinta.
boolean	**nextInPage**() Se va al siguiente elemento de una búsqueda PRS dentro de una página.
boolean	**nextPage**() Se va a la siguiente página de la búsqueda PRS y nos situamos sobre el primer elemento.
boolean	**nuevaSeccion**(int _idRevi, int _idSecc, java.lang.String _nombre, java.lang.String _descripcion) Inserta una nueva seccion dentro de la revista
boolean	**previousPage**() Se va a la página anterior de una búsqueda PRS y se situa sobre el primer elemento
void	**setAvanzarContador**(int _id) Introduce una visita más en el contador de la revista
void	**setBuscar**(java.lang.String _eMail) Se buscan todos las revistas que pertenecen a un cliente determinado Atributos que devuelve la tuplas de la búsqueda: ID, Nombre, Promocionada, Descripcion, eMail, Fecha, Logo, Estilo, Contador
void	**setBuscarById**(int _id) Busca una revista que se corresponda con el identificador de la revista.
void	**setBuscarEstilos**() Busca Todos los estilos que están disponibles para las revistas.

void	**setBuscarPalabrasClave**(int _id) Busca las palabras clave asociadas a una determinada Revista Atributos: Id, palabra
void	**setBuscarPalabrasClaveSeccion**(int _id) Busca las palabras clave asociadas a una determinada sección de una Revista Atributos: Id, palabra
void	**setBuscarQueRevisa**(java.lang.String _eMail) Se buscan todos las revistas que revisa un revisor Atributos que devuelve la tuplas de la búsqueda: ID, Nombre, Promocionada, Descripcion, eMail, Fecha, Logo, Estilo, Contador
void	**setBuscarQueRevisa**(java.lang.String _eMailRevi, java.lang.String _eMailCliente) Se buscan todos las revistas que revisa un revisor y que pertenecen a un determinado cliente.
void	**setConLogo**(int _id) Para marcar que una revista tiene un logo insertado
void	**setNoNext**() Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda (PRS) al realizar el siguiente Next o alguna de sus variantes.
boolean	**topPage**() Nos ponemos sobre el primer elemento de la página sobre la que estamos situados en una búsqueda PRS

Methods inherited from class java.lang.Object

clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

RevistaBean

public **RevistaBean**()

Realiza la conexión con la base de datos

Method Detail

getCountRevisores

public int **getCountRevisores**(int _id)

Para saber cuantos revisores tiene una revista.

Parameters:

_id - Es el identificador de la revista

Returns:

El número de revisores.

---

getCountRevistas

public int **getCountRevistas**()

Para saber cuantas revistas hay en el sistema.

Returns:

El número de revistas.

getMaxIdSecc

```
public int getMaxIdSecc(int _idRevi)
```

Nos devuelve la sección con identificado mayor de una determinada Revista, nos servirá para saber el identificador de la última sección introducida, ya que es la que tiene el identificador mayor.

Parameters:

`_idRevi` - Es el identificador de la Revista

Returns:

El identificador de la Sección, 0 en caso de error o no haber secciones

getCountRevistas

```
public int getCountRevistas(java.lang.String _eMail,
                            int _usuario)
```

Para saber cuantas Revistas tiene un cliente o gestiona un Revisor-

Parameters:

`_eMail` - Identificador del usuario

`_usuario` - (Class Constantes) Nos dice el usuario (Cliente o Revisor)

Returns:

El número de revistas

getCountArticulosQueRevisas

```
public int getCountArticulosQueRevisas(java.lang.String _eMail,
                                       int _idRevi)
```

Este metodo permite saber el número de artículos que esta revisando un determinado revisor de una revista dada

Parameters:

`_eMail` - Es el e-mail del revisor

`_idRevi` - Es el identificador de la revista

Returns:

El número de artículos que esta revisando el revisor de la revista.

getCountArticulosSinRevisor

```
public int getCountArticulosSinRevisor(int _idRevi)
```

Este metodo permite saber el número de artículos que No esta revisando un determinado revisor de una revista dada.

Parameters:

`_eMail` - Es el e-mail del revisor

`_idRevi` - Es el identificador de la revista

Returns:

El número de artículos que esta revisando el revisor de la revista.

getCountArticulosSinRevisorConInteres

```
public int getCountArticulosSinRevisorConInteres(java.lang.String _eMail,
                                                 int _idRevi)
```

Este metodo permite saber el número de artículos de una revista que no estan siendo revisados y que le puede interesera al revisor.

Parameters:

`_eMail` - Es el e-mail del revisor

`_idRevi` - Es el identificador de la revista

Returns:

El número de artículos de una revista sin revisor que le pueden interesar.

getCountArticulos

```
public int getCountArticulos(java.lang.String _eMail)
```

Este método permite saber el número de artículos que tienen todas las revistas de un determinado cliente

Parameters:

_eMail - Es el e-mail del cliente
Returns:
El número de artículos que tienen todas sus revistas.

---

getCountArticulos
`public int getCountArticulos(int _id)`
Este metodo permite saber el número de artículos que tiene una determinada revista
Parameters:
_id - Es el identificador de la revista
Returns:
El número de artículos que tiene la revista.

---

getCountArticulos
`public int getCountArticulos()`
Este metodo permite saber el número de artículos que hay albergados en el sistema
Returns:
El número de artículos que tiene todas las revistas albergadas en el sistema.

---

getCountArticulosFromSeccion
`public int getCountArticulosFromSeccion(int _id)`
Este metodo permite saber el número de artículos que tiene una sección determinada
Parameters:
_id - Es el identificador de la sección
Returns:
El número de artículos que tiene la SECCIÓN

---

getCountSecciones
`public int getCountSecciones(int _id)`
Este metodo permite saber el número de seccines que tiene una determinada revista
Parameters:
_id - Es el identificador de la revista
Returns:
El número de secciones que tiene la revista.

---

getCountSeccionesPrincipales
`public int getCountSeccionesPrincipales(int _id)`
Este metodo permite saber el número de seccines principales que tiene una determinada revista
Parameters:
_id - Es el identificador de la revista
Returns:
El número de secciones que tiene.

---

getCountSeccionesDependientes
`public int getCountSeccionesDependientes(int _id)`
Este metodo permite saber el número de seccines que depende de otra sección
Parameters:
_id - Es el identificador de la seccion
Returns:
El número de secciones que tiene .

---

getCountSecciones
```
public int getCountSecciones(int _idRevi,
                             int _idSec)
```
Este método permite saber el número de secciones que tiene una determinada sección de una revista.
Parameters:
_idRevi - Es el identificador de la Revista

_idSec - Es el identificador de la sección (0 -> No depende de ninguna sección si no que es una sección principal)
Returns:
El número de secciones que tiene esta sección

---

beforeFirst

```
public void beforeFirst()
```

Nos posicionamos antes del primer elemento de la búsqueda PRS

---

setConLogo

```
public void setConLogo(int _id)
```

Para marcar que una revista tiene un logo insertado
Parameters:
_id - identificador de la revista

---

insertarPalabraClave

```
public boolean insertarPalabraClave(int _id,
                                    java.lang.String _palabra)
```

Inserta una palabra clave asociada con una revista (las palabras clave son palabras que describen de que trata la revista
Parameters:
_id - Es el identificado de la revista
_palabra - Es la palabra clave que se inserta
Returns:
True -> Se ha insertado correctamente False -> Se ha producido algún error

---

insertarPalabraClaveSeccion

```
public boolean insertarPalabraClaveSeccion(int _id,
                                           java.lang.String _palabra)
```

Inserta una palabra clave asociada con una sección de una revista (las palabras clave son palabras que describen de que trata la sección)
Parameters:
_id - Es el identificado de la seccion
_palabra - Es la palabra clave que se inserta
Returns:
True -> Se ha insertado correctamente False -> Se ha producido algún error

---

nuevaSeccion

```
public boolean nuevaSeccion(int _idRevi,
                            int _idSecc,
                            java.lang.String _nombre,
                            java.lang.String _descripcion)
```

Inserta una nueva seccion dentro de la revista
Parameters:
_idRevi - Es el identificado de la revista
_idSecc - Es el identificador de la sección de la cual depende 0 -> Si es una sección principal
_nombre - Es el nombre que va a tener la sección
_descripcion - Es la descripción de la sección
Returns:
True -> Se ha insertado correctamente False -> Se ha producido algún error

---

cambiarDatosSeccion

```
public boolean cambiarDatosSeccion(int _idSecc,
                                   java.lang.String _nombre,
                                   java.lang.String _descripcion)
```

Cambia los datos sobre una sección de la revista
Parameters:

_idSecc - Es el identificador de la sección de la cual depende 0 -> Si es una sección principal
_nombre - Es el nombre que va a tener la sección
_descripcion - Es la descripción de la sección
Returns:
True -> Se ha modificado correctamente False -> Se ha producido algún error

---

NewRevista

```
public boolean NewRevista(java.lang.String _eMail,
                          java.lang.String _nombre,
                          java.lang.String _descripcion,
                          int _estilo,
                          boolean _promocionada,
                          java.sql.Date _fecha)
```

Inserta una nueva revista en el sistema
Parameters:
_eMail - Es la dirección de correo electrónico del dueño de la revista
_nombre - Es el nombre que va a tener la revista
_descripcion - Es una breve descripción de la revista
_estilo - que tiene visualmente la revista de nuestro cliente
_promocionada - Si promocionamos la revista para que otros usuarios la vean
_fecha - Fecha en la que fue creada la revista
Returns:
true -> Se ha efectuado correctamente false -> Pues no

---

cambiarDatos

```
public boolean cambiarDatos(int _id,
                            java.lang.String _nombre,
                            java.lang.String _descripcion,
                            int _estilo,
                            boolean _promocionada)
```

Cambia los datos que tenemos de una revista
Parameters:
_id - Es el identificador de la revista a la cual le vamos a cambiar los datos
_nombre - Es el nombre que va a tener la revista
_descripcion - Es una breve descripción de la revista
_estilo - que tiene visualmente la revista de nuestro cliente
_promocionada - Si promocionamos la revista para que otros usuarios la vean
Returns:
true -> Se ha efectuado correctamente false -> Pues no

---

isPrsVacio

```
public boolean isPrsVacio()
```

Para saber si hay alguna tupla devuelta por una búsqueda PRS
Returns:
True -> No esta vacio False -> Está vacío

---

setNoNext

```
public void setNoNext()
```

Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda (PRS) al realizar el siguiente Next o alguna de sus variantes. Muy util si se cambia de página y no queremos que en la primera iteración del Next realice un avance. Asi podemos tratar a todos los elementos de la misma forma en bucle while. Ya que al realizar el bucle se saltaria al hacer el next el elemento sobre el cual estamos situados (Al pasar de página nunca podemos estar antes del primer elemento,siempre nos dejará en el primero).

---

topPage

```
public boolean topPage()
```

Nos ponemos sobre el primer elemento de la página sobre la que estamos situados en una búsqueda PRS

nextInPage
`public boolean **nextInPage**()`
Se va al siguiente elemento de una búsqueda PRS dentro de una página.
Returns:
True -> Hay elemento posterior y nos hemos situados sobre el False -> No hay elemento posterior

next
`public boolean **next**()`
Pasa al siguiente elemento de la búsqueda PRS sin importarnos si el elmento siguiente está en la misma página o en otra distinta.
Returns:
boolean True -> Hay siguiente elemento y se ha podido ir a este False -> Pues no

previousPage
`public boolean **previousPage**()`
Se va a la página anterior de una búsqueda PRS y se situa sobre el primer elemento
Returns:
True -> Hay elemento anterior y nos hemos situados sobre el False -> No hay elemento anterior

nextPage
`public boolean **nextPage**()`
Se va a la siguiente página de la búsqueda PRS y nos situamos sobre el primer elemento.
Returns:
True -> Hay siguiente página y nos hemos situado en su primer elemento False -> No hay siguiente página

firstPage
`public boolean **firstPage**()`
Se va a la primera página de la búsqueda PRS y nos situamos sobre el primer elemento.
Returns:
True -> Hay primer página y nos hemos situado en su primer elemento False -> No hay ninguna página

lastPage
`public boolean **lastPage**()`
Se va a la última página de la búsqueda PRS y nos situamos sobre su primer elemento.
Returns:
True -> Nos hemos situado en la última página y en su primer elemento False -> No hay ninguna página

getPages
`public java.lang.String **getPages**()`
Para saber cuantas páginas ha devuelto la búsqueda PRS
Returns:
No devuelve el número de páginas en un String

isPaginaAnterior
`public boolean **isPaginaAnterior**()`
Para saber si hay alguna página anterior sobre la cual estamos situados en una búsqueda PRS
Returns:
True -> Hay una página anterior False -> No hay ninguna página anterior

isPaginaSiguiente
`public boolean **isPaginaSiguiente**()`
Para saber si hay alguna página posterior sobre la cual estamos situados en una búsqueda PRS
Returns:
True -> Hay una página False -> No la hay

getPage
`public java.lang.String` **`getPage`**`()`
Para saber sobre que páginas estamos situados en una búsqueda PRS
Returns:
Nos devuelve un String indicándonos el número de la página sobre la que estamos

---

getNext
`public boolean` **`getNext`**`()`
Se pasa a la siguiente tupla de la búsqueda (si es que la hay), si no hay además cierra las consultas a la base de datos.
Returns:
True -> Se ha pasado a la siguiente tupla de la búsqueda False -> Estamos al final y no hay más tuplas o no se ha efectuado la operación

---

getPrsID
`public int` **`getPrsID`**`()`
Para coseguir el ID de la tupla en la que estamos situados de la búsqueda PRS
Returns:
Nos devulve el ID 0 --> Error

---

getPrsNombre
`public java.lang.String` **`getPrsNombre`**`()`
Para coseguir el Nombre de la trevista en la que estamos situados de la búsqueda PRS
Returns:
Nos devulve el Nombre

---

getPrsPromocinada
`public boolean` **`getPrsPromocinada`**`()`
Para coseguir si esta o no promocionada la revista sobre la cual estamos situados de la búsqueda PRS.
Returns:
Nos devulve si esta o no promocionada

---

getPrsDescripcion
`public java.lang.String` **`getPrsDescripcion`**`()`
Para coseguir la descripcion de la revista sobre la que estamos situados de la búsqueda PRS.
Returns:
Nos devulve la descripcion de la revista

---

getPrsEmail
`public java.lang.String` **`getPrsEmail`**`()`
Para coseguir el Email del creador de la revista sobre la que estamos situados de la búsqueda PRS.
Returns:
Nos devulve el EMail de la revista

---

getPrsFecha
`public java.lang.String` **`getPrsFecha`**`()`
Para coseguir la Fecha de creación de la revista sobre la cual estamos situados en la búsqueda PRS.
Returns:
Nos devulve la Fecha de la revista

---

getPrsLogo
`public boolean` **`getPrsLogo`**`()`
Para saber si la revista sobre la que estamos situados tiene o no Logo dentro de una búsqueda PRS.
Returns:
Nos devulve si tiene o no logo la revista

---

getPrsEstilo

`public int` **`getPrsEstilo`**`()`
Para Saber el identificador del Estilo que usa la revista sobre la que estamos situados dentro de una búsqueda PRS.
Returns:
Nos devulve el Estilo de la revista 0 -> Error

---

getPrsContador
`public int` **`getPrsContador`**`()`
Para coseguir el número de visitas que tiene la revista sobre la cual estamos en una búsqueda PRS
Returns:
Nos devulve las visitas

---

buscarSecciones
`public void` **`buscarSecciones`**`(int _idRevi)`
Busca una serie de secciones que dependen directamente de una revista. Atributos: IDSeccion, ID, Nombre, Descripcion, Padre
Parameters:
`_idRevi` - Es la revista.

---

getSeccionPadre
`public int` **`getSeccionPadre`**`(int _id)`
Devuelve el identificador de la sección padre de una sección determinada
Parameters:
`_id` - Identificador de la sección
Returns:
El identificador de la sección padre, si no depende de ninguna sección devuelve 0

---

buscarSeccion
`public void` **`buscarSeccion`**`(int _id)`
Busca una seccion . Atributos: IDSeccion, ID, Nombre, Descripcion, Padre
Parameters:
`_id` - Identificador de la Seccion.

---

buscarSeccionesDependientes
`public void` **`buscarSeccionesDependientes`**`(int _idSecc)`
Busca una serie de secciones que dependen de otra seccion. Atributos: IDSeccion, ID, Nombre, Descripcion, Padre
Parameters:
`_idSecc` - Es el identificador de la seccion padre de las buscadas.

---

buscarTopRevistas
```
public void buscarTopRevistas(java.lang.String _palabra,
                              boolean _tituloRevi,
                              boolean _descripRevi,
                              boolean _palabrasRevi,
                              boolean _tituloSecc,
                              boolean _descripSecc,
                              boolean _palabrasSecc)
```
Se realiza una búqueda PRS sólo se realiza la búsqueda si se introduce un parámetro si no este se ignora al realizar la búsqueda. Atributos que devuelve la tuplas de la búsqueda: ID, Nombre, Promocionada, Descripcion, eMail, Fecha, Logo, Estilo, Contador
Parameters:
`_palabra` - Palabra de la búsqueda
`_tituloRevi` - Si se buscan las revistas que contengan la palabra en su título
`_descripRevi` - Si se buscan las revistas que contengan la palabra en su descripción
`_palabrasRevi` - Si se buscan las revistas que contengan la palabra en sus palabras clave

_tituloSecc - Si se buscan las revistas que contengan la palabra en el título de alguna de sus secciones
_descripSecc - Si se buscan las revistas que contengan la palabra en la descripción de alguna de sus secciones
_palabrasSecc - Si se buscan las revistas que contengan la palabra en las palabras de alguna de sus secciones

---

buscar

```
public void buscar(java.lang.String _nombre,
                   java.lang.String _nick,
                   int _ordenar,
                   int _orden)
```

Se realiza una búqueda PRS sólo se realiza la búsqueda si se introduce un parámetro Si no este se ignora al realizar la búsqueda. Atributos que devuelve la tuplas de la búsqueda: ID, Nombre, Promocionada, Descripcion, eMail, Fecha, Logo, Estilo, Contador

Parameters:

_nombre - Se realiza la búsqueda de todas las tuplas que tengan esta palabra en su nombre
_nick - Se buscan las revistas cuyo creador tenga el nick aquí introducido
_ordenar - 1 -> Orden ascendente 2 -> Orden descendente
_orden - 1 -> Por Titulo 3 -> Fecha creación

---

desconectar

```
public void desconectar()
```

Se desconecta y se cierran las consultas con la base de datos

---

setBuscarPalabrasClave

```
public void setBuscarPalabrasClave(int _id)
```

Busca las palabras clave asociadas a una determinada Revista Atributos: Id, palabra

Parameters:

_id - identificador de la palabra clave de la revista

---

setBuscarEstilos

```
public void setBuscarEstilos()
```

Busca Todos los estilos que están disponibles para las revistas. Attributos de cada tupla devuelta: ID, Nombre, File, Descripcion

---

setBuscarPalabrasClaveSeccion

```
public void setBuscarPalabrasClaveSeccion(int _id)
```

Busca las palabras clave asociadas a una determinada sección de una Revista Atributos: Id, palabra

---

setBuscarById

```
public void setBuscarById(int _id)
```

Busca una revista que se corresponda con el identificador de la revista. Atributos que devuelve la tuplas de la búsqueda: ID, Nombre, Promocionada, Descripcion, eMail, Fecha, Logo, Estilo, Contador

Parameters:

_id - Es el identificador de la revista

Returns:

True -> Operación realizada con éxito False -> Pues no

---

setBuscar

```
public void setBuscar(java.lang.String _eMail)
```

Se buscan todos las revistas que pertenecen a un cliente determinado Atributos que devuelve la tuplas de la búsqueda: ID, Nombre, Promocionada, Descripcion, eMail, Fecha, Logo, Estilo, Contador

Parameters:

_eMail - Es el identificador del cliente.

---

setBuscarQueRevisa
```
public void setBuscarQueRevisa(java.lang.String _eMail)
```
Se buscan todos las revistas que revisa un revisor Atributos que devuelve la tuplas de la búsqueda: ID, Nombre, Promocionada, Descripcion, eMail, Fecha, Logo, Estilo, Contador

Parameters:

`_eMail` - Es el identificador del revisor.

---

setBuscarQueRevisa
```
public void setBuscarQueRevisa(java.lang.String _eMailRevi,
                               java.lang.String _eMailCliente)
```
Se buscan todos las revistas que revisa un revisor y que pertenecen a un determinado cliente. Atributos que devuelve la tuplas de la búsqueda: ID, Nombre, Promocionada, Descripcion, eMail, Fecha, Logo, Estilo, Contador

Parameters:

`_eMailRevi` - Es el identificador del revisor.

`_eMailCliente` -

---

getId
```
public int getId(java.lang.String _nombre)
```
Dado el nombre de la revista nos devuelve el número identificatorio de esta.

Parameters:

`_nombre` - Es el nombre de la revista buscada

Returns:

El número identificatorio de la revista 0 --> Error

---

getIdSeccion
```
public int getIdSeccion(java.lang.String _nombre)
```
Dado el nombre de la sección nos devuelve el número identificatorio de esta

Parameters:

`_nombre` - Es el nombre de la revista buscada

Returns:

El número identificatorio de la revista 0 --> Error

---

getId
```
public int getId()
```
Se devuelve el ID de la revista sobre la cual estamos dentro de una búsqueda

Returns:

Un Entero 0 -> Error

---

getIdSeccion
```
public int getIdSeccion()
```
Se devuelve el Atributo IDSeccion sobre la tupla que estamos

Returns:

Un Entero 0 -> Error

---

getIdRevista
```
public int getIdRevista()
```
Se devuelve el ID de la revista

Returns:

Un Entero 0 -> Error

---

getPalabraClave
```
public java.lang.String getPalabraClave()
```
Se devuelve el nombre del atributo Palabra sobre el cual estamos situados

Returns:

Una cadena que es su nombre

---

getNombre
`public java.lang.String` **`getNombre`**`(int _id)`
Se devuelve el nombre de una revista pasando su identificador
Parameters:
`_id` - Es el identificador de la revista
Returns:
Una cadena que es el nombre de la revista o del estilo

---

getNombre
`public java.lang.String` **`getNombre`**`()`
Se devuelve el atributo nombre de la tupla sobre la cual estamos situados en una búsqueda
Returns:
Una cadena que es su nombre

---

getNombreSeccion
`public java.lang.String` **`getNombreSeccion`**`(int _id)`
Pasado el identificador de una sección devolvemos su nombre.
Parameters:
`_id` - Es el identificador de la seccion
Returns:
El nombre de la sección

---

isPromocionada
`public boolean` **`isPromocionada`**`()`
Para saber si esta a true o false el atributo Promocionada sobre la tupla en la que estamos situados en la búsqueda
Returns:
Si esta o no promocionada la revista

---

getDescripcion
`public java.lang.String` **`getDescripcion`**`()`
Se devuelve el atributo Descripcion sobre la tupla en la que estamos dentro de una búsqueda
Returns:
La cadena que representa la descripción

---

getExposicion
`public java.lang.String` **`getExposicion`**`()`
Devuelve el campo o atributo exposicion de una búsqueda
Returns:
Su dirección de correo electrónico

---

getEmail
`public java.lang.String` **`getEmail`**`()`
Devuelve el atributo Email de una búsqueda
Returns:
La dirección de correo electrónico

---

getFecha
`public java.lang.String` **`getFecha`**`()`
Se devuelve el atributo Fecha de una búsqueda
Returns:
Una cadena que es la fecha

---

isLogo
`public boolean` **`isLogo`**`()`
Se devuelve si el atributo Logo es true o false de la búsqueda
Returns:

Si tiene o no un logo la revista

---

isLogo
`public boolean isLogo(int _id)`
Se devuelve si tiene o no logo la revista
Parameters:
`_id` - Es el identificador de la revista
Returns:
Si tiene o no un logo la revista

---

getEstilo
`public int getEstilo()`
Se devuelve el Atributo Estilo de la búsqueda
Returns:
El identificador del estilo

---

getEstilo
`public int getEstilo(int _id)`
Se devuelve el identificador del Estilo que tiene la revista que se pasa como parámetro
Parameters:
`_id` - Identificador de la revista
Returns:
El identificador del estilo que tiene la revista

---

getFileEstilo
`public java.lang.String getFileEstilo()`
Se devuelve el atributo File de la búsqueda
Returns:
La cadena del archivo donde está el estilo

---

getNombreEstilo
`public java.lang.String getNombreEstilo(int _id)`
Se devuelve el Nombre del estilo que se pasa como parámetro
Parameters:
`_id` - Es el identificador del Estilo
Returns:
El nombre del estilo

---

getFileEstilo
`public java.lang.String getFileEstilo(int _id)`
Se devuelve el nombre del archivo donde está ubicado el estilo que tiene como identificador el pasado
Parameters:
`_id` - Es el identificador del Estilo
Returns:
Una cadena que es el nombre de archivo donde está contenido el estilo

---

getContador
`public int getContador(int _id)`
Devuelve el número de visitas de una determinada revista
Parameters:
`_id` - Es el identificador de la revista
Returns:
El número de visitas

---

eliminar
`public boolean eliminar(int _id)`

Se elimina la revista
Parameters:
`_id` - Es el identificador de la revista a eliminarla
Returns:
True -> Se ha eliminado correctamente False -> No se ha podido eliminar

---

eliminarSeccion

```
public boolean eliminarSeccion(int _idSecc,
                               boolean _descendientes)
```

Se elimina una sección de una revista puediendo optar por borrar también todas las que dependan de ella
Parameters:
`_idSecc` - Identificador de la sección que se quiere eliminar
`_descendientes` - True-> Se eliminan también las secciones descendientes False -> Solo la indicada
Returns:
True -> Se han eliminado correctamente False -> No se han elminado correctamente

---

eliminarPalabrasClave

```
public boolean eliminarPalabrasClave(int _id)
```

Elimina las palabras claves asociadas a una revista
Parameters:
`_id` - Es el identificador de la revista de la cual se le eliminan sus palabras clave
Returns:
True -> Se ha eliminado correctamente False -> No se ha podido eliminar

---

eliminarPalabrasClaveSeccion

```
public boolean eliminarPalabrasClaveSeccion(int _id)
```

Elimina las palabras claves asociadas a una sección
Parameters:
`_id` - Es el identificador de la revista de la cual se le eliminan sus palabras clave
Returns:
True -> Se ha eliminado correctamente False -> No se ha podido eliminar

---

Actualizar

```
public void Actualizar()
```

Actualiza la búsqueda PRS, si se produce algún cambio y se quieren actualizar los cambios producidos

---

setAvanzarContador

```
public void setAvanzarContador(int _id)
```

Introduce una visita más en el contador de la revista
Parameters:
`_id` - Identificador de la revista

---

isSeccionFinal

```
public boolean isSeccionFinal(int _id)
```

Nos indica si la sección que le indicamos, es una sección que tiene o no secciones dependientes
Parameters:
`_id` - Es el identificador de la seccion
Returns:
nos devuelve True -> Si no tiene secciones dependientes False -> Si tiene secciones dependientes

---

buscarSolicitudes

```
public void buscarSolicitudes(java.lang.String _eMail)
```

Busca las solicitudes que hacen los usuarios para ser revisores de alguna de las revistas de un determinado usuario cliente Atributos de las tuplas devueltas en la búsqueda: IDRevi, eMail, Exposicion, Fecha
Parameters:
`_eMail` - Es el eMail del usuario dueño de la revista

buscarSolicitud

```
public void buscarSolicitud(java.lang.String _eMail,
                            int _idRevi)
```

Busca una solicitud para ser revisor y nos situa sobre el único resultado y nos situamos sobre ella Atributos de las tuplas devueltas en la búsqueda: IDRevi, eMail, Exposicion, Fecha

Parameters:

`_eMail` - Es el eMail del usuario que ha hecho la solicitud

`_idRevi` - Es el identificador de la revista en la cual ha hecho la solicitud

eliminarSolicitud

```
public boolean eliminarSolicitud(java.lang.String _eMail,
                                 int _idRevi)
```

Elimina una solicitud para ser revisor de una de las revistas

Parameters:

`_eMail` - Es el eMail del usuario que ha hecho la solicitud

`_idRevi` - Es el identificador de la revista en la cual ha hecho la solicitud

finalize

```
protected void finalize()
                throws java.lang.Throwable
```

Cuando se elimina el objeto hay que cerrar las consultas y la conexión a la base de datos, si aun siguen activadas

Overrides:

`finalize` in class `java.lang.Object`

tablas

Class SectoresBean

```
java.lang.Object
  |
  +--tablas.SectoresBean
```

public class **SectoresBean**
extends java.lang.Object
Clase que sirve para acceder a la base de datos y que gestiona la tabla Sectores.

Constructor Summary
SectoresBean() Realiza la conexión con la base de datos

Method Summary	
void	**cerrar**() Se Cierra cualquier consulta que halla quedado abierta con la base de datos.
void	**desconectar**() Se Cierra cualquier consulta abierta se cierra la conexión a la base de datos
int	**getIdSector**() Se devuelve el Identificador de la tupla sobre la cual estamos situados.
int	**getIdSector**(java.lang.String _sector) Se obtiene el ID del sector asociado con el que le pasamos, no afecta a las búsquedas
boolean	**getNext**()

	Se pasa a la siguiente tupla de la búsqueda (si es que la hay), si se llega al final se cierra la consulta con la base de datos.
java.lang.String	**getNombreSector**() Se Devuelve el nombre del sector sobre el cual estamos situados en la tabla
java.lang.String	**getNombreSector**(int _id) Se obtiene el nombre del sector asociado con el identificador (ID) que le pasamos, no afecta a las búsquedas
boolean	**Insertar**(java.lang.String _sector) Se inserta una tupla en la tabla
void	**setBuscarTodos**() Se realiza una búsqueda en la cual estan todas las tuplas de Sectores.
boolean	**setEliminar**(int _id) Se elimina la tupla de la tabla que este asociada con el ID que se le pase

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

SectoresBean
public **SectoresBean**()
Realiza la conexión con la base de datos

Method Detail

Insertar
public boolean **Insertar**(java.lang.String _sector)
Se inserta una tupla en la tabla
Parameters:
_sector - Es el nombre del sector que se va introducir en la tabla
Returns:
True -> Se ha insertado en la tabla False -> No se ha insertado en la tabla

---

getNext
public boolean **getNext**()
Se pasa a la siguiente tupla de la búsqueda (si es que la hay), si se llega al final se cierra la consulta con la base de datos.
Returns:
True -> Se ha pasado a la siguiente tupla de la búsqueda False -> Estamos al final y no hay más tuplas o no se ha efectuado la operación

---

getIdSector
public int **getIdSector**()
Se devuelve el Identificador de la tupla sobre la cual estamos situados.
Returns:
Devuelve el número el ID de la tupla sobre la cual estamos situados.

---

getIdSector
public int **getIdSector**(java.lang.String _sector)
Se obtiene el ID del sector asociado con el que le pasamos, no afecta a las búsquedas
Parameters:
_sector - Se pasa el nombre del sector del cual queremos obtener su ID
Returns:

Devuelve el ID asociado, 0 si no esta contenido en la tabla

---

getNombreSector
`public java.lang.String` **`getNombreSector`**`(int _id)`
Se obtiene el nombre del sector asociado con el identificador (ID) que le pasamos, no afecta a las búsquedas
Parameters:
`_id` - Identificador (ID) del sector del cual queremos obtener su nombre
Returns:
Devuelve el nombre del sector.

---

getNombreSector
`public java.lang.String` **`getNombreSector`**`()`
Se Devuelve el nombre del sector sobre el cual estamos situados en la tabla
Returns:
Nos devuelve su nombre, si hay algún error y no se puediera obtener devolvemos la cadena vacía.

---

setEliminar
`public boolean` **`setEliminar`**`(int _id)`
Se elimina la tupla de la tabla que este asociada con el ID que se le pase
Parameters:
`_id` - Es el ID de la tupla que se quiere borrar de la tabla
Returns:
True -> Se ha efectuado el borrado False -> No se ha efectuado el borrado

---

setBuscarTodos
`public void` **`setBuscarTodos`**`()`
Se realiza una búsqueda en la cual estan todas las tuplas de Sectores. La consulta queda abierta y deberá de ser cerrada con posterioridad.
Returns:
True -> Se ha realizado la búsqueda con éxito False -> No se ha podido llevar a cabo la búsqueda

---

cerrar
`public void` **`cerrar`**`()`
Se Cierra cualquier consulta que halla quedado abierta con la base de datos.

---

desconectar
`public void` **`desconectar`**`()`
Se Cierra cualquier consulta abierta se cierra la conexión a la base de datos

tablas

Class sqlAux

```
java.lang.Object
  |
  +--tablas.sqlAux
```

---

public class **sqlAux**
extends java.lang.Object
Esta clase es una ayuda al acceso de base de datos, contiene métodos comunes y típicos que gestionan o acceden a la base de datos.

Constructor Summary	
sqlAux()	

Method Summary	
static boolean	**eliminar**(java.lang.String _sql, java.sql.Connection con) Eliminar la información de la base de datos, para lo cual se introduce la sentencia sql, pero no hay que poner en la sentencia el encabezado 'DELETE FROM'
Static int	**getCount**(java.lang.String _sql, java.sql.Connection con) Cuenta las tuplas de una determinada consulta
Static boolean	**insertar**(java.lang.String _sql, java.sql.Connection con) Inserta información a la base de datos, para lo cual se introduce la sentencia sql, pero no hay que poner en la sentencia el encabezado 'INSERT INTO'
static boolean	**modificar**(java.lang.String _sql, java.sql.Connection con) Modifica la información de la base de datos, para lo cual se introduce la sentencia sql, pero no hay que poner en la sentencia el encabezado 'UPDATE INTO'

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

sqlAux

```
public sqlAux()
```

Method Detail

getCount

```
public static int getCount(java.lang.String _sql,
                           java.sql.Connection con)
```

Cuenta las tuplas de una determinada consulta

Parameters:

_sql - La sentencia sql de la búsqueda de la cual queremos contar sus tuplas

Returns:

El número de tuplas

---

insertar

```
public static boolean insertar(java.lang.String _sql,
                               java.sql.Connection con)
```

Inserta información a la base de datos, para lo cual se introduce la sentencia sql, pero no hay que poner en la sentencia el encabezado 'INSERT INTO'

Parameters:

_sql - La sentencia sql de la búsqueda de la cual queremos contar sus tuplas

Returns:

El número de tuplas

---

modificar

```
public static boolean modificar(java.lang.String _sql,
                                java.sql.Connection con)
```

Modifica la información de la base de datos, para lo cual se introduce la sentencia sql, pero no hay que poner en la sentencia el encabezado 'UPDATE INTO'

Parameters:

_sql - La sentencia sql de la búsqueda de la cual queremos contar sus tuplas

Returns:

El número de tuplas

---

eliminar

```
public static boolean eliminar(java.lang.String _sql,
```

java.sql.Connection con)

Eliminar la información de la base de datos, para lo cual se introduce la sentencia sql, pero no hay que poner en la sentencia el encabezado 'DELETE FROM'

Parameters:

_sql - La sentencia sql de la búsqueda de la cual queremos contar sus tuplas

Returns:

El número de tuplas

tablas

Class UsuarioBean

```
java.lang.Object
  |
  +--tablas.UsuarioBean
```

---

public class **UsuarioBean**
extends java.lang.Object

Clase que sirve para gestionar a los usuarios que acceden al sistema. Hay búsquedas PRS que almacena el resultado de la búsqueda en páginas.

Constructor Summary
UsuarioBean()   Realiza la conexión con la base de datos

Method Summary	
boolean	**absolutePage**(int _page) Nos lleva a la página que le indiquemos en una búsqueda PRS
void	**Actualizar**() Actualiza la búsqueda PRS, si se produce algún cambio y se quieren actualizar los cambios producidos
void	**beforeFirst**() Nos posicionamos antes del primer elemento de la búsqueda PRS
void	**buscar**(boolean _isRegistrados, boolean _isAdministradores, boolean _isClientes, boolean _isCriticos) Busca a todos los usuarios que este dentro de alguno de estos grupos, se trata de una búsqueda PRS Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso, Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario
void	**buscarAdministradores**(java.lang.String _eMail, java.lang.String _login, java.lang.String _nombre, int _ordenar, int _orden) Se realiza una búqueda de administradores, se trata de una búsqueda PRS, se realiza la búsqueda si se introduce un parámetro, si no este se ignora al realizar la búsqueda.
void	**buscarClientes**(java.lang.String _eMail, java.lang.String _login, java.lang.String _nombre, java.lang.String _revista, int _ordenar, int _orden) Se realiza una búqueda de clientes, se trata de una búsqueda PRS, se realiza la búsqueda si se introduce un parámetro Si no este se ignora al realizar la búsqueda.
void	**buscarRevisores**(int _id)

	Se realiza una búqueda de los revisores atendiendo a la revista que revisan Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso, Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario
void	**buscarRevisores**(java.lang.String _eMail, java.lang.String _login, java.lang.String _revista, int _ordenar, int _orden) Se realiza una búqueda de revisores, se trata de una búsqueda PRS, sólo se realiza la búsqueda si se introduce un parámetro, si no este se ignora al realizar la búsqueda.
void	**buscarRevisores**(java.lang.String _eMail, java.lang.String _login, java.lang.String _nombre, java.lang.String _revista, int _ordenar, int _orden) Se realiza una búqueda de revisores, se trata de una búsqueda PRS, sólo se realiza la búsqueda si se introduce un parámetro Si no este se ignora al realizar la búsqueda.
void	**desconectar**() Desconecta de la base de datos
protected void	**finalize**() Cuando se elimina un objeto de esta clase habrá que cerrar las consultas y conexiones que aun pudieran estar abiertas
boolean	**first**() Nos posicionamos en el primer elemento de una búsqueda PRS
boolean	**firstPage**() Se va a la primera página de la búsqueda PRS y nos situamos sobre el primer elemento.
java.lang.String	**getAnio**() Para conseguir el Año almacenado en la Fecha de la tupla encontrada en la búsqueda (Es el año de nacimiento)
java.lang.String	**getApellidos**() Para conseguir los Apellidos de la tupla encontrada en la búsqueda
java.lang.String	**getCalle**() Para conseguir la Calle de la tupla encontrada en la búsqueda
java.lang.String	**getCodigo**() Para conseguir el CódPostal de la tupla encontrada en la búsqueda
java.lang.String	**getContrasenia**() Para conseguir la Contraseña de la tupla encontrada en la búsqueda
int	**getCountAdministradores**() Para saber cuantos Administradores hay en el sistema.
int	**getCountClientes**() Para saber cuantos clientes hay en el sistema.
int	**getCountRevisa**(java.lang.String _eMail) Para saber cuantas revistas revisa un revisor.
int	**getCountRevisa**(java.lang.String _eMailRevisor, java.lang.String _eMailCliente) Para saber cuantas revistas revisa un revisor de un cliente determinado.
int	**getCountRevisores**() Para saber cuantos Revisores hay en el sistema.

int	**getCountRevisores**(java.lang.String _eMail) Para saber cuantos Revisores tiene en todas sus revistas un cliente.
java.lang.String	**getDia**() Para conseguir el día almacenado en la Fecha de la tupla encontrada en la búsqueda (Es el día de nacimiento)
int	**getEducacion**() Para conseguir el Estudio de la tupla encontrada en la búsqueda
java.lang.String	**getEmail**() Para conseguir el eMail de la tupla encontrada en la búsqueda
java.lang.String	**getEmail**(int _id) Para conseguir el eMail del usario pasándole su identificador.
boolean	**getExisteEMail**(java.lang.String _eMail) Sirve para saber algún usuario insertado en el sistema que tenga el Email pasado como parámetro
boolean	**getExisteLogin**(java.lang.String _login) Sirve para saber si hay algun usuario insertado con el login pasado
java.lang.String	**getFecha**() Para conseguir la fecha completa almacenada en la tupla encontrada en la búsqueda (fecha nacimiento)
java.lang.String	**getFechaAdministrador**(java.lang.String _eMail) Se devuelve la fecha en la que se insertó un Administrador en el sistema
java.lang.String	**getFechaCliente**(java.lang.String _eMail) Se devuelve la fecha asociada con la inserción de un cliente
java.lang.String	**getFirma**() Para conseguier la firma con la que envia las notas y eMail este usuario, campo Firma de la búsqueda
int	**getId**(java.lang.String _eMail) Para conseguir el identificador del usuario mediante su eMail
java.lang.String	**getLetra**() Para conseguir la Letra de la tupla encontrada en la búsqueda
java.lang.String	**getLogin**() Para conseguir el Login de la tupla encontrada en la búsqueda
java.lang.String	**getMes**() Para conseguir el mes almacenado en la Fecha de la tupla encontrada en la búsqueda (Es el mes de nacimiento)
boolean	**getNext**() Pasa al siguiente elemento de la búsqueda (Para búsquedas normales y no PRS)
java.lang.String	**getNick**(java.lang.String _eMail) Nos dice el nick del cliente que ha creado la revista
java.lang.String	**getNombre**() Para conseguir el Nombre de la tupla encontrada en la búsqueda
java.lang.String	**getNoticias**() Nos indica si el usuario de la búsqueda recibe las noticias también por correo electrónico
java.lang.String	**getPage**() Para saber la página sobre la que estamos situados en una búsqueda PRS

java.lang.String	**getPages**() Para saber cuantas páginas ha devuelto la búsqueda PRS
int	**getPais**() Para conseguir el Pais de la tupla encontrada en la búsqueda.
java.lang.String	**getPalabraClave**() Para conseguir la palabra clave asociada con un usuario, campo Palabra de la búsqueda
java.lang.String	**getPiso**() Para conseguir el Piso de la tupla encontrada en la búsqueda
java.lang.String	**getPoblacion**() Para conseguir la Poblacion de la tupla encontrada en la búsqueda
java.lang.String	**getPortal**() Para conseguir el Portal de la tupla encontrada en la búsqueda
int	**getProfesion**() Para conseguir la Ocupación de la tupla encontrada en la búsqueda
int	**getProvincia**() Para conseguir la Provincia de la tupla encontrada en la búsqueda
java.lang.String	**getPrsEMail**() Para coseguir el E-Mail de la tupla en la que estamos situados de la búsqueda PRS
java.lang.String	**getPrsLogin**() Para coseguir el Login de la tupla en la que estamos situados de la búsqueda PRS.
java.lang.String	**getPrsNombre**() Para coseguir el Nombre de la tupla en la que estamos situados de la búsqueda PRS
java.lang.String	**getRecibirEmail**() Para conseguir la Ocupación de la tupla encontrada en la búsqueda
java.lang.String	**getRecibirNotas**() Para saber si el usuario debe de recibir las notas también por E-Mail, tupla de la búsqueda y campo RecibirNotas
int	**getSector**() Para conseguir el Sector de la tupla encontrada en la búsqueda
java.lang.String	**getSexo**() Para conseguir el Sexo de la tupla encontrada en la búsqueda
java.lang.String	**getTelefono**() Para conseguir el TeléfonoContacto de la tupla encontrada en la búsqueda
boolean	**Insertar**(java.lang.String _login, java.lang.String _eMail, java.lang.String _contrasena, java.lang.String _nombre, java.lang.String _apellidos, int _pais, int _provincia, java.lang.String _poblacion, java.lang.String _calle, int _portal, int _piso, java.lang.String _letra, java.lang.String _codigo, java.lang.String _telefono, java.sql.Date _fecha, int _sector, int _profesion, int _educacion, java.lang.String _sexo) Se inserta un nuevo Usuario en el sistema.
boolean	**InsertarAdministrador**(java.lang.String _eMail) Se inserta un nuevo administrador en el sistema.

boolean	**insertarCliente**(java.lang.String _eMail, java.sql.Date _fecha) Se inserta un nuevo cliente en el sistema.
boolean	**insertarPalabraClave**(java.lang.String _eMail, java.lang.String _palabra) Inserta una palabra clave asociada con una revisor (las palabras clave son palabras que describen los temas en los que es un experto este revisor)
boolean	**InsertarRevisor**(java.lang.String _eMail, int _idRevi, java.sql.Date _fecha) Se inserta un nuevo Revisor asociado a la revista que va a revisar.
boolean	**isAdministrador**(java.lang.String _eMail) Se comprueba si el Email pasado se corresponde con algun Administrador
boolean	**isCliente**(java.lang.String _eMail) Se comprueba si el Email pasado se corresponde con algun Cliente
boolean	**isNoticias**() Nos indica si el usuario de la búsqueda recibe las noticias también por correo electrónico, campo RecibirNoticias de la búsqueda
boolean	**isPaginaAnterior**() Para saber si hay alguna página anterior sobre la cual estamos situados en una búsqueda PRS
boolean	**isPaginaSiguiente**() Para saber si hay alguna página posterior sobre la cual estamos situados en una búsqueda PRS
boolean	**isPrsRecibirEmails**() Para coseguir el campo RecibirEmails de la tupla en la que estamos situados de la búsqueda PRS
boolean	**isRecibirEmail**() Para conseguir el campo RecibirEmails de la tupla encontrada en la búsqueda
boolean	**isRecibirEmail**(java.lang.String _eMail) Para saber si el usuario tiene activada la recepcion de eMails, tupla enocntrada en la búsqueda campo RecibirEmails
boolean	**isRecibirNotas**() Para saber si el usuario debe de recibir las notas también por E-Mail, tupla de la búsqueda y campo RecibirNotas
boolean	**isRevisor**(int _idUsuario, int _idRevi) Se comprueba si el usuario introducido es Revisor de la revista introducida.
boolean	**isRevisor**(java.lang.String _eMail) Se comprueba si el email pasado se corresponde con algun revisor
boolean	**lastPage**() Se va a la última página de la búsqueda PRS y nos situamos sobre su primer elemento.
boolean	**next**() Pasa al siguiente elemento de la búsqueda PRS
boolean	**nextInPage**() Se va al siguiente elemento dentro de una página en una búsqueda PRS
boolean	**nextPage**() Se va a la siguiente página de la búsqueda PRS y nos situamos sobre el primer

	elemento.
boolean	**nextPalabraClave**() Pasa a la siguiente palabra clave asociada con un usuario previamente habrá que realizar la búsqueda return true -> Hay siguiente elemento y se situa sobre el False -> No hay siguiente elemento
boolean	**previousPage**() Se va a la página anterior de una búsqueda PRS y nos situamos sobre el primer elemento.
boolean	**setActualizar**(java.lang.String _eMail, java.lang.String _firma, boolean _recibirEmail, boolean _noticias, boolean _recibirNotas) Para actualizar los valores contenido sobre un usuario
boolean	**setActualizar**(java.lang.String _eMail, java.lang.String _nombre, java.lang.String _apellidos, int _pais, int _provincia, java.lang.String _poblacion, java.lang.String _calle, int _portal, int _piso, java.lang.String _letra, java.lang.String _codigo, java.lang.String _telefono, java.sql.Date _fecha, int _sector, int _profesion, int _educacion, java.lang.String _sexo) Para actualizar los valores contenido sobre un usuario
boolean	**setBuscar**(java.lang.String _eMail) Sirve para buscar un usuario Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso, Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario
void	**setBuscarPalabrasClave**(java.lang.String _eMail) Busca las palabras clave asociadas a una determinado Revisor Atributos de las tuplas devueltas por la búsqueda: eMail, Palabra
void	**setChangePassword**(java.lang.String _eMail, java.lang.String _nuevaContra) Para actualizar la contraseña de un usuario
boolean	**setEliminar**(java.lang.String _eMail) Para Eliminar un usuario del sistema
boolean	**setEliminarAdministrador**(java.lang.String _eMail) Se elimina al usuario como administrador del sistema
boolean	**setEliminarCliente**(java.lang.String _eMail) Se elimina al usuario como cliente, y por tanto se eliminan sus revistas, y artículos.
boolean	**setEliminarPalabrasClave**(java.lang.String _eMail) Elimina las palabras clave asociadas con un usuario
boolean	**setEliminarRevisor**(java.lang.String _eMail, int _idRevi) Se elimina a un usuario como revisor de una determinada revista
boolean	**setListar**() Para hacer una búsqueda en la que aparecen todos los usuario Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso, Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario

void	**setNoNext**() Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda al realizar el siguiente Next o alguna de sus variantes.
boolean	**solicitarSerRevisor**(java.lang.String _eMail, java.lang.String _exposicion, int _idRevista) Se inserta una solicitud para ser revisor de alguna de las revistas
boolean	**topPage**() Nos ponemos sobre el primer elemento de la página sobre la que estamos situados en una búsqueda PRS

Methods inherited from class java.lang.Object
clone, equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

UsuarioBean

public **UsuarioBean**()

Realiza la conexión con la base de datos

Method Detail

setEliminar

public boolean **setEliminar**(java.lang.String _eMail)

Para Eliminar un usuario del sistema

Parameters:

_eMail - Es el cooreo del usuario que se va a eliminar

---

setEliminarRevisor

public boolean **setEliminarRevisor**(java.lang.String _eMail, int _idRevi)

Se elimina a un usuario como revisor de una determinada revista

Parameters:

_eMail - Es la dirección de correo electrónico del usuario

_idRevi - Es el identificador de la revista

Returns:

True -> Se ha eliminado False -> No se ha eliminado

---

setEliminarAdministrador

public boolean **setEliminarAdministrador**(java.lang.String _eMail)

Se elimina al usuario como administrador del sistema

Parameters:

_eMail - Dirección E-Mail del usuario

Returns:

True -> Se ha eliminado False -> No se ha eliminado

---

setEliminarCliente

public boolean **setEliminarCliente**(java.lang.String _eMail)

Se elimina al usuario como cliente, y por tanto se eliminan sus revistas, y artículos.

Parameters:

_eMail - Es el correo del usuario

Returns:

True -> Se ha eliminado False -> No se ha eliminado

---

isRevisor

`public boolean` **`isRevisor`**`(java.lang.String _eMail)`
Se comprueba si el email pasado se corresponde con algun revisor
Parameters:
`_eMail` - Correo electrónico del cual queremos saber si esta contenido en la tabla
Returns:
True -> El eMail esta contenido en la tabla False -> El email no esta en la tabla

---

isRevisor

```
public boolean isRevisor(int _idUsuario,
                         int _idRevi)
```

Se comprueba si el usuario introducido es Revisor de la revista introducida.
Parameters:
`_idUsuario` - Es el identificador del revisor
`_idRevi` - Es el identificador de la revista
Returns:
True -> Si es revisor de susodicha revista False -> Em caso contrario

---

isAdministrador

`public boolean` **`isAdministrador`**`(java.lang.String _eMail)`
Se comprueba si el Email pasado se corresponde con algun Administrador
Parameters:
`_eMail` - Correo electrónico del cual queremos saber si esta contenido en Administra
Returns:
True -> El eMail esta contenido en la tabla False -> El email no esta en la tabla

---

isCliente

`public boolean` **`isCliente`**`(java.lang.String _eMail)`
Se comprueba si el Email pasado se corresponde con algun Cliente
Parameters:
`_eMail` - Correo electrónico del cual queremos saber si esta contenido en Cliente
Returns:
True -> El eMail esta contenido en la tabla False -> El email no esta en la tabla

---

getExisteLogin

`public boolean` **`getExisteLogin`**`(java.lang.String _login)`
Sirve para saber si hay algun usuario insertado con el login pasado
Parameters:
`_login` - Es el login
Returns:
true -> Esta en el Sistema False -> No esta en el sistema

---

getExisteEMail

`public boolean` **`getExisteEMail`**`(java.lang.String _eMail)`
Sirve para saber algún usuario insertado en el sistema que tenga el Email pasado como parámetro
Parameters:
`_eMail` - Es el eMail sobre cual se quiere mirar si hay alguna tupla ya insertada
Returns:
true -> Esta introducido el usuario False -> No esta en la tabla

---

getCountRevisores

`public int` **`getCountRevisores`**`()`
Para saber cuantos Revisores hay en el sistema.
Returns:
El número de Revisores.

---

getCountRevisores

```
public int getCountRevisores(java.lang.String _eMail)
```
Para saber cuantos Revisores tiene en todas sus revistas un cliente.
Parameters:
`_eMail` - Es el email del cliente
Returns:
El número de revisores.

---

getCountRevisa
```
public int getCountRevisa(java.lang.String _eMail)
```
Para saber cuantas revistas revisa un revisor.
Parameters:
`_eMail` - Es el email del revisor
Returns:
El número de revistas que revisa.

---

getCountRevisa
```
public int getCountRevisa(java.lang.String _eMailRevisor,
                          java.lang.String _eMailCliente)
```
Para saber cuantas revistas revisa un revisor de un cliente determinado.
Parameters:
`_eMailRevisor` - Es el email del revisor
`_eMailCliente` - Es el email del Cliente
Returns:
El número de revistas de un cliente que revisa este revisor..

---

getCountClientes
```
public int getCountClientes()
```
Para saber cuantos clientes hay en el sistema.
Returns:
El número de Clientes.

---

getCountAdministradores
```
public int getCountAdministradores()
```
Para saber cuantos Administradores hay en el sistema.
Returns:
El número de Administradores.

---

insertarCliente
```
public boolean insertarCliente(java.lang.String _eMail,
                               java.sql.Date _fecha)
```
Se inserta un nuevo cliente en el sistema.
Parameters:
`_eMail` - Correo electrónico del usuario que se inserta en Cliente
`_fecha` - La fecha asociada con la insercción del cliente
Returns:
True -> El eMail se ha insertado en la tabla False -> El email no se ha insertado en la tabla

---

InsertarRevisor
```
public boolean InsertarRevisor(java.lang.String _eMail,
                               int _idRevi,
                               java.sql.Date _fecha)
```
Se inserta un nuevo Revisor asociado a la revista que va a revisar.
Parameters:
`_eMail` - Correo electrónico del usuario que se inserta como revisor en una determinada revista
`_idRevi` - Identificador de la revista que va a revisar el usuario
`_fecha` - Es la fecha que se asocia al revisor (Para saber desde cuando es revisor)
Returns:

True -> El eMail se ha insertado en la tabla False -> El email no se ha insertado en la tabla

---

getFechaCliente

```
public java.lang.String getFechaCliente(java.lang.String _eMail)
```

Se devuelve la fecha asociada con la inserción de un cliente

Returns:

La fecha en la que el cliente se insertó en el sistema

InsertarAdministrador

```
public boolean InsertarAdministrador(java.lang.String _eMail)
```

Se inserta un nuevo administrador en el sistema.

Parameters:

`_eMail` - Correo electrónico del usuario que se inserta como Administrador

Returns:

True -> Se ha insertado False -> No se ha insertado

---

getFechaAdministrador

```
public java.lang.String getFechaAdministrador(java.lang.String _eMail)
```

Se devuelve la fecha en la que se insertó un Administrador en el sistema

Returns:

La fecha en la cual se insertó en el sitema

---

absolutePage

```
public boolean absolutePage(int _page)
```

Nos lleva a la página que le indiquemos en una búsqueda PRS

Parameters:

`_page` - Es la página de la búsqueda a la que queremos ir.

Returns:

True -> Se ha podido ir a la página indicada False -> Pues no se ha podido ir

---

Actualizar

```
public void Actualizar()
```

Actualiza la búsqueda PRS, si se produce algún cambio y se quieren actualizar los cambios producidos

---

solicitarSerRevisor

```
public boolean solicitarSerRevisor(java.lang.String _eMail,
                                   java.lang.String _exposicion,
                                   int _idRevista)
```

Se inserta una solicitud para ser revisor de alguna de las revistas

Parameters:

`_eMail` - El Email del usuario registrado que solicita ser revisor

`_exposición` - Su explicación que nos da para aceptar su solicitud

`idRevista` - Es el identificador de la revista de la que quiere ser revisor

---

Insertar

```
public boolean Insertar(java.lang.String _login,
                        java.lang.String _eMail,
                        java.lang.String _contrasena,
                        java.lang.String _nombre,
                        java.lang.String _apellidos,
                        int _pais,
                        int _provincia,
                        java.lang.String _poblacion,
                        java.lang.String _calle,
                        int _portal,
                        int _piso,
                        java.lang.String _letra,
                        java.lang.String _codigo,
```

```
                        java.lang.String _telefono,
                        java.sql.Date _fecha,
                        int _sector,
                        int _profesion,
                        int _educacion,
                        java.lang.String _sexo)
```

Se inserta un nuevo Usuario en el sistema.

Parameters:

`_login` - Login que se va insertar en la tabla (No debe haber ninguna tupla con este mismo campo), es obligatoria su insercción.

`_eMail` - eMail que se va insertar en la tabla (No debe haber ninguna tupla con este mismo campo), es obligatoria su insercción.

`_contrasena` - Es la Contrasena que se va insertar en la tabla y es obligatoria su insercción.

`_nombre` - Es el Nombre que se va a insertar en la tabla.

`_apellidos` - Es el valor con el que se va a rellenar el campo Apellidos

`_pais` - Es el valor con el que se rellena el campo Pais si se inserta debe estar asociado con alguna tupla de la tabla Paises.

`_provincia` - Es el valor con el que va a rellenar el campo Provincia y debe estar relacionado si se inserta con alguna tupla de la tabla Provincias

`_poblacion` - Es el valor con el que se rellena el campo Población

`_calle` - Es el valor con el que se rellena el campo Calle

`_portal` - Es el valor con el que se rellena el campo Portal

`_piso` - Es el valor con el que se rellena el campo Piso

`_letra` - Es el valor con el que se rellena el campo Letra.

`_codigo` - Es el valor con el que se rellena el campo CódPostal.

`_telefono` - Es el valor con el que se rellena el campo TeléfonoContacto.

`_fecha` - Es el valor con el que se rellena el campo FechaNacimiento.

`_sector` - Es el valor con el que se rellena el campo Sector y debe de estar asociado con alguna tupla de la tabla Sectores.

`_profesion` - Es el valor con el que se rellena el campo Ocupación y debe de estar asociado con alguna tupla de la tabla Profesiones.

`_educacion` - Es el valor con el que se rellena el campo Estudios y debe de estar asociado con alguna tupla de la tabla Estudios.

`_sexo` - Es el valor con el que se rellena el campo Sexo.

Returns:

True -> Se ha insertado en la tabla False -> No se ha insertado en la tabla

---

setBuscar

```
public boolean setBuscar(java.lang.String _eMail)
```

Sirve para buscar un usuario Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso, Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario

Parameters:

`_eMail` - El Correo electrónico del usuario

Returns:

true -> Se ha encontrado false -> No se ha encontrado

---

buscar

```
public void buscar(boolean _isRegistrados,
                   boolean _isAdministradores,
                   boolean _isClientes,
                   boolean _isCriticos)
```

Busca a todos los usuarios que este dentro de alguno de estos grupos, se trata de una búsqueda PRS Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso,

Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario
Parameters:
`_isRegistrados` - True -> Busca a los usuarios registrados del sistema
`_isAdministradores` - True -> Busca a los usuarios que sean tambien administradores
`_isCliente` - True --> Busca a los usuarios que sean también Clientes
`_isCriticos` - True -> Busca a los usuarios que sean también Criticos

---

getId
`public int` **`getId`**`(java.lang.String _eMail)`
Para conseguir el identificador del usuario mediante su eMail
Parameters:
`_eMail` - Es el correo electrónico del usuario.
Returns:
identificador

---

getEmail
`public java.lang.String` **`getEmail`**`(int _id)`
Para conseguir el eMail del usario pasándole su identificador.
Parameters:
`_id` - Es el identificador que identifica al usuario
Returns:
Su eMail

---

getEmail
`public java.lang.String` **`getEmail`**`()`
Para conseguir el eMail de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo eMail de la tupla

---

getLogin
`public java.lang.String` **`getLogin`**`()`
Para conseguir el Login de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo Login de la tupla

---

getContrasenia
`public java.lang.String` **`getContrasenia`**`()`
Para conseguir la Contraseña de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo Contraseña de la tupla

---

getNombre
`public java.lang.String` **`getNombre`**`()`
Para conseguir el Nombre de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo Nombre de la tupla

---

getApellidos
`public java.lang.String` **`getApellidos`**`()`
Para conseguir los Apellidos de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo Apellidos de la tupla

---

getSexo
`public java.lang.String` **`getSexo`**`()`

Para conseguir el Sexo de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo Sexo de la tupla

---

getPais
`public int` **`getPais`**`()`
Para conseguir el Pais de la tupla encontrada en la búsqueda.
Returns:
El valor contenido en el campo Pais de la tupla. Es un entero que es el identificador de un pais de la tabla Paises

---

getCalle
`public java.lang.String` **`getCalle`**`()`
Para conseguir la Calle de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo Calle de la tupla

---

getLetra
`public java.lang.String` **`getLetra`**`()`
Para conseguir la Letra de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo Letra de la tupla

---

getPortal
`public java.lang.String` **`getPortal`**`()`
Para conseguir el Portal de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo Portal de la tupla. Devuelve 0 si no hay ningún valor.

---

getSector
`public int` **`getSector`**`()`
Para conseguir el Sector de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo Sector de la tupla, es el identificador de un sector de la tabla sectores.

---

getPiso
`public java.lang.String` **`getPiso`**`()`
Para conseguir el Piso de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo Piso de la tupla, devuelve 0 si no hay ningún valor.

---

getPoblacion
`public java.lang.String` **`getPoblacion`**`()`
Para conseguir la Poblacion de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo Poblacion de la tupla

---

getCodigo
`public java.lang.String` **`getCodigo`**`()`
Para conseguir el CódPostal de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo CódPostal de la tupla

---

getProvincia
`public int` **`getProvincia`**`()`
Para conseguir la Provincia de la tupla encontrada en la búsqueda
Returns:

El valor contenido en el campo Provincia de la tupla, este es el identificador de una provincia contenida en la tabla Provincias.

---

getTelefono

`public java.lang.String` **`getTelefono`**`()`

Para conseguir el TeléfonoContacto de la tupla encontrada en la búsqueda

Returns:

El valor contenido en el campo TeléfonoContacto de la tupla

---

getAnio

`public java.lang.String` **`getAnio`**`()`

Para conseguir el Año almacenado en la Fecha de la tupla encontrada en la búsqueda (Es el año de nacimiento)

Returns:

El año contenido en el campo Fecha de la tupla, si no hay ninguna fecha se devuelve la cadena "año"

---

getDia

`public java.lang.String` **`getDia`**`()`

Para conseguir el día almacenado en la Fecha de la tupla encontrada en la búsqueda (Es el día de nacimiento)

Returns:

El día contenido en el campo Fecha de la tupla, si no hay ninguna fecha se devuelve la cadena vacía.

---

getMes

`public java.lang.String` **`getMes`**`()`

Para conseguir el mes almacenado en la Fecha de la tupla encontrada en la búsqueda (Es el mes de nacimiento)

Returns:

El mes contenido en el campo Fecha de la tupla, si no hay ninguna fecha se devuelve la cadena vacía.

---

getFecha

`public java.lang.String` **`getFecha`**`()`

Para conseguir la fecha completa almacenada en la tupla encontrada en la búsqueda (fecha nacimiento)

Returns:

Una cadena con la fecha del tipo: 4 de Agosto de 1978

---

getEducacion

`public int` **`getEducacion`**`()`

Para conseguir el Estudio de la tupla encontrada en la búsqueda

Returns:

El valor contenido en el campo Estudios de la tupla, este es el identificador de un estudio almacenado en la tabla Estudios.

---

getProfesion

`public int` **`getProfesion`**`()`

Para conseguir la Ocupación de la tupla encontrada en la búsqueda

Returns:

El valor contenido en el campo Ocupación de la tupla, este es el identificador de una ocupación almacenada en la tabla Profesiones.

---

getRecibirEmail

`public java.lang.String` **`getRecibirEmail`**`()`

Para conseguir la Ocupación de la tupla encontrada en la búsqueda

Returns:

El valor contenido en el campo Ocupación de la tupla, este es el identificador de una ocupación almacenada en la tabla Profesiones.

---

isRecibirEmail
`public boolean` **`isRecibirEmail`**`()`
Para conseguir el campo RecibirEmails de la tupla encontrada en la búsqueda
Returns:
El valor contenido en el campo RecibirEmails de la tupla. en la tabla Profesiones.

---

isRecibirEmail
`public boolean` **`isRecibirEmail`**`(java.lang.String _eMail)`
Para saber si el usuario tiene activada la recepcion de eMails, tupla enocntrada en la búsqueda campo RecibirEmails
Parameters:
`_eMail` - Es el E-Mail del usuario.
Returns:
Si tiene o no activada la recepción de E-Mails

---

getRecibirNotas
`public java.lang.String` **`getRecibirNotas`**`()`
Para saber si el usuario debe de recibir las notas también por E-Mail, tupla de la búsqueda y campo RecibirNotas
Returns:
True -> Recibe las notas también via eMail.

---

isRecibirNotas
`public boolean` **`isRecibirNotas`**`()`
Para saber si el usuario debe de recibir las notas también por E-Mail, tupla de la búsqueda y campo RecibirNotas
Returns:
True -> Recibe las notas también via eMail.

---

getNoticias
`public java.lang.String` **`getNoticias`**`()`
Nos indica si el usuario de la búsqueda recibe las noticias también por correo electrónico
Returns:
True -> Las recibe también vía eMail False -> Pues no

---

isNoticias
`public boolean` **`isNoticias`**`()`
Nos indica si el usuario de la búsqueda recibe las noticias también por correo electrónico, campo RecibirNoticias de la búsqueda
Returns:
True -> Las recibe también vía eMail False -> Pues no

---

getFirma
`public java.lang.String` **`getFirma`**`()`
Para conseguier la firma con la que envia las notas y eMail este usuario, campo Firma de la búsqueda
Returns:
Una cadena que es su firma

---

desconectar
`public void` **`desconectar`**`()`
Desconecta de la base de datos

---

setActualizar
```
public boolean setActualizar(java.lang.String _eMail,
                             java.lang.String _firma,
                             boolean _recibirEmail,
                             boolean _noticias,
```

```
                                boolean _recibirNotas)
```

Para actualizar los valores contenido sobre un usuario

Parameters:

`_eMail` - Para reconocer a la tupla que se va a actualizar.
`_firma` - Actualiza con el valor pasado el campo Firma.
`_recibirEmail` - Actualiza con el valor pasado el campo RecibirEmail.
`_noticias` - Actualiza con el valor pasado el campo RecibirNoticias.
`_recibirNotas` - Actualiza con el valor pasado el campo RecibirNotas.

Returns:

True -> La operación se ha realizado con éxito. False -> Pues no se ha realizado con éxito.

---

setActualizar

```
public boolean setActualizar(java.lang.String _eMail,
                             java.lang.String _nombre,
                             java.lang.String _apellidos,
                             int _pais,
                             int _provincia,
                             java.lang.String _poblacion,
                             java.lang.String _calle,
                             int _portal,
                             int _piso,
                             java.lang.String _letra,
                             java.lang.String _codigo,
                             java.lang.String _telefono,
                             java.sql.Date _fecha,
                             int _sector,
                             int _profesion,
                             int _educacion,
                             java.lang.String _sexo)
```

Para actualizar los valores contenido sobre un usuario

Parameters:

`_eMail` - Para reconocer a la tupla que se va a actualizar.
`_nombre` - Actualiza con el valor pasado el campo Nombre.
`_apellidos` - Actualiza con el valor pasado el campo Apellidos.
`_pais` - Actualiza con el valor pasado el campo Pais.
`_provincia` - Actualiza con el valor pasado el campo Provincia.
`_poblacion` - Actualiza con el valor pasado el campo Población.
`_calle` - Actualiza con el valor pasado el campo Calle.
`_portal` - Actualiza con el valor pasado el campo Portal.
`_piso` - Actualiza con el valor pasado el campo Piso.
`_codigo` - Actualiza con el valor pasado el campo CódPostal.
`_telefono` - Actualiza con el valor pasado el campo TeléfonoContacto.
`_fecha` - Actualiza con el valor pasado el campo FechaNacimiento.
`_sector` - Actualiza con el valor pasado el campo Sector.
`_profesion` - Actualiza con el valor pasado el campo Ocupación.
`_educacion` - Actualiza con el valor pasado el campo Estudios.
`_sexo` - Actualiza con el valor pasado el campo Sexo.

Returns:

True -> La operación se ha realizado con éxito. False -> Pues no se ha realizado con éxito.

---

setChangePassword

```
public void setChangePassword(java.lang.String _eMail,
                              java.lang.String _nuevaContra)
```

Para actualizar la contraseña de un usuario

Parameters:

`_eMail` - El correo electrónico del usuario
`_nuevaContra` - Es el valor con el que se actualiza el campo Contraseña.

setListar

```
public boolean setListar()
```

Para hacer una búsqueda en la que aparecen todos los usuario Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso, Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario

Returns:

True -> Se ha realizado con éxito la operación False -> Pues no se ha realizado con éxito la operación

getNext

```
public boolean getNext()
```

Pasa al siguiente elemento de la búsqueda (Para búsquedas normales y no PRS)

Returns:

True -> Se ha realizado con éxito la operación False -> Pues no se ha realizado con éxito la operación

nextPalabraClave

```
public boolean nextPalabraClave()
```

Pasa a la siguiente palabra clave asociada con un usuario previamente habrá que realizar la búsqueda return true -> Hay siguiente elemento y se situa sobre el False -> No hay siguiente elemento

getPalabraClave

```
public java.lang.String getPalabraClave()
```

Para conseguir la palabra clave asociada con un usuario, campo Palabra de la búsqueda

Returns:

la palabra clave

buscarAdministradores

```
public void buscarAdministradores(java.lang.String _eMail,
                                  java.lang.String _login,
                                  java.lang.String _nombre,
                                  int _ordenar,
                                  int _orden)
```

Se realiza una búqueda de administradores, se trata de una búsqueda PRS, se realiza la búsqueda si se introduce un parámetro, si no este se ignora al realizar la búsqueda. Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso, Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario

Parameters:

`_eMail` - Se realiza la búsqueda de todas las tuplas que contengan esa palabra en su eMail.

`_orden` - 1 -> Se ordena por la Login (NICK) 2 -> Se ordena por Nombre 3 -> E-Mail

`_ordenar` - 1 -> Orden ascendente 2 -> Orden descendente

`_nombre` - Se realiza la búsqueda de todas las tuplas que contengan esa palabra en su Nombre

`_login` - Se realiza la búsqueda de todas las tuplas que contengan esa palabra en su Login

buscarClientes

```
public void buscarClientes(java.lang.String _eMail,
                           java.lang.String _login,
                           java.lang.String _nombre,
                           java.lang.String _revista,
                           int _ordenar,
                           int _orden)
```

Se realiza una búqueda de clientes, se trata de una búsqueda PRS, se realiza la búsqueda si se introduce un parámetro Si no este se ignora al realizar la búsqueda. Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso, Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario

Parameters:
_eMail - Se realiza la búsqueda de todas las tuplas que contengan esa palabra en su eMail.
_orden - 1 -> Se ordena por la Login (NICK) 2 -> Se ordena por Nombre 3 -> E-Mail
_ordenar - 1 -> Orden ascendente 2 -> Orden descendente
_nombre - Se realiza la búsqueda de todas las tuplas que contengan esa palabra en su Nombre
_login - Se realiza la búsqueda de todas las tuplas que contengan esa palabra en su Login
_revista - Se realiza la búsqueda de todas las tuplas que contenga esa palabra en el título de alguna de sus revistas asociadas

---

buscarRevisores

```
public void buscarRevisores(java.lang.String _eMail,
                            java.lang.String _login,
                            java.lang.String _nombre,
                            java.lang.String _revista,
                            int _ordenar,
                            int _orden)
```

Se realiza una búqueda de revisores, se trata de una búsqueda PRS, sólo se realiza la búsqueda si se introduce un parámetro Si no este se ignora al realizar la búsqueda. Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso, Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario
Parameters:
_eMail - Se realiza la búsqueda de todas las tuplas que contengan esa palabra en su eMail.
_orden - 1 -> Se ordena por la Login (NICK) 2 -> Se ordena por Nombre 3 -> E-Mail
_ordenar - 1 -> Orden ascendente 2 -> Orden descendente
_nombre - Se realiza la búsqueda de todas las tuplas que contengan esa palabra en su Nombre
_login - Se realiza la búsqueda de todas las tuplas que contengan esa palabra en su Login
_revista - Se realiza la búsqueda de todas las tuplas que contenga esa palabra en el título de alguna de sus revistas de las cuales son críticos.

---

buscarRevisores

```
public void buscarRevisores(java.lang.String _eMail,
                            java.lang.String _login,
                            java.lang.String _revista,
                            int _ordenar,
                            int _orden)
```

Se realiza una búqueda de revisores, se trata de una búsqueda PRS, sólo se realiza la búsqueda si se introduce un parámetro, si no este se ignora al realizar la búsqueda. Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso, Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario
Parameters:
_eMail - Se realiza la búsqueda de todos los revisores donde sean revisores de alguna de las revistas del cliente que tenga este E-Mail.
_login - Es el login que tiene que tener el revisor
_revista - Es el nombre de la revista.
_ordenar - El campo por el cual se ordenan los datos
_orden - 1 Ascdentemente 2 Descendentemente

---

buscarRevisores

```
public void buscarRevisores(int _id)
```

Se realiza una búqueda de los revisores atendiendo a la revista que revisan Atributos devueltos por la búsqueda: eMail, Login, Contraseña, Nombre, Apellidos, Calle, Letra, Piso, Portal, Población, CódPostal, Provincia, TeléfonoContacto, FechaNacimiento, Estudios, Sector, Ocupación, RecibirNotas, RecibirEmails, Sexo, Pais, RecibirNoticias, Firma, IdUsuario
Parameters:

`_id` - Identificador de la revista que deven de revisar

---

topPage
`public boolean` **`topPage`**`()`
Nos ponemos sobre el primer elemento de la página sobre la que estamos situados en una búsqueda PRS

---

setNoNext
`public void` **`setNoNext`**`()`
Sirve para evitar avanzar el cursor sobre el resultado de la búsqueda al realizar el siguiente Next o alguna de sus variantes. Muy util si se cambia de página y no queremos que en la primera iteración del Next realice un avance. Asi podemos tratar a todos los elementos de la misma forma en bucle while. Ya que al realizar el bucle se saltaria al hacer el next el elemento sobre el cual estamos situados (Al pasar de página nunca podemos estar antes del primer elemento,siempre nos dejará en el primero).

---

next
`public boolean` **`next`**`()`
Pasa al siguiente elemento de la búsqueda PRS
Returns:
boolean True -> Hay siguiente elemento y se ha podido ir a este False -> Pues no

---

nextInPage
`public boolean` **`nextInPage`**`()`
Se va al siguiente elemento dentro de una página en una búsqueda PRS
Returns:
True -> Hay elemento posterior y nos hemos situados sobre el False -> No hay elemento posterior

---

getPrsEMail
`public java.lang.String` **`getPrsEMail`**`()`
Para coseguir el E-Mail de la tupla en la que estamos situados de la búsqueda PRS
Returns:
Nos devulve el eMail

---

getPrsLogin
`public java.lang.String` **`getPrsLogin`**`()`
Para coseguir el Login de la tupla en la que estamos situados de la búsqueda PRS.
Returns:
Nos devulve el Login

---

getPrsNombre
`public java.lang.String` **`getPrsNombre`**`()`
Para coseguir el Nombre de la tupla en la que estamos situados de la búsqueda PRS
Returns:
Nos devulve el Nombre

---

isPrsRecibirEmails
`public boolean` **`isPrsRecibirEmails`**`()`
Para coseguir el campo RecibirEmails de la tupla en la que estamos situados de la búsqueda PRS
Returns:
Nos devulve el Nombre

---

isPaginaAnterior
`public boolean` **`isPaginaAnterior`**`()`
Para saber si hay alguna página anterior sobre la cual estamos situados en una búsqueda PRS
Returns:
True -> Hay una página anterior False -> No hay ninguna página anterior

---

isPaginaSiguiente

`public boolean` **`isPaginaSiguiente`**`()`
Para saber si hay alguna página posterior sobre la cual estamos situados en una búsqueda PRS
Returns:
True -> Hay una página False -> No la hay

---

getPage
`public java.lang.String` **`getPage`**`()`
Para saber la página sobre la que estamos situados en una búsqueda PRS
Returns:
Nos devuelve un String indicándonos el número de la página sobre la que estamos

---

getPages
`public java.lang.String` **`getPages`**`()`
Para saber cuantas páginas ha devuelto la búsqueda PRS
Returns:
No devuelve el número de páginas en un String

---

nextPage
`public boolean` **`nextPage`**`()`
Se va a la siguiente página de la búsqueda PRS y nos situamos sobre el primer elemento.
Returns:
True -> Hay siguiente página y nos hemos situado en su primer elemento False -> No hay siguiente página

---

previousPage
`public boolean` **`previousPage`**`()`
Se va a la página anterior de una búsqueda PRS y nos situamos sobre el primer elemento.
Returns:
True -> Hay elemento anterior y nos hemos situados sobre el False -> No hay elemento anterior

---

first
`public boolean` **`first`**`()`
Nos posicionamos en el primer elemento de una búsqueda PRS
Returns:
true -> Se ha podido ir False -> No se ha podido ir.

---

beforeFirst
`public void` **`beforeFirst`**`()`
Nos posicionamos antes del primer elemento de la búsqueda PRS

---

firstPage
`public boolean` **`firstPage`**`()`
Se va a la primera página de la búsqueda PRS y nos situamos sobre el primer elemento.
Returns:
True -> Hay primer página y nos hemos situado en su primer elemento False -> No hay ninguna página

---

lastPage
`public boolean` **`lastPage`**`()`
Se va a la última página de la búsqueda PRS y nos situamos sobre su primer elemento.
Returns:
True -> Nos hemos situado en la última página y en su primer elemento False -> No hay ninguna página

---

getNick
`public java.lang.String` **`getNick`**`(java.lang.String _eMail)`
Nos dice el nick del cliente que ha creado la revista
Parameters:
`_eMail` - Es el E-Mail del creador de la revista

Returns:
su nick

---

insertarPalabraClave

```
public boolean insertarPalabraClave(java.lang.String _eMail,
                                    java.lang.String _palabra)
```

Inserta una palabra clave asociada con una revisor (las palabras clave son palabras que describen los temas en los que es un experto este revisor)
Parameters:
`_eMail` - Es el identificado del revisor
`_palabra` - Es la palabra clave que se inserta
Returns:
True -> Se ha insertado correctamente False -> Se ha producido algún error

---

setBuscarPalabrasClave

```
public void setBuscarPalabrasClave(java.lang.String _eMail)
```

Busca las palabras clave asociadas a una determinado Revisor Atributos de las tuplas devueltas por la búsqueda: eMail, Palabra
Parameters:
`_eMail` - Correo electrónico del usuario

---

setEliminarPalabrasClave

```
public boolean setEliminarPalabrasClave(java.lang.String _eMail)
```

Elimina las palabras clave asociadas con un usuario
Parameters:
`_eMail` - Correo electrónico del usuario al cual queremos eliminar sus palabras clave
Returns:
true -> Se ha eliminado False -> No se ha eliminado correctamente

---

finalize

```
protected void finalize()
                throws java.lang.Throwable
```

Cuando se elimina un objeto de esta clase habrá que cerrar las consultas y conexiones que aun pudieran estar abiertas
Overrides:
`finalize` in class `java.lang.Object`

tablas

Class Constantes

```
java.lang.Object
  |
  +--tablas.Constantes
```

All Implemented Interfaces:
java.io.Serializable

---

public class **Constantes**
extends java.lang.Object
implements java.io.Serializable
Clase que contiene una serie de constantes
See Also:
Serialized Form

Field Summary

static int	**ADMINISTRADOR** Representa a los usuarios que son Administradores del sistema
static int	**ASCENDENTE** Búsquedas Acendente
static int	**CLIENTE** Representa a los usuarios que son Clientes del sistema
static int	**CLIENTE2** Representa a los usuarios que son Clientes, pero que cuando mandan notas estas están dirigidas a todos los administradores, en vez de a un solo usuario
static int	**DESCENDENTE** Búsquedas descendentes
static int	**EMAIL** Búsquedas realizadas por Email
static int	**ERROR** Cuando se comete un Error
static int	**LOGIN** Búsquedas realizadas por Login
static int	**MAXCARACTERES** Numero de caracteres que se muestran por linea en las secciones de tercer nivel en una revista Es decir si se pasa de estos caracteres las secciones siguientes se mostrarán en otra línea
static int	**NOMBRE** Búsquedas realizadas por Nombre
static int	**NOREGISTRADO** Representa a los usuarios que son Usuarios No Registrado
static int	**REVISOR** Representa a los usuarios que son Revisores del sistema
static int	**REVISOR2** Representa a los usuarios que son Revisores, pero que cuando mandan notas estas están dirigidas a todos los administradores, en vez de a un solo usuario
static int	**USUARIO** Representa a los usuarios Registrados, pero que cuando mandan notas estas están dirigidas a todos los administradores, en vez de a un solo usuario

Constructor Summary
Constantes()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

ADMINISTRADOR

```
public static final int ADMINISTRADOR
```

Representa a los usuarios que son Administradores del sistema

CLIENTE
`public static final int` **`CLIENTE`**
Representa a los usuarios que son Clientes del sistema

REVISOR
`public static final int` **`REVISOR`**
Representa a los usuarios que son Revisores del sistema

NOREGISTRADO
`public static final int` **`NOREGISTRADO`**
Representa a los usuarios que son Usuarios No Registrado

CLIENTE2
`public static final int` **`CLIENTE2`**
Representa a los usuarios que son Clientes, pero que cuando mandan notas estas están dirigidas a todos los administradores, en vez de a un solo usuario

---

REVISOR2
`public static final int` **`REVISOR2`**
Representa a los usuarios que son Revisores, pero que cuando mandan notas estas están dirigidas a todos los administradores, en vez de a un solo usuario

---

USUARIO
`public static final int` **`USUARIO`**
Representa a los usuarios Registrados, pero que cuando mandan notas estas están dirigidas a todos los administradores, en vez de a un solo usuario

---

ERROR
`public static final int` **`ERROR`**
Cuando se comete un Error

LOGIN
`public static final int` **`LOGIN`**
Búsquedas realizadas por Login

NOMBRE
`public static final int` **`NOMBRE`**
Búsquedas realizadas por Nombre

EMAIL
`public static final int` **`EMAIL`**
Búsquedas realizadas por Email

ASCENDENTE
`public static final int` **`ASCENDENTE`**
Búsquedas Acendente

DESCENDENTE
`public static final int` **`DESCENDENTE`**
Búsquedas descendentes

MAXCARACTERES
`public static final int` **`MAXCARACTERES`**
Numero de caracteres que se muestran por linea en las secciones de tercer nivel en una revista Es decir si se pasa de estos caracteres las secciones siguientes se mostrarán en otra línea

Constructor Detail

Constantes
```
public Constantes()
```

usuario

Class GestorUsuarioBean

```
java.lang.Object
  |
  +--usuario.GestorUsuarioBean
```

All Implemented Interfaces:
java.io.Serializable

---

public class **GestorUsuarioBean**
extends java.lang.Object
implements java.io.Serializable
Clase que da soporte a la gestión de un usuario, todas aquellas operación con una complejidad moderada o alta y que no esten soportadas por las clases que dan soporte a la gestión de las tablas de la Base de Datos. Tienen cambida en esta clase. El fin de esta clase es quitar complejidad al servlet que gestiona los eventos de los usuario, para modular u estructurar la aplicación.
See Also:
Serialized Form

Constructor Summary
GestorUsuarioBean() Crea los objetos que necesita.

Method Summary	
void	**desconectar**() Deberá ser invocado antes de eliminar el objeto, ya que sirve para liberar recursos
java.lang.String	**getAcepto**() Se devuelve si se aceptan los términos del portal.
java.lang.String	**getAnio**() Se recupera el Año que tenomos sobre el usuario.
java.lang.String	**getApellidos**() Se recuperan los Apellidos que tenomos sobre el usuario.
java.lang.String	**getBoletin**() Se devuelve si se quiere recibir los boletines
java.lang.String	**getCalle**() Se recupera la Calle que tenomos sobre el usuario.
java.lang.String	**getCodigo**() Se recupera el Codigo que tenomos sobre el usuario.
java.lang.String	**getDia**() Para utilizar este método previamente hay que utilizar getNoDia y setDia, dependiendo si el dia introducido es igual al introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se muestra
java.lang.String	**getEducacion**() Para utilizar este método previamente hay que utilizar getNoEducacion y putEducacion, dependiendo si la educación introducida es igual a la introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se

	muestra.
java.lang.String	**getEmail**() Se recupera el eMail que tenemo sobre el usuario.
java.lang.String	**getEmail2**() Se recupera el eMail2 que tenemo sobre el usuario.
boolean	**getExisteEMail**(java.lang.String _eMail) Para saber si hay algun usuario con este eMail itroducido en el sistema
java.lang.String	**getGx**() Para utilizar este método previamente hay que utilizar setGx, dependiendo si el gusto introducido esta insertado en la sesión devolverá " checked" para selecionar esta opción de la página en la que se muestra.
java.lang.String	**getHombre**() Si el sexo del usuario de la sesión es Hombre se devolverá " selected", es decir para selecionar esta opción de la lista.
java.lang.String	**getLetra**() Se recupera la Letra que tenomos sobre el usuario.
java.lang.String	**getMes**() Para utilizar este método previamente hay que utilizar getNoMes y putMes, dependiendo si el mes introducido es igual al introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se muestra
java.lang.String	**getMujer**() Si el sexo del usuario de la sesión es Mujer se devolverá " selected", es decir para selecionar esta opción de la lista.
java.lang.String	**getNick**() Se recupera el nick que tenemo sobre el usuario.
java.lang.String	**getNoDia**() Antes de llamar a getDia o setDia se tiene que llamar a este método.
java.lang.String	**getNoEducacion**() Antes de llamar a getNoEducacion o setEducacion se tiene que llamar a este método.
java.lang.String	**getNombre**() Se recuper el Nombre que tenomos sobre el usuario.
java.lang.String	**getNoMes**() Antes de llamar a getMes o setMes se tiene que llamar a este método.
java.lang.String	**getNoPais**() Antes de llamar a getPais o setPais se tiene que llamar a este método.
java.lang.String	**getNoProfesion**() Antes de llamar a getProfesion o setProfesion se tiene que llamar a este método.
java.lang.String	**getNoSector**() Antes de llamar a getSector o setSector se tiene que llamar a este método.
java.lang.String	**getPais**() Para utilizar este método previamente hay que utilizar getNoPais y putPais, dependiendo si el pais introducido es igual al introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se muestra
java.lang.String	**getPiso**() Se recupera el Piso que tenomos sobre el usuario.

java.lang.String	**getPoblacion**() Se recupera la Poblacion que tenomos sobre el usuario.
java.lang.String	**getPortal**() Se recupera el Portal que tenomos sobre el usuario.
java.lang.String	**getProfesion**() Para utilizar este método previamente hay que utilizar getNoProfesion y putProfesion, dependiendo si la profesion introducida es igual a la introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se muestra.
java.lang.String	**getPropaganda**() Se devuelve si se quiere recibir la propaganda.
java.lang.String	**getProvincia**() Se recupera la Provincia que tenomos sobre el usuario.
java.lang.String	**getSector**() Para utilizar este método previamente hay que utilizar getNoSector y setSector, dependiendo si el sector introducido es igual al introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se muestra
java.lang.String	**getTelefono**() Se recupera el Telefono que tenomos sobre el usuario.
void	**init**(javax.servlet.http.HttpSession _sesion) Inicializa el objeto.
boolean	**isCorrect**(java.lang.String _contrasena) Primero hay que realizar una busqueda con setBusUsuario posteriormente se coge la contraseña de este usuario y se compara con la introducida.
void	**setBorrarGustos**(java.lang.String _eMail) Sirve para borrar todos los gustos que hay almacenados en la base de datos sobre un usuario.
void	**setBusUsuario**(java.lang.String _eMail) Buscamos el usuario introducido por parámetro
boolean	**setCambiarDatos**(java.lang.String _eMail, java.lang.String _nombre, java.lang.String _apellidos, java.lang.String _pais, java.lang.String _provincia, java.lang.String _poblacion, java.lang.String _calle, java.lang.String _portal, java.lang.String _piso, java.lang.String _letra, java.lang.String _zipCode, java.lang.String _telefono, java.lang.String _sexo, java.lang.String _anio, java.lang.String _mes, java.lang.String _dia, java.lang.String _sector, java.lang.String _profesion, java.lang.String _educacion) Para cambiar los datos que tenemos almacenados sobre el usuario.
boolean	**setCambioContrasenia**(java.lang.String _eMail, java.lang.String _password1, java.lang.String _password2) Se cambia la contraseña del usuario si estas están correctamente hechas y además coincide.
boolean	**setConfigurar**(java.lang.String _eMail, java.lang.String _firma, java.lang.String _boletin, java.lang.String _noticias, java.lang.String _propaganda) Almacena en la base de datos la configuración del usuario.
void	**setDia**(java.lang.String _dia)

	Se introduce el número del dia para ser comparado con el que tenemos en la sesión del usuario. getDia es el método que nos devuelve si este dia es o no el que estaba selecionado
void	**setEducacion**(java.lang.String _educacion) Se introduce la educación para ser comparada con la que tenemos en la sesión del usuario. getEducacion es el método que nos devuelve si esta educacion es o no la que estaba selecionada.
boolean	**setEliminar**(java.lang.String _eMail) Se elimina al usuario que tenga por eMail el introducido.
void	**setGx**(java.lang.String numero) Se introduce un Gusto para ser comparado con losa que tenemos en la sesión del usuario. getGx es el método que nos devuelve si este gusto esta o no selecionado por el usuario.
void	**setMes**(java.lang.String _mes) Se introduce el número del mes para ser comparado con el que tenemos en la sesión del usuario. getMes es el método que nos devuelve si este mes es o no el que estaba selecionado
void	**setOlvidarDatos**() Se elimina de la sesión los datos introducidos
void	**setOlvidarGustos**() Se elimina de la sesión todos los gustos
void	**setOlvidarRegistroBasico**() Eliminamos de la sesión los datos almacenados en el método setRecordarRegistroBasico.
void	**setPais**(java.lang.String _pais) Se introduci el nombre de un pais para ser comparado con el que tenemos en la sesión del usuario. getPais es el método que nos devuelve si este país es o no el que estaba selecionado
void	**setProfesion**(java.lang.String _profesion) Se introduce la profesion para ser comparada con la que tenemos en la sesión del usuario. getProfesion es el método que nos devuelve si esta profesion es o no la que estaba selecionada.
void	**setRecordarDatos**(java.lang.String _nombre, java.lang.String _apellidos, java.lang.String _pais, java.lang.String _provincia, java.lang.String _poblacion, java.lang.String _calle, java.lang.String _portal, java.lang.String _piso, java.lang.String _letra, java.lang.String _codigo, java.lang.String _telefono, java.lang.String _sexo, java.lang.String _anio, java.lang.String _mes, java.lang.String _dia, java.lang.String _sector, java.lang.String _profesion, java.lang.String _educacion) Guardamos en la sesión los datos que aquí se pasan como parámetros.
void	**setRecordarGusto**(java.lang.Boolean _gusto) Con este método se almacena en la sesión un gusto, para recordar que guston ha introducido el usuario para su registro.
void	**setRecordarGusto**(java.lang.String _gusto) Con este método se almacena en la sesión un gusto, para recordar que guston ha introducido el usuario para su registro.
void	**setRecordarRegistroBasico**(java.lang.String _nick,

	java.lang.String _eMail, java.lang.String _eMail2, java.lang.String _password1, java.lang.String _password2, java.lang.String _nombre, java.lang.String _apellidos, java.lang.String _pais, java.lang.String _provincia, java.lang.String _poblacion, java.lang.String _calle, java.lang.String _portal, java.lang.String _piso, java.lang.String _letra, java.lang.String _codigo, java.lang.String _telefono, java.lang.String _sexo, java.lang.String _anio, java.lang.String _mes, java.lang.String _dia, java.lang.String _sector, java.lang.String _profesion, java.lang.String _educacion, java.lang.String _acepto) Almacenamos los datos pasados en la sesión si esta activada la visualización estos datos serán los devueltos por el objeto.
void	**setRecuperarDatos**(java.lang.String _eMail) Se almacena en la sesión los datos que tenemos de un usuario determinado que está almacenado en la base de datos.
boolean	**setRegistroBasico**(java.lang.String _nick, java.lang.String _eMail, java.lang.String _eMail2, java.lang.String _password1, java.lang.String _password2, java.lang.String _nombre, java.lang.String _apellidos, java.lang.String _pais, java.lang.String _provincia, java.lang.String _poblacion, java.lang.String _calle, java.lang.String _portal, java.lang.String _piso, java.lang.String _letra, java.lang.String _codigo, java.lang.String _telefono, java.lang.String _sexo, java.lang.String _anio, java.lang.String _mes, java.lang.String _dia, java.lang.String _sector, java.lang.String _profesion, java.lang.String _educacion, java.lang.String _acepto) Se registra a un nuevo usuario.
boolean	**setRegistroGusto**(java.lang.String _eMail, java.lang.String _gusto) Sirve para registrar un gusto en la base de datos asociado a un usuario.
void	**setSector**(java.lang.String _sector) Se introduce el sector para ser comparado con el que tenemos en la sesión del usuario. getSector es el método que nos devuelve si este sector es o no el que estaba selecionado
void	**setVisualizar**(boolean _error) Sirve para que cuando le pidamos los datos del usuario le devolvamos los que tenemos almacenados sobre el o si lo preferimos los valores por defecto.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

GestorUsuarioBean

public **GestorUsuarioBean**()

Crea los objetos que necesita.

Method Detail

init

```
public void init(javax.servlet.http.HttpSession _sesion)
```

Inicializa el objeto.

Parameters:

`_sesion` - Se le pasa la sesión del usuario que esté navegando.

---

setVisualizar

```
public void setVisualizar(boolean _error)
```

Sirve para que cuando le pidamos los datos del usuario le devolvamos los que tenemos almacenados sobre el o si lo preferimos los valores por defecto. Es muy util cuando un usuario se registra ya recordamos los valores que introdujo y despues si se ha cometido un error se le muestra la misma página indicando los errores pero que sigan con los campos que ha introducido. En cambio si abrimos una página en limpio se muestran los valores por defecto o en blanco que tenemos de un usuario. Por defecto está desactivado.

Parameters:

`True` - Para que se devuelvan los datos del usuario False se le devolverán los valores nulos o por defecto.

---

setBorrarGustos

```
public void setBorrarGustos(java.lang.String _eMail)
```

Sirve para borrar todos los gustos que hay almacenados en la base de datos sobre un usuario.

Parameters:

`_eMail` - Es el identificador (eMail) del usuario.

---

setRegistroGusto

```
public boolean setRegistroGusto(java.lang.String _eMail,
                                java.lang.String _gusto)
```

Sirve para registrar un gusto en la base de datos asociado a un usuario.

Parameters:

`_eMail` - Es el identificador (eMail) del usuario.

`_gusto` - Es el nombre del gusto si este está en la base de datos se procederá a asociarlo con el usuario.

Returns:

True -> Se a realizado con éxito la operación False -> No se ha realizado con éxito la operación.

---

setRecordarRegistroBasico

```
public void setRecordarRegistroBasico(java.lang.String _nick,
                                      java.lang.String _eMail,
                                      java.lang.String _eMail2,
                                      java.lang.String _password1,
                                      java.lang.String _password2,
                                      java.lang.String _nombre,
                                      java.lang.String _apellidos,
                                      java.lang.String _pais,
                                      java.lang.String _provincia,
                                      java.lang.String _poblacion,
                                      java.lang.String _calle,
                                      java.lang.String _portal,
                                      java.lang.String _piso,
                                      java.lang.String _letra,
                                      java.lang.String _codigo,
                                      java.lang.String _telefono,
                                      java.lang.String _sexo,
                                      java.lang.String _anio,
                                      java.lang.String _mes,
                                      java.lang.String _dia,
                                      java.lang.String _sector,
                                      java.lang.String _profesion,
                                      java.lang.String _educacion,
                                      java.lang.String _acepto)
```

Almacenamos los datos pasados en la sesión si esta activada la visualización estos datos serán los devueltos por el objeto. Nos sirve por si están mal introducidos los datos de registro del usuario y queremos volver a visualizar los datos que había introducido. Los parámetros introducidos son los que posteriormente podremos haceder a ellos mediante los metodos get y el nombre de la propiedad

---

setOlvidarRegistroBasico

```
public void setOlvidarRegistroBasico()
```

Eliminamos de la sesión los datos almacenados en el método setRecordarRegistroBasico.

---

setConfigurar

```
public boolean setConfigurar(java.lang.String _eMail,
                             java.lang.String _firma,
                             java.lang.String _boletin,
                             java.lang.String _noticias,
                             java.lang.String _propaganda)
```

Almacena en la base de datos la configuración del usuario.

Parameters:

`_eMail` - Es el identifiador del usuario sobre el cual queremos almacenar su configuración.

`_firma` - Es la firma del usuario, es decir será el texto que se añada a cada uno de los mensages que envíe el usuario.

`_boletin` - Indica si quiere que el usuario reciba boletines informativos a su correo electrónico.

`_noticias` - Indica si quiere que el usuario reciba noticias a su correo electrónico.

`_propaganda` - Indica si quiere el usuario recibir propaganda y promociones del portal.

Returns:

True -> Se ha insertado la configuración del usuario False -> No se ha podido insertar

---

setCambiarDatos

```
public boolean setCambiarDatos(java.lang.String _eMail,
                               java.lang.String _nombre,
                               java.lang.String _apellidos,
                               java.lang.String _pais,
                               java.lang.String _provincia,
                               java.lang.String _poblacion,
                               java.lang.String _calle,
                               java.lang.String _portal,
                               java.lang.String _piso,
                               java.lang.String _letra,
                               java.lang.String _zipCode,
                               java.lang.String _telefono,
                               java.lang.String _sexo,
                               java.lang.String _anio,
                               java.lang.String _mes,
                               java.lang.String _dia,
                               java.lang.String _sector,
                               java.lang.String _profesion,
                               java.lang.String _educacion)
```

Para cambiar los datos que tenemos almacenados sobre el usuario.

Parameters:

`_eMail` - Es el identifiador del usuario sobre el cual queremos almacenar su configuración.

Returns:

True -> Se ha insertado la configuración del usuario False -> No se ha podido insertar

---

setRegistroBasico

```
public boolean setRegistroBasico(java.lang.String _nick,
                                 java.lang.String _eMail,
                                 java.lang.String _eMail2,
                                 java.lang.String _password1,
                                 java.lang.String _password2,
                                 java.lang.String _nombre,
```

```
                                    java.lang.String _apellidos,
                                    java.lang.String _pais,
                                    java.lang.String _provincia,
                                    java.lang.String _poblacion,
                                    java.lang.String _calle,
                                    java.lang.String _portal,
                                    java.lang.String _piso,
                                    java.lang.String _letra,
                                    java.lang.String _codigo,
                                    java.lang.String _telefono,
                                    java.lang.String _sexo,
                                    java.lang.String _anio,
                                    java.lang.String _mes,
                                    java.lang.String _dia,
                                    java.lang.String _sector,
                                    java.lang.String _profesion,
                                    java.lang.String _educacion,
                                    java.lang.String _acepto)
```

Se registra a un nuevo usuario. Si hay algún error se recuerda el error que se ha producido. Los parámetros se coresponden con los datos que el usuario a introducido.

Returns:

True -> Se ha insertado al nuevo usuario False -> No se ha podido insertar

---

getNick

```
public java.lang.String getNick()
```

Se recupera el nick que tenemo sobre el usuario. Puede provenir de la sesión o de la base de datos o el valor por defecto dependiendo a que métodos hemos llamado antes.

Returns:

Devuelve el valor quetenemos sobre el nick

---

getEmail

```
public java.lang.String getEmail()
```

Se recupera el eMail que tenemo sobre el usuario. Puede provenir de la sesión, de la base de datos o el valor por defecto dependiendo a que métodos hemos llamado antes.

Returns:

Devuelve el valor quetenemos sobre el nick

---

getEmail2

```
public java.lang.String getEmail2()
```

Se recupera el eMail2 que tenemo sobre el usuario. Puede provenir de la sesión o puede estar el valor por defecto.

Returns:

Devuelve el valor quetenemos sobre el nick

---

getNombre

```
public java.lang.String getNombre()
```

Se recuper el Nombre que tenomos sobre el usuario. Puede provenir de la sesió, pues estar su valor por defecto , dependerá del método que se ha llamado con anterioridad.

Returns:

Devuelve el valor que tenemos sobre el nombre.

---

getApellidos

```
public java.lang.String getApellidos()
```

Se recuperan los Apellidos que tenomos sobre el usuario. Puede provenir de la sesió, pues estar su valor por defecto , dependerá del método que se ha llamado con anterioridad.

Returns:

Devuelve el valor que tenemos sobre los apellidos.

---

getPais

`public java.lang.String` **`getPais`**`()`
Para utilizar este método previamente hay que utilizar getNoPais y putPais, dependiendo si el pais introducido es igual al introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se muestra
Returns:
Devuelve la cadena vacía si el pais introducido no se corresponde con el que se está recordando en la sesión si fuese igual devuelve " selected" para indicar que del menú desplegable este pais es el que se había introducido

---

setPais
`public void` **`setPais`**`(java.lang.String _pais)`
Se introduci el nombre de un pais para ser comparado con el que tenemos en la sesión del usuario. getPais es el método que nos devuelve si este país es o no el que estaba selecionado

---

getNoPais
`public java.lang.String` **`getNoPais`**`()`
Antes de llamar a getPais o setPais se tiene que llamar a este método.
Returns:
Nos devuelve " selected" si no hay ningún pais en la sesión del usuario, es decir no se ha introducido previamente ningún pais. Si hay algún pais se devuelve la cadena vacía.

---

getProvincia
`public java.lang.String` **`getProvincia`**`()`
Se recupera la Provincia que tenomos sobre el usuario. Puede provenir de la sesión, pues estar su valor por defecto , dependerá del método que se ha llamado con anterioridad.
Returns:
Devuelve el valor que tenemos sobre la provincia.

---

getPoblacion
`public java.lang.String` **`getPoblacion`**`()`
Se recupera la Poblacion que tenomos sobre el usuario. Puede provenir de la sesió, pues estar su valor por defecto , dependerá del método que se ha llamado con anterioridad.
Returns:
Devuelve el valor que tenemos sobre la Poblacion.

---

getCalle
`public java.lang.String` **`getCalle`**`()`
Se recupera la Calle que tenomos sobre el usuario. Puede provenir de la sesión, pues estar su valor por defecto dependerá del método que se ha llamado con anterioridad.
Returns:
Devuelve el valor que tenemos sobre la Calle.

---

getPortal
`public java.lang.String` **`getPortal`**`()`
Se recupera el Portal que tenomos sobre el usuario. Puede provenir de la sesión, pues estar su valor por defecto dependerá del método que se ha llamado con anterioridad.
Returns:
Devuelve el valor que tenemos sobre el Portal.

---

getPiso
`public java.lang.String` **`getPiso`**`()`
Se recupera el Piso que tenomos sobre el usuario. Puede provenir de la sesión, pues estar su valor por defecto dependerá del método que se ha llamado con anterioridad.
Returns:
Devuelve el valor que tenemos sobre el Piso.

---

getLetra

`public java.lang.String` **`getLetra`**`()`
Se recupera la Letra que tenomos sobre el usuario. Puede provenir de la sesión, pues estar su valor por defecto dependerá del método que se ha llamado con anterioridad.
Returns:
Devuelve el valor que tenemos sobre la Letra.

---

getCodigo
`public java.lang.String` **`getCodigo`**`()`
Se recupera el Codigo que tenomos sobre el usuario. Puede provenir de la sesión, pues estar su valor por defecto dependerá del método que se ha llamado con anterioridad.
Returns:
Devuelve el valor que tenemos sobre el Codigo.

---

getTelefono
`public java.lang.String` **`getTelefono`**`()`
Se recupera el Telefono que tenomos sobre el usuario. Puede provenir de la sesión, pues estar su valor por defecto dependerá del método que se ha llamado con anterioridad.
Returns:
Devuelve el valor que tenemos sobre el Telefono.

---

getHombre
`public java.lang.String` **`getHombre`**`()`
Si el sexo del usuario de la sesión es Hombre se devolverá " selected", es decir para selecionar esta opción de la lista.
Returns:
Devuelve " selected" si el hombre en caso de ser Mujer la cadena vacía.

---

getMujer
`public java.lang.String` **`getMujer`**`()`
Si el sexo del usuario de la sesión es Mujer se devolverá " selected", es decir para selecionar esta opción de la lista.
Returns:
Devuelve " selected" si es Mujer en caso de ser Hombre la cadena vacía.

---

getAnio
`public java.lang.String` **`getAnio`**`()`
Se recupera el Año que tenomos sobre el usuario. Puede provenir de la sesión, pues estar su valor por defecto dependerá del método que se ha llamado con anterioridad.
Returns:
Devuelve el valor que tenemos sobre el Año de nacimiento del usuario, si no hay ninguno en la sesión se devuelve la cadena "año".

---

setMes
`public void` **`setMes`**`(java.lang.String _mes)`
Se introduce el número del mes para ser comparado con el que tenemos en la sesión del usuario. getMes es el método que nos devuelve si este mes es o no el que estaba selecionado
Parameters:
`_mes` - Se corresponde con el número del mes, del cual queremos saber si estaba o no selecionado.

---

getNoMes
`public java.lang.String` **`getNoMes`**`()`
Antes de llamar a getMes o setMes se tiene que llamar a este método.
Returns:
Nos devuelve " selected" si no hay ningún mes en la sesión del usuario, es decir no se ha introducido previamente ningún mes. Si hay algún mes se devuelve la cadena vacía.

---

getMes

`public java.lang.String` **`getMes`**`()`
Para utilizar este método previamente hay que utilizar getNoMes y putMes, dependiendo si el mes introducido es igual al introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se muestra
Returns:
Devuelve la cadena vacía si el mes introducido no se corresponde con el que se está recordando en la sesión si fuese igual devuelve " selected" para indicar que del menú desplegable este mes es el que se había introducido

---

getNoDia
`public java.lang.String` **`getNoDia`**`()`
Antes de llamar a getDia o setDia se tiene que llamar a este método.
Returns:
Nos devuelve " selected" si no hay ningún dia en la sesión del usuario, es decir no se ha introducido previamente ningún dia. Si hay algún dia se devuelve la cadena vacía.

---

getDia
`public java.lang.String` **`getDia`**`()`
Para utilizar este método previamente hay que utilizar getNoDia y setDia, dependiendo si el dia introducido es igual al introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se muestra
Returns:
Devuelve la cadena vacía si el dia introducido no se corresponde con el que se está recordando en la sesión si fuese igual devuelve " selected" para indicar que del menú desplegable este dia es el que se había introducido

---

setDia
`public void` **`setDia`**`(java.lang.String _dia)`
Se introduce el número del dia para ser comparado con el que tenemos en la sesión del usuario. getDia es el método que nos devuelve si este dia es o no el que estaba selecionado
Parameters:
`_dia` - Se corresponde con el número del dia, del cual queremos saber si estaba o no selecionado.

---

getNoSector
`public java.lang.String` **`getNoSector`**`()`
Antes de llamar a getSector o setSector se tiene que llamar a este método.
Returns:
Nos devuelve " selected" si no hay ningún Sector en la sesión del usuario, es decir no se ha introducido previamente ningún Sector. Si hay algún Sector se devuelve la cadena vacía.

---

getSector
`public java.lang.String` **`getSector`**`()`
Para utilizar este método previamente hay que utilizar getNoSector y setSector, dependiendo si el sector introducido es igual al introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se muestra
Returns:
Devuelve la cadena vacía si el sector introducido no se corresponde con el que se está recordando en la sesión si fuese igual devuelve " selected" para indicar que del menú desplegable este sector es el que se había introducido.

---

setSector
`public void` **`setSector`**`(java.lang.String _sector)`
Se introduce el sector para ser comparado con el que tenemos en la sesión del usuario. getSector es el método que nos devuelve si este sector es o no el que estaba selecionado
Parameters:
`_sector` - Se corresponde con el sector, del cual queremos saber si estaba o no selecionado.

---

getNoProfesion
`public java.lang.String **getNoProfesion**()`
Antes de llamar a getProfesion o setProfesion se tiene que llamar a este método.
Returns:
Nos devuelve " selected" si no hay ningúna Profesion en la sesión del usuario, es decir no se ha introducido previamente. Si hay algúna Profesion se devuelve la cadena vacía.

---

getProfesion
`public java.lang.String **getProfesion**()`
Para utilizar este método previamente hay que utilizar getNoProfesion y putProfesion, dependiendo si la profesion introducida es igual a la introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se muestra.
Returns:
Devuelve la cadena vacía si la profesion introducida no se corresponde con el que se está recordando en la sesión si fuese igual devuelve " selected" para indicar que del menú desplegable esta profesion es la que se había introducido.

---

setProfesion
`public void **setProfesion**(java.lang.String _profesion)`
Se introduce la profesion para ser comparada con la que tenemos en la sesión del usuario. getProfesion es el método que nos devuelve si esta profesion es o no la que estaba selecionada.
Parameters:
`_profesion` - Se corresponde con la profesion, de la cual queremos saber si estaba o no selecionada.

---

getNoEducacion
`public java.lang.String **getNoEducacion**()`
Antes de llamar a getNoEducacion o setEducacion se tiene que llamar a este método.
Returns:
Nos devuelve " selected" si no hay ningúna Educacion en la sesión del usuario, es decir no se ha introducido previamente. Si hay algúna Educación se devuelve la cadena vacía.

---

getEducacion
`public java.lang.String **getEducacion**()`
Para utilizar este método previamente hay que utilizar getNoEducacion y putEducacion, dependiendo si la educación introducida es igual a la introducido en la sesión devolverá " selected" para selecionar esta opción de la página en la que se muestra.
Returns:
Devuelve la cadena vacía si la educacion introducida no se corresponde con el que se está recordando en la sesión si fuese igual devuelve " selected" para indicar que del menú desplegable esta educacion es la que se había introducido.

---

setEducacion
`public void **setEducacion**(java.lang.String _educacion)`
Se introduce la educación para ser comparada con la que tenemos en la sesión del usuario. getEducacion es el método que nos devuelve si esta educacion es o no la que estaba selecionada.
Parameters:
`_educacion` - Se corresponde con la educacion, de la cual queremos saber si estaba o no selecionada.

---

getBoletin
`public java.lang.String **getBoletin**()`
Se devuelve si se quiere recibir los boletines
Returns:
Se devuelve " Checked" si previamente se ha introducido en la sesión el valor del Boletin y este es true. En caso contrario se devuelve la cadena vacía.

---

getPropaganda
`public java.lang.String **getPropaganda**()`

Se devuelve si se quiere recibir la propaganda.
Returns:
Se devuelve " Checked" si previamente se ha introducido en la sesión el valor de la progaganda y esta es true. En caso contrario se devuelve la cadena vacía.

---

getAcepto
`public java.lang.String` **`getAcepto`**`()`
Se devuelve si se aceptan los términos del portal.
Returns:
Se devuelve " Checked" si previamente se ha introducido en la sesión el valor de Acepto y este es true. En caso contrario se devuelve la cadena vacía.

---

setGx
`public void` **`setGx`**`(java.lang.String numero)`
Se introduce un Gusto para ser comparado con losa que tenemos en la sesión del usuario. getGx es el método que nos devuelve si este gusto esta o no selecionado por el usuario.
Parameters:
`numero` - Se corresponde con el numero del gusto, del cual queremos saber si estaba o no selecionado.

---

getGx
`public java.lang.String` **`getGx`**`()`
Para utilizar este método previamente hay que utilizar setGx, dependiendo si el gusto introducido esta insertado en la sesión devolverá " checked" para selecionar esta opción de la página en la que se muestra.
Returns:
Devuelve la cadena vacía si el gusto introducido no se corresponde con alguno de los que se están recordando en la sesión, si estuviese en la sesion devuelve " CHECKED" para indicar que de las opciones este gusto se había selecionado

---

setBusUsuario
`public void` **`setBusUsuario`**`(java.lang.String _eMail)`
Buscamos el usuario introducido por parámetro
Parameters:
`_eMail` - Es el identificador del usuario (eMail)

---

getExisteEMail
`public boolean` **`getExisteEMail`**`(java.lang.String _eMail)`
Para saber si hay algun usuario con este eMail itroducido en el sistema
Parameters:
`_eMail` - Es el identificador del usuario (eMail)
Returns:
True -> Existe False -> No hay ningún usuario con ese eMail en la base de Datos

---

isCorrect
`public boolean` **`isCorrect`**`(java.lang.String _contrasena)`
Primero hay que realizar una busqueda con setBusUsuario posteriormente se coge la contraseña de este usuario y se compara con la introducida.
Parameters:
`_contrasena` - Es la contraseña que queremos ver si se corresponde con la del usuario
Returns:
True -> Son iguales False -> Caso contrario

---

setEliminar
`public boolean` **`setEliminar`**`(java.lang.String _eMail)`
Se elimina al usuario que tenga por eMail el introducido.
Parameters:
`_eMail` - Es el identificador del usuario (eMail).

Returns:
True -> Se ha eliminado False -> Caso contrario

---

setCambioContrasenia

```
public boolean setCambioContrasenia(java.lang.String _eMail,
                                    java.lang.String _password1,
                                    java.lang.String _password2)
```

Se cambia la contraseña del usuario si estas están correctamente hechas y además coincide. Si hubiese algún fallo se almacena.

Parameters:

`_eMail` - Identificador (eMail) del usuario al cual se le quiere cambiar la contraseña

`_password1` - Contraseña nueva

`_password2` - Confirmación de la nueva contraseña.

Returns:
Se devuelve si el cambio de contraseña se ha llevado a cabo (true)

---

setOlvidarDatos

```
public void setOlvidarDatos()
```

Se elimina de la sesión los datos introducidos

---

setRecordarGusto

```
public void setRecordarGusto(java.lang.String _gusto)
```

Con este método se almacena en la sesión un gusto, para recordar que guston ha introducido el usuario para su registro.

Parameters:

`_gusto` - Es el nombre del gusto a recordar, si está o no introducido.

---

setRecordarGusto

```
public void setRecordarGusto(java.lang.Boolean _gusto)
```

Con este método se almacena en la sesión un gusto, para recordar que guston ha introducido el usuario para su registro.

Parameters:

`_gusto` - Se inserta si el gusto está o no selecionado por el usuario (true seleccionado false caso contrario)

---

setOlvidarGustos

```
public void setOlvidarGustos()
```

Se elimina de la sesión todos los gustos

---

setRecordarDatos

```
public void setRecordarDatos(java.lang.String _nombre,
                             java.lang.String _apellidos,
                             java.lang.String _pais,
                             java.lang.String _provincia,
                             java.lang.String _poblacion,
                             java.lang.String _calle,
                             java.lang.String _portal,
                             java.lang.String _piso,
                             java.lang.String _letra,
                             java.lang.String _codigo,
                             java.lang.String _telefono,
                             java.lang.String _sexo,
                             java.lang.String _anio,
                             java.lang.String _mes,
                             java.lang.String _dia,
                             java.lang.String _sector,
                             java.lang.String _profesion,
                             java.lang.String _educacion)
```

Guardamos en la sesión los datos que aquí se pasan como parámetros.

setRecuperarDatos

`public void` **`setRecuperarDatos`**`(java.lang.String _eMail)`

Se almacena en la sesión los datos que tenemos de un usuario determinado que está almacenado en la base de datos.

Parameters:

`_eMail` - Identificador (eMail) del usuario.

desconectar

`public void` **`desconectar`**`()`

Deberá ser invocado antes de eliminar el objeto, ya que sirve para liberar recursos

6.3 Clases de los Controladores

Class eventosAdministrador

```
java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
              |
              +--eventosAdministrador
```

All Implemented Interfaces:

java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

public class **eventosAdministrador**
extends javax.servlet.http.HttpServlet

Servlet que controla los eventos producidos por un Administrador y que controla las acciones a realizar. También se encarga de recuperar los Parametros que nos identifican el evento y las acciones a realizar.

See Also:

Serialized Form

Constructor Summary
`eventosAdministrador``()`

Method Summary	
`void`	**`destroy`**`()` Cuando se destruye el Servlet se deconectaran los objtos creados.
`void`	**`doGet`**`(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response)` Gestiona las peticiones de los Administradores tipo Get
`void`	**`doPost`**`(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response)` Gestiona las peticiones de los Administradores tipo POST
`void`	**`init`**`()` Inicializa los objetos que necesita el servlet

Methods inherited from class javax.servlet.http.HttpServlet
`doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service`

Methods inherited from class javax.servlet.GenericServlet
getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, log, log

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

eventosAdministrador

```
public eventosAdministrador()
```

Method Detail

init

```
public void init()
```

Inicializa los objetos que necesita el servlet

Overrides:

`init` in class `javax.servlet.GenericServlet`

---

doPost

```
public void doPost(javax.servlet.http.HttpServletRequest request,
                   javax.servlet.http.HttpServletResponse response)
            throws javax.servlet.ServletException,
                   java.io.IOException
```

Gestiona las peticiones de los Administradores tipo POST

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

---

doGet

```
public void doGet(javax.servlet.http.HttpServletRequest request,
                  javax.servlet.http.HttpServletResponse response)
           throws javax.servlet.ServletException,
                  java.io.IOException
```

Gestiona las peticiones de los Administradores tipo Get

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

---

destroy

```
public void destroy()
```

Cuando se destruye el Servlet se deconectaran los objtos creados.

Overrides:

`destroy` in class `javax.servlet.GenericServlet`

Class eventosCliente

```
java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
              |
              +--eventosCliente
```

All Implemented Interfaces:

java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

public class **eventosCliente**
extends javax.servlet.http.HttpServlet
Servlet que controla los eventos producidos por un Cliente y que controla las acciones a realizar. También se encarga de recuperar los Parametros que nos identifican el evento y las acciones a realizar.
See Also:
Serialized Form

Constructor Summary
eventosCliente()

Method Summary	
void	**destroy**() Cuando se destruye el Servlet se deconectaran los objtos creados.
void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Gestiona las peticiones tipo GET de los clientes
void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Gestiona las peticiones tipo POST de los clientes
javax.servlet.ServletConfig	**getServletConfig**() Devuelve la configuración del Servlet
void	**init**() Inicializa los objetos que necesita el servlet
void	**init**(javax.servlet.ServletConfig config) Inicializa el Servlet

Methods inherited from class javax.servlet.http.HttpServlet
doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet
getInitParameter, getInitParameterNames, getServletContext, getServletInfo, getServletName, log, log

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

eventosCliente
public **eventosCliente**()

Method Detail

init
```
public final void init(javax.servlet.ServletConfig config)
                throws javax.servlet.ServletException
```
Inicializa el Servlet
Overrides:
`init` in class `javax.servlet.GenericServlet`

---

getServletConfig
```
public final javax.servlet.ServletConfig getServletConfig()
```
Devuelve la configuración del Servlet
Overrides:
`getServletConfig` in class `javax.servlet.GenericServlet`

---

init
```
public void init()
```
Inicializa los objetos que necesita el servlet
Overrides:
`init` in class `javax.servlet.GenericServlet`

---

doPost
```
public void doPost(javax.servlet.http.HttpServletRequest request,
                   javax.servlet.http.HttpServletResponse response)
            throws javax.servlet.ServletException,
                   java.io.IOException
```
Gestiona las peticiones tipo POST de los clientes
Overrides:
`doPost` in class `javax.servlet.http.HttpServlet`

---

doGet
```
public void doGet(javax.servlet.http.HttpServletRequest request,
                  javax.servlet.http.HttpServletResponse response)
           throws javax.servlet.ServletException,
                  java.io.IOException
```
Gestiona las peticiones tipo GET de los clientes
Overrides:
`doGet` in class `javax.servlet.http.HttpServlet`

---

destroy
```
public void destroy()
```
Cuando se destruye el Servlet se deconectaran los objtos creados.
Overrides:
`destroy` in class `javax.servlet.GenericServlet`

Class eventosPortal

```
java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
              |
              +--eventosPortal
```

All Implemented Interfaces:
java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

public class **eventosPortal**
extends javax.servlet.http.HttpServlet

Servlet que controla los eventos producidos por distintos usuarios y que tienen como finalidad la comunicación entre los usuarios También se encarga de recuperar los Parametros que nos identifican el evento y las acciones a realizar.

See Also:

Serialized Form

Constructor Summary
eventosPortal()

Method Summary	
void	**destroy**() Cuando se destruye el Servlet se deconectaran los objtos creados.
void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Gestiona las peticiones tipo GET de los clientes
void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Gestiona las peticiones tipo POST de los clientes
void	**init**() Inicializa los objetos que necesita el servlet

Methods inherited from class javax.servlet.http.HttpServlet
doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet
getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, log, log

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

eventosPortal

```
public eventosPortal()
```

Method Detail

init

```
public void init()
```

Inicializa los objetos que necesita el servlet

Overrides:

init in class javax.servlet.GenericServlet

doPost

```
public void doPost(javax.servlet.http.HttpServletRequest request,
                   javax.servlet.http.HttpServletResponse response)
            throws javax.servlet.ServletException,
                   java.io.IOException
```

Gestiona las peticiones tipo POST de los clientes
Overrides:
`doPost` in class `javax.servlet.http.HttpServlet`

---

doGet

```
public void doGet(javax.servlet.http.HttpServletRequest request,
                  javax.servlet.http.HttpServletResponse response)
           throws javax.servlet.ServletException,
                  java.io.IOException
```

Gestiona las peticiones tipo GET de los clientes
Overrides:
`doGet` in class `javax.servlet.http.HttpServlet`

---

destroy

```
public void destroy()
```

Cuando se destruye el Servlet se deconectaran los objtos creados.
Overrides:
`destroy` in class `javax.servlet.GenericServlet`

Class eventosRevisor

```
java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
              |
              +--eventosRevisor
```

All Implemented Interfaces:
java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

public class **eventosRevisor**
extends javax.servlet.http.HttpServlet
Servlet que controla los eventos producidos por los revisores También se encarga de recuperar los Parametros que nos identifican el evento y realiza las acciones oportunas.
See Also:
Serialized Form

Constructor Summary
eventosRevisor()

Method Summary	
void	**destroy**() Cuando se destruye el Servlet se deconectaran los objtos creados.
void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Gestiona las peticiones tipo GET de los clientes
void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Gestiona las peticiones tipo POST de los clientes
void	**init**() Inicializa los objetos que necesita el servlet

Methods inherited from class javax.servlet.http.HttpServlet
doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet
getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, log, log

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

eventosRevisor

```
public eventosRevisor()
```

Method Detail

init

```
public void init()
```

Inicializa los objetos que necesita el servlet

Overrides:

`init` in class `javax.servlet.GenericServlet`

---

doPost

```
public void doPost(javax.servlet.http.HttpServletRequest request,
                   javax.servlet.http.HttpServletResponse response)
            throws javax.servlet.ServletException,
                   java.io.IOException
```

Gestiona las peticiones tipo POST de los clientes

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

---

doGet

```
public void doGet(javax.servlet.http.HttpServletRequest request,
                  javax.servlet.http.HttpServletResponse response)
           throws javax.servlet.ServletException,
                  java.io.IOException
```

Gestiona las peticiones tipo GET de los clientes

Overrides:

`doGet` in class `javax.servlet.http.HttpServlet`

---

destroy

```
public void destroy()
```

Cuando se destruye el Servlet se deconectaran los objtos creados.

Overrides:

`destroy` in class `javax.servlet.GenericServlet`

Class eventosRevista

```
java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
```

```
                |
                +--eventosRevista
```

All Implemented Interfaces:
java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

public class **eventosRevista**
extends javax.servlet.http.HttpServlet
Servlet que controla los eventos producidos cuando se está visualizando una revista También se encarga de recuperar los Parametros que nos identifican el evento y realiza las acciones oportunas.
See Also:
Serialized Form

Constructor Summary
eventosRevista()

Method Summary	
void	**destroy**() Cuando se destruye el Servlet se deconectaran los objtos creados.
void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Gestiona las peticiones tipo GET de los clientes
void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Gestiona las peticiones tipo POST de los clientes
javax.servlet.ServletConfig	**getServletConfig**() Devuelve la configuración del Servlet
void	**init**() Inicializa los objetos que necesita el servlet
void	**init**(javax.servlet.ServletConfig config) Inicializa el Servlet

Methods inherited from class javax.servlet.http.HttpServlet
doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet
getInitParameter, getInitParameterNames, getServletContext, getServletInfo, getServletName, log, log

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

eventosRevista

```
public eventosRevista()
```

Method Detail

init
```
public final void init(javax.servlet.ServletConfig config)
                throws javax.servlet.ServletException
```
Inicializa el Servlet
Overrides:
`init` in class `javax.servlet.GenericServlet`

---

getServletConfig
```
public final javax.servlet.ServletConfig getServletConfig()
```
Devuelve la configuración del Servlet
Overrides:
`getServletConfig` in class `javax.servlet.GenericServlet`

---

init
```
public void init()
```
Inicializa los objetos que necesita el servlet
Overrides:
`init` in class `javax.servlet.GenericServlet`

---

doPost
```
public void doPost(javax.servlet.http.HttpServletRequest request,
                   javax.servlet.http.HttpServletResponse response)
            throws javax.servlet.ServletException,
                   java.io.IOException
```
Gestiona las peticiones tipo POST de los clientes
Overrides:
`doPost` in class `javax.servlet.http.HttpServlet`

---

doGet
```
public void doGet(javax.servlet.http.HttpServletRequest request,
                  javax.servlet.http.HttpServletResponse response)
           throws javax.servlet.ServletException,
                  java.io.IOException
```
Gestiona las peticiones tipo GET de los clientes
Overrides:
`doGet` in class `javax.servlet.http.HttpServlet`

---

destroy
```
public void destroy()
```
Cuando se destruye el Servlet se deconectaran los objtos creados.
Overrides:
`destroy` in class `javax.servlet.GenericServlet`

Class eventosUsuario

```
java.lang.Object
  |
  +--javax.servlet.GenericServlet
        |
        +--javax.servlet.http.HttpServlet
              |
              +--eventosUsuario
```

All Implemented Interfaces:
java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

---

public class **eventosUsuario**
extends javax.servlet.http.HttpServlet

Servlet que controla los eventos producidos por un Usuario y que controla las acciones comunes a todos los usuarios. También se encarga de recuperar los Parametros que nos identifican el evento y las acciones a realizar.

See Also:

Serialized Form

Constructor Summary
eventosUsuario()

Method Summary	
void	**destroy**() Cuando se destruye el Servlet se deconectaran los objtos creados.
void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Gestiona las peticiones tipo GET de los clientes
void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) Gestiona las peticiones tipo POST de los clientes
void	**init**() Inicializa los objetos que necesita el servlet

Methods inherited from class javax.servlet.http.HttpServlet
doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet
getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, log, log

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

eventosUsuario

```
public eventosUsuario()
```

Method Detail

init

```
public void init()
```

Inicializa los objetos que necesita el servlet

Overrides:

`init` in class `javax.servlet.GenericServlet`

doPost

```
public void doPost(javax.servlet.http.HttpServletRequest request,
                   javax.servlet.http.HttpServletResponse response)
            throws javax.servlet.ServletException,
                   java.io.IOException
```

Gestiona las peticiones tipo POST de los clientes
Overrides:
`doPost` in class `javax.servlet.http.HttpServlet`

---

doGet

```
public void doGet(javax.servlet.http.HttpServletRequest request,
                  javax.servlet.http.HttpServletResponse response)
           throws javax.servlet.ServletException,
                  java.io.IOException
```

Gestiona las peticiones tipo GET de los clientes
Overrides:
`doGet` in class `javax.servlet.http.HttpServlet`

---

destroy

```
public void destroy()
```

Cuando se destruye el Servlet se deconectaran los objtos creados.
Overrides:
`destroy` in class `javax.servlet.GenericServlet`

6.4 Clases de utilería

usuario

Class ErrorBean

```
java.lang.Object
  |
  +--usuario.ErrorBean
```

All Implemented Interfaces:
java.io.Serializable

---

public class **ErrorBean**
extends java.lang.Object
implements java.io.Serializable
Esta clase almancena en la sesión del usuario los errores (cadenas de texto) que se ha producido en un formulario. Con posterioridad podrán ser recuperados para su visualización.
See Also:
Serialized Form

Constructor Summary
ErrorBean()

Method Summary	
java.lang.String	**getError**() Devuelve el último error introducido y lo saca de la sesión.
void	**init**(javax.servlet.http.HttpSession _sesion) Inicializo el objeto con la sesión del usuario.
boolean	**isVacio**() Nos devuelve si hay algún error introducido.
void	**setError**(java.lang.String _cadena) Se introduce un error en la sesión del usuario.

void	**setReset**() Elimina los errores almacenados en la sesión del usuario.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

ErrorBean
`public` **ErrorBean**()

Method Detail

init
`public void` **init**(javax.servlet.http.HttpSession _sesion)
Inicializo el objeto con la sesión del usuario.
Parameters:
`_sesion` - Se corresponde con el objeto que almacena la sesión del usuairo.

---

setReset
`public void` **setReset**()
Elimina los errores almacenados en la sesión del usuario.

---

isVacio
`public boolean` **isVacio**()
Nos devuelve si hay algún error introducido.
Returns:
True -> Está vacío False -> Hay errores introducidos.

---

setError
`public void` **setError**(java.lang.String _cadena)
Se introduce un error en la sesión del usuario.
Parameters:
`_cadena` - Es la cadena de texto que representa al error que se ha producido.

---

getError
`public java.lang.String` **getError**()
Devuelve el último error introducido y lo saca de la sesión.
Returns:
Devuelve la cadena del error insertado, si no hay errores devuelve "No hay Errores"

utiles

Class Cadenas

```
java.lang.Object
  |
  +--utiles.Cadenas
```

---

public class **Cadenas**
extends java.lang.Object
Esta clase saca de un String las palabras que conforman la cadena. Las palabras se identifican por estar separadas de espacios o por comas.

Constructor Summary

Cadenas(java.lang.String _cadena) Para crear un objeto Cadenas con una palabra introducida para ser tratada

Method Summary	
java.lang.String	**getWord**() Nos devuelve la primera palabra contenida y elimina esta plabra de la cadena pasada
void	**ignorarEspaciosYComas**() Ignora los espacios en blanco que hay en el principio de la cadena insertada en el objeto

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Cadenas

public **Cadenas**(java.lang.String _cadena)

Para crear un objeto Cadenas con una palabra introducida para ser tratada

Parameters:

_cadena - Es la cadena con la cual se quiere crear este objeto

Method Detail

ignorarEspaciosYComas

public void **ignorarEspaciosYComas**()

Ignora los espacios en blanco que hay en el principio de la cadena insertada en el objeto

---

getWord

public java.lang.String **getWord**()

Nos devuelve la primera palabra contenida y elimina esta plabra de la cadena pasada

utiles

Class Email

java.lang.Object
  |
  +--**utiles.Email**

---

public class **Email**
extends java.lang.Object
Esta clase da soporte al tratamiento de correos electrónicos.

Constructor Summary
Email()

Method Summary	
static boolean	**correcto**(java.lang.String aux) Comprueba si la cadena pasada tienen la estructura de una dirección de correo electrónico.

static boolean	**send**(java.lang.String para, java.lang.String asunto, java.lang.String texto) Envia un correo electrónico.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Email
```
public Email()
```

Method Detail

correcto
```
public static boolean correcto(java.lang.String aux)
```
Comprueba si la cadena pasada tienen la estructura de una dirección de correo electrónico.
Parameters:
aux - Es la cadena que queremos comprobar si tienen estructura de correo electrónico.
Returns:
True -> Tiene forma de eMail False -> Caso contrario.

---

send
```
public static boolean send(java.lang.String para,
                           java.lang.String asunto,
                           java.lang.String texto)
```
Envia un correo electrónico. La dirección del remitente es pgre@pgre.com.
Parameters:
para - Es la dirección del destinatario del correo electrónico
asunto - Es el asunto del correo electrónico
texto - Es el texto del correo electrónico.
Returns:
True -> Se ha enviado al servidor de correo satisfactoriamente False -> No se ha enviado satisfactoriamente al servidor de correo.

utiles

Class Encriptar

```
java.lang.Object
  |
  +--utiles.Encriptar
```

---

public class **Encriptar**
extends java.lang.Object
Esta clase nos sirve para realizar las encriptaciones

Constructor Summary
Encriptar()

Method Summary	
static java.lang.String	**cadena**(java.lang.String aux) Nos encripta una cadena de texto

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Encriptar
`public Encriptar()`

Method Detail

cadena
`public static java.lang.String cadena(java.lang.String aux)`

Nos encripta una cadena de texto

Parameters:

`aux` - Es la cadena a encriptar

Returns:

La cadena encriptada

utiles

Class Fechas

```
java.lang.Object
  |
  +--utiles.Fechas
```

public class **Fechas**
extends java.lang.Object
Esta clase nos gestiona el tratamiento de fechas.

Constructor Summary	
Fechas()	Constructor por defecto.

Method Summary	
static boolean	**correcta**(int year, int month, int day) No dice si la fecha pasada es correcta o no.
static long	**countDais**(java.sql.Date fecha) Dada un fecha cuenta los dias que tienen desde el 1 - 1- 0000
java.sql.Date	**createDate**() Crea un fecha para bases de datos con la fecha del sistema
static java.sql.Date	**createDate**(java.lang.String anio, java.lang.String mes, java.lang.String dia) Creamos un fecha para base de datos mediante los parámetros pasados.
static long	**daisAfter1900**(java.sql.Date fecha) Cuenta los días que hay entre un fecha y el 1 de enero de 1900
int	**getAnio**() Devolvemos el año de la fecha que contiene el objeto.
int	**getDia**()

	Devuelve el día del mes de la fecha almacenada en el objeto.
int	**getDiaDelAnio**() Devuelve el día del año correspondiente a la fecha almacenada.
int	**getDiaDeLaSemana**() Devuelve el día de la semana en digitos de la fecha almacenada
java.lang.String	**getDiaSemana**() Devolvemos el día de la semana que se corresponde con la fecha albergada en el objeto.
java.lang.String	**getFecha**() Devuelve la fecha albergada en el objeto.
java.lang.String	**getMes**() Devolvemos el nombre del mes que se corresponde con la fecha albergada en el objeto.
int	**getMesInt**() Devolvemos el mes de la fecha albergada en el objeto.
int	**getSemanaDelAnio**() Devuelve el número de la semana del año correspondiente a la fecha almacenada
int	**getSemanaDelMes**() Nos devuelve el número de la semana que se corresponde al mes de la fecha alberfada en el objeto.
static java.lang.String	**NombreMes**(int m) Devolvemos el nombre de un mes pasandole el número que le corresponde

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Fechas
`public Fechas()`
Constructor por defecto. Recoge la fecha del sistema y la almacena en el objeto.

Method Detail

getAnio
`public int getAnio()`
Devolvemos el año de la fecha que contiene el objeto.

---

getMes
`public java.lang.String getMes()`
Devolvemos el nombre del mes que se corresponde con la fecha albergada en el objeto.

---

getDiaSemana
`public java.lang.String getDiaSemana()`
Devolvemos el día de la semana que se corresponde con la fecha albergada en el objeto.
Returns:
Devovemos el nombre del día de la semana: "Lunes" .. "Domingo"

---

getMesInt

```
public int getMesInt()
```
Devolvemos el mes de la fecha albergada en el objeto. Devolvemos el més en forma de dígitos.

---

getFecha
```
public java.lang.String getFecha()
```
Devuelve la fecha albergada en el objeto. P.e. "Lunes, 8 de Agosto de 2002"

---

getDia
```
public int getDia()
```
Devuelve el día del mes de la fecha almacenada en el objeto.

---

getDiaDelAnio
```
public int getDiaDelAnio()
```
Devuelve el día del año correspondiente a la fecha almacenada.

---

getSemanaDelAnio
```
public int getSemanaDelAnio()
```
Devuelve el número de la semana del año correspondiente a la fecha almacenada

---

getSemanaDelMes
```
public int getSemanaDelMes()
```
Nos devuelve el número de la semana que se corresponde al mes de la fecha alberfada en el objeto.

---

getDiaDeLaSemana
```
public int getDiaDeLaSemana()
```
Devuelve el día de la semana en digitos de la fecha almacenada

---

correcta
```
public static final boolean correcta(int year,
                                     int month,
                                     int day)
```
No dice si la fecha pasada es correcta o no.
Parameters:
`year` - Año de la fecha que se va a comprobar
`month` - Mes de la fecha que se va a comprobar (1 -> Enero, 12 Diciembre)
`day` - El día del mes de la fecha que se va a comprobar.
Returns:
true -> Es correcta false -> No es correcta

NombreMes
```
public static final java.lang.String NombreMes(int m)
```
Devolvemos el nombre de un mes pasandole el número que le corresponde
Parameters:
`m` - El mes Enero -> 1 .. Diciembre -> 12.
Returns:
El nombre del mes

---

createDate
```
public static final java.sql.Date createDate(java.lang.String anio,
                                              java.lang.String mes,
                                              java.lang.String dia)
```
Creamos un fecha para base de datos mediante los parámetros pasados.
Parameters:
`anio` - Año de la fecha
`mes` - Mes de la fecha, deben ser cadenas que se puedan convertir a enteros.
`dia` - Dia del mes de la fecha.
Returns:

Devuelve un tipo Date con la fecha pasada.

---

createDate

```
public java.sql.Date createDate()
```

Crea un fecha para bases de datos con la fecha del sistema

Returns:

Devuelve un objeto Date con la fecha actual del sistema.

---

countDais

```
public static final long countDais(java.sql.Date fecha)
```

Dada un fecha cuenta los dias que tienen desde el 1 - 1- 0000

---

daisAfter1900

```
public static final long daisAfter1900(java.sql.Date fecha)
```

Cuenta los días que hay entre un fecha y el 1 de enero de 1900