Anexo IV. Documentación del código

1. Introducción

En este capítulo se recoge parte de la documentación *javadoc* [37] generada a partir de los comentarios del código de las clases e interfaces Java desarrolladas en el proyecto, dividido en sus correspondientes paquetes.

1.1 Herramienta "DocCheck"

Antes de ejecutar la herramienta para generar los comentarios se ha utilizado una herramienta desarrollada en Sun, llamada "Sun Doc Check Doclet" [36], o "DocCheck", para comprobar los comentarios *javadoc*. Se ejecuta sobre el código fuente y genera un informe describiendo qué errores de etiquetas y de estilo tienen los comentarios y recomienda los cambios a realizar. "DocCheck" es un "doclet" para *javadoc*, o una extensión o "plug-in".Se ha utilizado la versión 1.2 Beta 2. La herramienta se ha ejecutado con la siguiente llamada en la línea de comandos:

```
C:\Documents and Settings\Yo\workspace>javadoc @JavadocDocCheck
@paquetes
```

Se han utilizado dos ficheros con la configuración. El primero, `JavadocDocCheck`, contiene las opciones de configuración generales para el "DocCheck", y se lista a continuación:

```
-doclet com.sun.tools.doclets.doccheck.DocCheck
-docletpath doccheck1.2b2\doccheck.jar
-sourcepath QTI\WEB-INF\src
-d "DocCheck"
```

Simplemente establece la localización de los archivos fuente de la extensión y de la clase que ejecuta, del código fuente de las clases a comprobar y el directorio de salida en el que guardar el informe.

El segundo archivo indica los paquetes que comprobar, que son todos los paquetes de la Aplicación Web:

```
control
utilidades
utilidades.etiquetas.util
utilidades.etiquetas.util.logging
xml.items
xml.test
```

La salida generada por la herramienta son una serie de páginas Web, informando de los errores encontrados, y algunas estadísticas sobre esos errores.

1.2 Conclusiones sobre "DocCheck"

La herramienta "DocCheck" es una útil herramienta que permite de una forma rápida comprobar las reglas y el estilo de los comentarios para *javadoc* de los paquetes Java de una forma fácil y rápida. A pesar de ello, quizás por ser una versión "beta", ha presentado un comportamiento extraño con el código de este proyecto: detecta la falta de muchas etiquetas `@throws` en las clases, a pesar de que están presentes. No obstante, con una configuración similar a la que hay que utilizar con *javadoc* para generar la documentación, se puede conseguir depurar esos comentarios para ajustarlos lo más posible a las recomendaciones dadas por Sun.

2. Documentación generada con la herramienta *javadoc*

Para generar la documentación se ha realizado la siguiente llamada a la herramienta *javadoc*:

```
C:\Documents and Settings\Yo\workspace>javadoc @JavadocOpciones
@paquetes
```

La configuración se ha incluido en dos archivos. El archivo `paquetes` es el mismo que el utilizado y listado más arriba, conteniendo los paquetes a comprobar. El archivo `JavadocOpciones` contiene las opciones para la herramienta, y se lista a continuación:

```
-d "Documentación Javadoc"
-sourcepath QTI\WEB-INF\src
-overview QTI\WEB-INF\src\overview.html
-private
-author
-version
-doctitle  "Aplicación Web <i>Herramienta de Creaci&oacute;n de Examen
QTI</i>"
-bottom "Herramienta de Creaci&oacute;n de Examen QTI. Universidad de
Sevilla."
-charset "iso-8859-1"
```

La configuración contiene elementos para especificar el directorio en el que generar los archivos, la localización del código sobre el que ejecutar la herramienta, la localización del archivo de descripción para la página inicial, una opción para que incluya también las variables y métodos privados, otra para que incluya el autor y la versión, el título de la página principal, el pie de las páginas generadas y el juego de caracteres HTML a utilizar en las páginas.

A continuación se lista la documentación generada con estas opciones. Se van a obviar las descripciones detalladas de campos, métodos, clases, paquetes, y la lista de los métodos heredados de la clase `Object` por razones de espacio. Para ver la documentación completa consultarla en el CD incluido con el proyecto.

2.1 Página principal

Aplicación Web *Herramienta de Creación de Examen QTI*

Clases Java de la Aplicación Web *Herramienta de Creación de Examen QTI*.

See:
Description

Packages	
control	Proporciona las clases y las interfaces para decidir y ejecutar la lógica de negocio asociada a cada JSP de la Aplicación Web.
utilidades	Proporciona clases para contener tipos de datos, clases resultado de las comprobaciones y clases con método estáticos para ayudar a realizar determinadas acciones repetitivas en la Aplicación Web.
utilidades.etiquetas.util	Contiene dos clases que implementan dos etiquetas personalizadas de la biblioteca `util`.
utilidades.etiquetas.util.logging	Proporciona todas las clases necesarias para el correcto funcionamiento de la etiqueta definida con el nombre de `logger`, encargada del registro de los errores que se producen en la Aplicación Web.
xml.items	Proporciona las clases necesarias para almacenar la información relativa a todos los tipos de ítems y escribirlos en disco.
xml.test	Proporciona todas las clases que definen un test, correspondiéndose con la estructura de clases que define la norma QTI para un test.

Clases Java de la Aplicación Web *Herramienta de Creación de Examen QTI*. Estos paquetes forman la lógica de negocio y de control de la Aplicación Web.

Version:
1.0
Author:
David Domínguez

Herramienta de Creación de Examen QTI. Universidad de Sevilla.

A partir de este punto se va a obviar la tabla del encabezamiento y la tabla y la frase del del pie de página, por ser la misma en todas las páginas Web de la documentación generada.

2.2 Paquete `control`

Package control

Proporciona las clases y las interfaces para decidir y ejecutar la lógica de negocio asociada a cada JSP de la Aplicación Web.

See:

Description

Interface Summary	
Control	Interfaz de todos los componentes de control implícitos encargados de manejar la lógica asociada a cada una de las páginas JSP.

Class Summary	
Choice	Componente de control implícito asociado a la página JSP "choice.jsp".
ControlFilter	Filtro que se encarga de decidir qué componente de control se debe ejecutar en función de la página JSP que esté siendo visitada.
Creartest	Componente de control implícito asociado a la página JSP "creartest.jsp".
Explorador	Componente de control implícito asociado a la página JSP "explorador.jsp".
Fileupload	Componente de control implícito asociado a la página JSP "fileupload.jsp".
Fin	Componente de control implícito asociado a la página JSP "fin.jsp".
Finitem	Componente de control implícito asociado a la página JSP "finitem.jsp".
Fintest	Componente de control implícito asociado a la página JSP "fintest.jsp".
Gapmatch	Componente de control implícito asociado a la página JSP "gapmatch.jsp".
Hottext	Componente de control implícito asociado a la página JSP

	"hottext.jsp".
Index	Componente de control implícito asociado a la página JSP "index.jsp".
Inlinechoice	Componente de control implícito asociado a la página JSP "inlinechoice.jsp".
Introducirpesos	Componente de control implícito asociado a la página JSP "introducirpesos.jsp".
Match	Componente de control implícito asociado a la página JSP "match.jsp".
Nuevodirectorio	Componente de control implícito asociado a la página JSP "nuevodirectorio.jsp".
Respuestaschoice	Componente de control implícito asociado a la página JSP "respuestaschoice.jsp".
Respuestasgapmatch	Componente de control implícito asociado a la página JSP "respuestasgapmatch.jsp".
Respuestashottext	Componente de control implícito asociado a la página JSP "respuestashottext.jsp".
Respuestasinlinechoice	Componente de control implícito asociado a la página JSP "respuestasinlinechoice.jsp".
Respuestasmatch	Componente de control implícito asociado a la página JSP "respuestasmatch.jsp".
Respuestastextentry	Componente de control implícito asociado a la página JSP "respuestastextentry.jsp".
Seleccionaritems	Componente de control implícito asociado a la página JSP "seleccionaritems.jsp".
Seleccionasignatura	Componente de control implícito asociado a la página JSP "seleccionasignatura.jsp".
Textentry	Componente de control implícito asociado a la página JSP "textentry.jsp".

Interfaz `control`

control

Interface Control

All Known Implementing Classes:

Choice, Creartest, Explorador, Fileupload, Fin, Finitem, Fintest, Gapmatch, Hottext, Index, Inlinechoice, Introducirpesos, Match, Nuevodirectorio, Respuestaschoice, Respuestasgapmatch, Respuestashottext, Respuestasinlinechoice, Respuestasmatch, Respuestastextentry, Seleccionaritems, Seleccionasignatura, Textentry

---

`interface` **`Control`**

Interfaz de todos los componentes de control implícitos encargados de manejar la lógica asociada a cada una de las páginas JSP. Su único método es el que ejecuta toda la lógica. Todas las clases encargadas de manejar la lógica de las páginas JSP deben implementar esta clase y su método para que pueda realizarse correctamente.

Author:
David Domínguez

Method Summary	
boolean	**doLogic**(HttpServletRequest request, HttpServletResponse response) Realiza la lógica de control asociada a las JSPs.

Clase `Choice`

control
Class Choice

```
java.lang.Object
  └ control.Choice
```

All Implemented Interfaces:
Control

```
public class Choice
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "choice.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros iniciales de la toma de datos de una pregunta tipo "Choice" sean correctos, en cuyo caso redirecciona la respuesta a la siguiente página de toma de datos de tipo "Choice". La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el único que contiene, y que es el encargado de realizar la lógica asociada a la página "choice.jsp".

Author:
David Domínguez
See Also:
`Control`

Constructor Summary	
Choice()	

Method Summary	
boolean	**doLogic**(HttpServletRequest request, HttpServletResponse response)

	Realiza la comprobación de los parámetros que se recogen en la página "choice.jsp".

Clase `ControlFilter`

control

Class ControlFilter

```
java.lang.Object
  └control.ControlFilter
```

```
public class ControlFilter
extends java.lang.Object
```

Filtro que se encarga de decidir qué componente de control se debe ejecutar en función de la página JSP que esté siendo visitada. La clase implementa la interfaz `Filter`, y como tal, se ejecutará antes de que la petición del cliente llegue al Servlet destino (real o JSP compilado a Servlet). La clase implementa los tres métodos que tiene la interfaz `Filter`. El método `doFilter` que se ejecuta con cada petición a cualquier página del proyecto como está configurado en "web.xml", es el método de control de la Aplicación Web, encargado de decidir qué clase contiene la lógica asociada a una página JSP determinada.

Author:
David Domínguez

See Also:
`javax.servlet.Filter`

Field Summary

protected FilterConfig	**config** Objeto de configuración del Filtro.

Constructor Summary

ControlFilter()

Method Summary

void	**destroy**() Se llama cuando el filtro se descarga de la memoria, típicamente cuando la Aplicación se descarga.
void	**doFilter**(ServletRequest req, ServletResponse res, FilterChain chain) Método centro de control de la Aplicación, que se encarga de decidir qué componente está asociado a cada página JSP visitada (si lo tiene).
void	**init**(FilterConfig filterConfig) Método que arranca y configura el filtro.

Clase `Creartest`

control
Class Creartest

`java.lang.Object`
  └ **`control.Creartest`**

All Implemented Interfaces:
Control

```
public class Creartest
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "creartest.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros iniciales de la toma de datos para crear un nuevo test sean correctos, en cuyo caso redirecciona la respuesta a la página de seleccionar los ítems a incluir en el test. La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el único que contiene, y que es el encargado de realizar la lógica asociada a la página "creartest.jsp".

Author:
David Domínguez
See Also:
`Control`

Constructor Summary

`Creartest``()`

Method Summary

`boolean`	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Realiza la comprobación de los parámetros que se recogen en la página "creartest.jsp".

Clase `Explorador`

control
Class Explorador

`java.lang.Object`
  └ **`control.Explorador`**

All Implemented Interfaces:
Control

```
public class Explorador
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "explorador.jsp". Es la lógica asociada a esa página JSP. Su método `doLogic` se ejecutará siempre que sea llamada la página. La página es el frame de la mitad izquierda de la página "buscadorimagenes.jsp", y es el explorador de imágenes y directorios para seleccionar una imagen. Se encarga de la navegación por los directorios de imágenes, mostrando los subdirectorios y las imágenes que se encuentran en el directorio actual, permitiendo entrar en subdirectorios y volver atrás, previsualizando a su vez las imágenes en el frame derecho de la página para elegir una.

Author:
David Domínguez
See Also:
`Control`

Constructor Summary

`Explorador``()`

Method Summary

`boolean`	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Lógica control asociada a la página "explorador.jsp".

Clase `Fileupload`

control
Class Fileupload

```
java.lang.Object
  └ control.Fileupload
```

All Implemented Interfaces:
Control

```
public class Fileupload
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "fileupload.jsp". Es la lógica asociada a esa página JSP. Su método `doLogic` se ejecutará siempre que sea llamada la página. Su función es la de permitir al usuario seleccionar un directorio en el servidor al que subir las imágenes seleccionadas en la página.

Author:
David Domínguez
See Also:
`Control`

Constructor Summary

Fileupload()

Method Summary

boolean	doLogic(HttpServletRequest request, HttpServletResponse response) Lógica de control asociada a la página "fileupload.jsp", que se encarga de subir un archivo al servidor.

Clase `Fin`

control

Class Fin

```
java.lang.Object
  └control.Fin
```

All Implemented Interfaces:

Control

```
public class Fin
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "fin.jsp". Es la lógica asociada a esa página JSP. Su método `doLogic` se ejecutará siempre que sea llamada la página. Sólo se encarga de disponer en la página los mensajes de éxito o fracaso creando el ítem o el test XML del QTI.

Author:

David Domínguez

See Also:

Control

Constructor Summary

Fin()

Method Summary

boolean	doLogic(HttpServletRequest request, HttpServletResponse response) Será llamado por el filtro "ControlFilter" en el caso de que detecte que la página requerida es "fin.jsp".

Clase `Finitem`

control
Class Finitem
`java.lang.Object`
  `└` **`control.Finitem`**
All Implemented Interfaces:
Control

```
public class Finitem
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "finitem.jsp". Es la lógica asociada a esa página JSP. Su método `doLogic` se ejecutará siempre que sea llamada la página. Permite al usuario elegir el directorio dentro del de los ítems donde almacenar el ítem XML y lo intenta crear y almacenar en disco.

Author:
David Domínguez
See Also:
`Control`

Constructor Summary

`Finitem``()`

Method Summary

boolean	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Será llamado por el filtro `ControlFilter` en el caso de que detecte que la página requerida es "fin.jsp".

Clase `Fintest`

control
Class Fintest
`java.lang.Object`
  `└` **`control.Fintest`**
All Implemented Interfaces:
Control

```
public class Fintest
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "fintest.jsp". Es la lógica asociada a esa página JSP. Su método `doLogic` se ejecutará siempre que sea llamada la

página. Permite al usuario elegir el directorio dentro del de test donde almacenar el test XML y lo intenta crear y almacenar en disco.

Author:
David Domínguez
See Also:
`Control`

Constructor Summary

`Fintest``()`

Method Summary

boolean	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Será llamado por el filtro `ControlFilter` en el caso de que detecte que la página requerida es "fintest.jsp".

Clase `Gapmatch`

control
Class Gapmatch

`java.lang.Object`
`└ control.Gapmatch`
All Implemented Interfaces:
Control

```
public class Gapmatch
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "gapmatch.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros iniciales de la toma de datos de una pregunta tipo "Gap Match" sean correctos, en cuyo caso redirecciona la respuesta a la siguiente página de toma de datos de tipo "Gap Match". La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el único que contiene, y que es el encargado de realizar la lógica asociada a la página "gapmatch.jsp".

Author:
David Domínguez
See Also:
`Control`

Constructor Summary

Gapmatch()

Method Summary

boolean	doLogic(HttpServletRequest request, HttpServletResponse response) Realiza la comprobación de los parámetros que se recogen en la página "gapmatch.jsp".

Clase Hottext

control

Class Hottext

java.lang.Object
  └ **control.Hottext**

All Implemented Interfaces:

Control

```
public class Hottext
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "hottext.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método doLogic siempre que sea llamada la página. Su función es comprobar que todos los parámetros iniciales de la toma de datos de una pregunta tipo "Hot Text" sean correctos, en cuyo caso redirecciona la respuesta a la siguiente página de toma de datos de tipo "Hot Text". La clase implementa la interfaz Control, implementando su método doLogic, que es el único que contiene, y que es el encargado de realizar la lógica asociada a la página "hottext.jsp".

Author:

David Domínguez

See Also:

Control

Constructor Summary

Hottext()

Method Summary

boolean	doLogic(HttpServletRequest request, HttpServletResponse response) Realiza la comprobación de los parámetros que se recogen en la página "hottext.jsp".

Clase `Index`

control
Class Index
`java.lang.Object`
  `└` **`control.Index`**
All Implemented Interfaces:
Control

```
public class Index
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "index.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Sólo se preocupa de inicializar la sesión del usuario con una nueva para borrar cualquier objeto antiguo almacenado en sesión. La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el único que contiene, y que es el encargado de realizar la lógica asociada a la página "index.jsp".

Author:
David Domínguez
See Also:
`Control`

Constructor Summary

`Index``()`

Method Summary

boolean	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Reinicia el objeto sesión para eliminar cualquier objeto a desechar almacenado ahí.

Clase `Inlinechoice`

control
Class Inlinechoice
`java.lang.Object`
  `└` **`control.Inlinechoice`**
All Implemented Interfaces:
Control

```
public class Inlinechoice
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "inlinechoice.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros iniciales de la toma de datos de una pregunta tipo "Inline Choice" sean correctos, en cuyo caso redirecciona la respuesta a la siguiente página de toma de datos de tipo "Inline Choice". La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el único que contiene, y que es el encargado de realizar la lógica asociada a la página "inlinechoice.jsp".

Author:
David Domínguez
See Also:
`Control`

Constructor Summary

`Inlinechoice``()`

Method Summary

`boolean`	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Realiza la comprobación de los parámetros que se recogen en la página "inlinechoice.jsp".

Clase `Introducirpesos`

control
Class Introducirpesos

```
java.lang.Object
  └ control.Introducirpesos
```

All Implemented Interfaces:
Control

```
public class Introducirpesos
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "introducirpesos.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que los datos introducidos en la página de introducción de los pesos de los ítems del test sean correctos. La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el encargado de realizar la lógica asociada a la página "introducirpesos.jsp". Contiene también un método para leer la tabla de pesos de la página y otro método para rellenar los pesos que se dejen vacíos.

Author:
David Domínguez
See Also:

Control

Constructor Summary

Introducirpesos()

Method Summary

boolean	**doLogic**(HttpServletRequest request, HttpServletResponse response) Realiza la comprobación de los parámetros que se recogen en la página "introducirpesos.jsp".
void	**leePesos**(HttpServletRequest request, java.lang.String origen, java.lang.Double[] pesos, EstadoWeb estado, java.lang.String tipoPeso) Lee los pesos introducidos en la página de introducción de pesos del objeto `HttpServletRequest`, con el nombre común de los campos dado por el `String origen` y rellena el array de `Double` pesos, comprobando que sean números decimales correctos, y actualizando el valor del objeto `estado` en caso de error.
void	**rellenaPesos**(java.lang.Double[] pesos, double relleno) Rellena las posiciones vacías del array de `Double pesos` con el valor indicado por el parámetro `relleno`.

Clase `Match`

control

Class Match

`java.lang.Object`
  └ **`control.Match`**

All Implemented Interfaces:
Control

```
public class Match
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "match.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros iniciales de la toma de datos de una pregunta tipo "Match" sean correctos, en cuyo caso redirecciona la respuesta a la siguiente página de toma de datos de tipo "Match". La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el único que contiene, y que es el encargado de realizar la lógica asociada a la página "match.jsp".

Author:
David Domínguez
See Also:

Control

Constructor Summary

Match()

Method Summary

boolean	**doLogic**(HttpServletRequest request, HttpServletResponse response) Realiza la comprobación de los parámetros que se recogen en la página "match.jsp".

Clase Nuevodirectorio

control

Class Nuevodirectorio

java.lang.Object
  └ **control.Nuevodirectorio**

All Implemented Interfaces:

Control

```
public class Nuevodirectorio
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "nuevodirectorio.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método doLogic siempre que sea llamada la página. Se encarga de crear un nuevo directorio.

Author:

David Domínguez

See Also:

Control

Constructor Summary

Nuevodirectorio()

Method Summary

boolean	**doLogic**(HttpServletRequest request, HttpServletResponse response) Se encarga de crear un nuevo subdirectorio con el nombre indicado por DirNombre en el directorio DirectorioActual, parámetros ambos que recibe en el request.

Clase `Respuestaschoice`

control

Class Respuestaschoice

`java.lang.Object`
  `└` **`control.Respuestaschoice`**

All Implemented Interfaces:

Control

```
public class Respuestaschoice
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "respuestaschoice.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros finales de la toma de datos de una pregunta tipo "Choice" sean correctos, en cuyo caso redirecciona la respuesta a la página "seleccionasignatura.jsp" para elegir donde guardar el ítem. La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el encargado de realizar la lógica asociada a la página "respuestaschoice.jsp". Tiene otro método que simplemente se ejecuta si los parámetros son correctos para redireccionar la respuesta y terminar de crear el objeto `ChoiceXML`.

Author:

David Domínguez

See Also:

`Control`

Constructor Summary

`Respuestaschoice``()`

Method Summary

`boolean`	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Realiza la comprobación de los parámetros que se recogen en la página "respuestaschoice.jsp".
`private void`	**`terminaChoice`**`(HttpServletRequest req, HttpServletResponse res, java.lang.String[] respuestas, Identificador[] identif, EnteroPositivo max, boolean[] correct, boolean[] fija)` Este método se ejecuta una vez que se comprueba que todos los parámetros son correctos.

Clase `Respuestasgapmatch`

control

Class Respuestasgapmatch

`java.lang.Object`
  `└` **`control.Respuestasgapmatch`**

All Implemented Interfaces:

Control

```
public class Respuestasgapmatch
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "respuestasgapmatch.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros finales de la toma de datos de una pregunta tipo "Gap Match" sean correctos, en cuyo caso redirecciona la respuesta a la página "seleccionasignatura.jsp" para elegir donde guardar el ítem. La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el encargado de realizar la lógica asociada a la página "respuestaschoice.jsp". Tiene otro método que simplemente se ejecuta si los parámetros son correctos para redireccionar la respuesta y terminar de crear el objeto `GapmatchXML`.

Author:

David Domínguez

See Also:

`Control`

Constructor Summary

`Respuestasgapmatch``()`	

Method Summary

`boolean`	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Realiza la comprobación de los parámetros que se recogen en la página "respuestasgapmatch.jsp".
`private void`	**`terminaGapmatch`**`(HttpServletRequest req, HttpServletResponse res, java.lang.String[] textos, java.lang.String[] opc, Identificador[] iHuec, Identificador[] iOpc, EnteroPositivo[] maxOpc, java.lang.String[] correct, boolean[] fija)` Este método se ejecuta una vez que se comprueba que todos los parámetros son correctos.

Clase `Respuestashottext`

control

Class Respuestashottext

`java.lang.Object`
  `└` **`control.Respuestashottext`**

All Implemented Interfaces:

Control

```
public class Respuestashottext
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "respuestashottext.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros finales de la toma de datos de una pregunta tipo "Hot Text" sean correctos, en cuyo caso redirecciona la respuesta a la página "seleccionasignatura.jsp" para elegir donde guardar el ítem. La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el encargado de realizar la lógica asociada a la página "respuestashottext.jsp". Tiene otro método que simplemente se ejecuta si los parámetros son correctos para redireccionar la respuesta y terminar de crear el objeto `HottextXML`.

Author:

David Domínguez

See Also:

`Control`

Constructor Summary

`Respuestashottext``()`

Method Summary

`boolean`	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Realiza la comprobación de los parámetros que se recogen en la página "respuestashottext.jsp".
`private void`	**`terminaGapmatch`**`(HttpServletRequest req, HttpServletResponse res, java.lang.String[] textos, java.lang.String[] hott, Identificador[] iHott, EnteroPositivo max, boolean[] correct)` Este método se ejecuta una vez que se comprueba que todos los parámetros son correctos.

Clase `Respuestasinlinechoice`

control

Class Respuestasinlinechoice

```
java.lang.Object
  └ control.Respuestasinlinechoice
```

All Implemented Interfaces:

Control

```
public class Respuestasinlinechoice
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "respuestasinlinechoice.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros finales de la toma de datos de una pregunta tipo "Inline Choice" sean correctos, en cuyo caso redirecciona la respuesta a la página "seleccionasignatura.jsp" para elegir donde guardar el ítem. La clase implementa la interfaz `Control`, implementando su método `doLogic` , que es el encargado de realizar la lógica asociada a la página "respuestasinlinechoice.jsp".Tiene otro método que simplemente se ejecuta si los parámetros son correctos para redireccionar la respuesta y terminar de crear el objeto `InlinechoiceXML`.

Author:

David Domínguez

See Also:

`Control`

Constructor Summary

`Respuestasinlinechoice``()`

Method Summary

`boolean`	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Realiza la comprobación de los parámetros que se recogen en la página "respuestasinlinechoice.jsp".
`private void`	**`terminaInlinechoice`**`(HttpServletRequest req, HttpServletResponse res, java.lang.String[] respuestas, Identificador[] identif, java.lang.String correct, boolean[] fija, java.lang.String texto1, java.lang.String texto2)` Este método se ejecuta una vez que se comprueba que todos los parámetros son correctos.

Clase `Respuestasmatch`

control

Class Respuestasmatch

`java.lang.Object`
  └ **`control.Respuestasmatch`**

All Implemented Interfaces:
Control

```
public class Respuestasmatch
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "respuestasmatch.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros finales de la toma de datos de una pregunta tipo "Match" sean correctos, en cuyo caso redirecciona la respuesta a la página "seleccionasignatura.jsp" para elegir donde guardar el ítem. La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el encargado de realizar la lógica asociada a la página "respuestasmatch.jsp". Tiene otro método que simplemente se ejecuta si los parámetros son correctos para redireccionar la respuesta y terminar de crear el objeto `MatchXML`.

Author:
David Domínguez
See Also:
`Control`

Constructor Summary

`Respuestasmatch``()`	

Method Summary

`boolean`	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Realiza la comprobación de los parámetros que se recogen en la página "respuestasmatch.jsp".
`private void`	**`terminaMatch`**`(HttpServletRequest req, HttpServletResponse res, java.lang.String[] resp1, java.lang.String[] resp2, Identificador[] identif1, Identificador[] identif2, EnteroPositivo max, EnteroPositivo[] max1, EnteroPositivo[] max2, boolean[][] correct, boolean[] fija1, boolean[] fija2)` Este método se ejecuta una vez que se comprueba que todos los parámetros son correctos.

Clase `Respuestastextentry`

control

Class Respuestastextentry

`java.lang.Object`
  `└` **`control.Respuestastextentry`**

All Implemented Interfaces:

Control

```
public class Respuestastextentry
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "respuestastextentry.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros finales de la toma de datos de una pregunta tipo "Text Entry" sean correctos, en cuyo caso redirecciona la respuesta a la página "seleccionasignatura.jsp" para elegir donde guardar el ítem. La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el encargado de realizar la lógica asociada a la página "respuestastextentry.jsp". Tiene otro método que simplemente se ejecuta si los parámetros son correctos para redireccionar la respuesta y terminar de crear el objeto `TextentryXML`.

Author:

David Domínguez

See Also:

`Control`

Constructor Summary

`Respuestastextentry``()`

Method Summary

`boolean`	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Realiza la comprobación de los parámetros que se recogen en la página "respuestashottext.jsp".
`private void`	**`terminaTextentry`**`(HttpServletRequest req, HttpServletResponse res, java.lang.String respuesta, EnteroPositivo longitud, java.lang.String texto1, java.lang.String texto2)` Este método se ejecuta una vez que se comprueba que todos los parámetros son correctos.

Clase `Seleccionaritems`

control

Class Seleccionaritems

```
java.lang.Object
  └ control.Seleccionaritems
```

All Implemented Interfaces:

Control

```
public class Seleccionaritems
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "seleccionaritems.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es mostrar al usuario los ítems a seleccionar de los directorios de las asignaturas y permitirle elegir los que desea añadir al test, que se mostrarán en la capa de la derecha de la misma página. La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el encargado de realizar la lógica asociada a la página "seleccionaritems.jsp". Contiene, además, otros métodos para ayudar en las tareas de añadir, eliminar y marcar a los ítems ya seleccionados.

Author:

David Domínguez

See Also:

`Control`

Constructor Summary

`Seleccionaritems``()`

Method Summary

`private void`	**`añadeItem`**`(HttpServletRequest request)` Añade los ítems que se han seleccionado en la página marcando sus checkboxes, a la lista de ítems a añadir que se encuentra en sesión.
`boolean`	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Muestra el directorio de los ítems de una asignatura permitiendo navegar por los subdirectorios y elegir los ítems a incluir en el test, que se mostrarán en otra capa de la misma página.
`private void`	**`eliminaItem`**`(HttpServletRequest request)` Elimina los ítems que se han seleccionado en la página marcando sus checkboxes, de la lista de ítems a añadir que se encuentra en sesión.
`private boolean[]`	**`marcaItemsSeleccionados`**`(java.util.Vector itemsDisponibles, HttpServletRequest request)` Rellena un array de `boolean` indicando con true la posición de los elementos del vector de ítems disponibles para seleccionar que ya se han seleccionado.

Clase `Seleccionasignatura`

control
Class Seleccionasignatura

```
java.lang.Object
  └ control.Seleccionasignatura
```

All Implemented Interfaces:
Control

```
public class Seleccionasignatura
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "seleccionasignatura.jsp". Es la lógica asociada a esa página JSP. Su método `doLogic` se ejecutará siempre que sea llamada la página. Se encarga de pedir al usuario una asignatura de trabajo.

Author:
David Domínguez
See Also:
`Control`

Constructor Summary

Seleccionasignatura()

Method Summary

boolean	**doLogic**(HttpServletRequest request, HttpServletResponse response) Será llamado por el filtro `ControlFilter`" en el caso de que detecte que la página requerida es "seleccionasignatura.jsp".

Clase `Textentry`

control
Class Textentry

```
java.lang.Object
  └ control.Textentry
```

All Implemented Interfaces:
Control

```
public class Textentry
extends java.lang.Object
implements Control
```

Componente de control implícito asociado a la página JSP "textentry.jsp". Es la lógica asociada a esa página JSP. Se ejecutará su método `doLogic` siempre que sea llamada la página. Su función es comprobar que todos los parámetros iniciales de la toma de datos

de una pregunta tipo "Text Entry" sean correctos, en cuyo caso redirecciona la respuesta a la siguiente página de toma de datos de tipo "Text Entry". La clase implementa la interfaz `Control`, implementando su método `doLogic`, que es el único que contiene, y que es el encargado de realizar la lógica asociada a la página "textentry.jsp".

Author:
David Domínguez
See Also:
`Control`

Constructor Summary	
`Textentry``()`	

Method Summary	
`boolean`	**`doLogic`**`(HttpServletRequest request, HttpServletResponse response)` Realiza la comprobación de los parámetros que se recogen en la página "textentry.jsp".

2.3 Paquete `utilidades`

Package utilidades

Proporciona clases para contener tipos de datos, clases resultado de las comprobaciones y clases con método estáticos para ayudar a realizar determinadas acciones repetitivas en la Aplicación Web.

See:
Description

Class Summary	
Comprobaciones	Clase con diversos métodos estáticos para leer los datos introducidos en las páginas de tomas de datos de los ítems y el test y comprobar, en aquellos que haga falta, que sean correctos.
EnteroPositivo	Clase que contiene la representación de un número entero positivo.
EstadoWeb	Clase que representa el estado de comprobación de una Web.
Identificador	Clase que almacena la representación de un tipo *identifier* válido según la norma QTI del IMS.
ListaArchivos	Clase que contiene métodos estáticos que devuelven determinadas listas del contenido de los directorios.
MensajeEstado	Clase que contiene un estado y un mensaje.
ParserNeutro	Clase que contiene métodos estáticos para escribir con formato XML cadenas que ya tienen algún formato XHTML y comprobar que sean

	cadenas XML válidas.
ResumenItem	Clase que representa un resumen de los ítems (archivos XML independientes que representan al ítem).
UrlUtils	Clase con métodos para codificar URLs.

Clase `Comprobaciones`

utilidades

Class Comprobaciones

```
java.lang.Object
  └utilidades.Comprobaciones
```

```
public class Comprobaciones
extends java.lang.Object
```

Clase con diversos métodos estáticos para leer los datos introducidos en las páginas de tomas de datos de los ítems y el test y comprobar, en aquellos que haga falta, que sean correctos.

Author:
David Domínguez

Constructor Summary

Comprobaciones()	

Method Summary

static void	**leeCheckBoxes**(boolean[] respFija, java.lang.String origen, HttpServletRequest request) Método para ayudar a leer un conjunto de checkboxes con un nombre base común de una página Web.
static int	**sumaCorrectas**(boolean[] respCorrecta, java.lang.String origen, HttpServletRequest request) Método para ayudar a leer un conjunto de checkboxes con un nombre base común de una página Web informando del número que están marcadas.
static void	**testMaximo**(EnteroPositivo maxChoices, EstadoWeb estado, java.lang.String campo, int numCorrectas, int numMaxOpciones) Comprueba que el número máximo de elecciones introducido sea correcto en preguntas con un solo grupo de respuestas y entre las que se puede seleccionar más de una como correcta.
static void	**testTablaMaximos**(EnteroPositivo[] maxChoices, java.lang.String campo, EstadoWeb estado, java.lang.String[] max, java.lang.String grupo, java.lang.String elemento, int[] numCorrectas,

	int numOpciones) Comprueba los números máximos de elecciones introducidos para una pregunta en la que hay varios grupos de opciones, y donde hay al menos un conjunto de elementos en el que cada elemento tiene su propio número máximo de elecciones con elementos de otro grupo.

Clase `EnteroPositivo`

utilidades

Class EnteroPositivo

```
java.lang.Object
  └utilidades.EnteroPositivo
```

```
public class EnteroPositivo
extends java.lang.Object
```

Clase que contiene la representación de un número entero positivo. Se encarga de recibirlo como un `String` y de comprobar que sea efectivamente un número entero positivo. Contiene métodos para almacenar y recoger el número, e informa de cualquier anomalía en él.

Author:
David Domínguez

Field Summary

private int	**enteroPositivo** Variable con el número ya en un entero.
private java.lang.String	**stringNumero** Variable que contiene el número en un String.

Constructor Summary

EnteroPositivo(java.lang.String stringNumero) Constructor que recibe la representación del número en un `String`.

Method Summary

MensajeEstado	**esPositivo**() Comprueba que el número `String` almacenado sea un número entero positivo.
int	**getEnteroPositivo**() Método "get" para el número entero positivo como un entero.
java.lang.String	**getStringNumero**() Método "get" para el número entero positivo almacenado como un `String`.

Clase `EstadoWeb`

utilidades
Class EstadoWeb

```
java.lang.Object
  └utilidades.MensajeEstado
      └utilidades.EstadoWeb
```

```
public class EstadoWeb
extends MensajeEstado
```

Clase que representa el estado de comprobación de una Web. La clase hereda de `MensajeEstado`, añadiéndole la funcionalidad de almacenar además en qué componente del formulario de la página web se debe localizar el foco en caso de error. Se encarga de almacenar el estado de la comprobación de los datos en cualquier toma de datos de creación de un ítem o un test, y, en caso de error, contiene los mensajes de aviso a mostrar al usuario y dónde posicionar el foco en la página.

Author:
David Domínguez
See Also:
MensajeEstado

Field Summary

protected java.lang.String	**focusON** Nombre del campo erróneo sobre el que irá el foco en la página.

Fields inherited from class utilidades.MensajeEstado
estado, mensaje

Constructor Summary

EstadoWeb() El constructor llama al constructor de la superclase e inicializa la posición del foco a null.

Method Summary

void	**añadeError**(java.lang.String aviso, java.lang.String focus) Añade un error al estado de la comprobación.
java.lang.String	**getFocusON**() Método "get" para obtener la posición del foco.

Methods inherited from class utilidades.MensajeEstado
getMensaje, isEstado, setEstado, setMensaje

Clase `Identificador`

utilidades

Class Identificador

```
java.lang.Object
  └utilidades.Identificador
```

```
public class Identificador
extends java.lang.Object
```

Clase que almacena la representación de un tipo *identifier* válido según la norma QTI del IMS. Contiene métodos para comprobar si es válido, para obtener el identificador, un método estático para comprobar que en un grupo de identificadores sean todos distintos entre sí, y otro método estático para leer un grupo de identificadores del request y comprobar que sean válidos.

Author:
David Domínguez

Field Summary

private java.lang.String	**identificador** Variable de clase con el identificador String genérico

Constructor Summary

Identificador(java.lang.String identificador) Constructor de la clase en el que simplemente se inicializa el valor de la variable de clase con el valor recibido en el parámetro.

Method Summary

static MensajeEstado	**compruebaIdentificadores**(Identificador[] tabla) Método estático para comprobar que los identificadores de una tabla de objetos `Identificador` sean todos diferentes entre sí.
MensajeEstado	**esIdentificador**() Comprueba si el identificador almacenado en la variable `identificador` de tipo `String` de la clase es un identificador válido según la especificación del QTI.
java.lang.String	**getIdentificador**() Método "get" para obtener el valor de la variable de clase.
static void	**testIdentificadores**(Identificador[] identificadores, java.lang.String origen, EstadoWeb estado, HttpServletRequest request, java.lang.String pre) Método estático para leer un grupo de identificadores recibidos como parámetros en el request.

Clase `ListaArchivos`

utilidades
Class ListaArchivos

```
java.lang.Object
  └utilidades.ListaArchivos
```

```
public class ListaArchivos
extends java.lang.Object
```

Clase que contiene métodos estáticos que devuelven determinadas listas del contenido de los directorios.

Author:
David Domínguez

Constructor Summary

ListaArchivos()

Method Summary

static java.util.Vector	**ListaDirectorios**(java.io.File directorio) Devuelve un vector listando los nombres de los directorios que se incluyen en `directorio`.
static java.util.Vector	**ListaImagenes**(java.io.File directorio) Lista las imágenes incluidas en `directorio` (no en subdirectorios de éste).
static java.util.Vector	**ListaItems**(java.io.File directorio) Lista los ítems (en realidad archivos con extensión "xml") incluidos en `directorio` (no en subdirectorios de éste).

Clase `MensajeEstado`

utilidades
Class MensajeEstado

```
java.lang.Object
  └utilidades.MensajeEstado
```

Direct Known Subclasses:
EstadoWeb

```
public class MensajeEstado
extends java.lang.Object
```

Clase que contiene un estado y un mensaje. Se utiliza como salida de muchas comprobaciones, avisando de si es correcta y de qué ha fallado en su caso.

Author:
David Domínguez

Field Summary

protected boolean	**estado** Estado de la comprobación.
protected java.lang.String	**mensaje** Mensaje a almacenar en caso de que haya error.

Constructor Summary

MensajeEstado() Constructor que simplemente pone el estado correcto, inicializando el estado a true.

Method Summary

java.lang.String	**getMensaje**() Método "get" para obtener el mensaje de la clase.
boolean	**isEstado**() Método para obtener el estado de la clase.
void	**setEstado**(boolean estado) Método "set" para establecer el estado de la clase.
void	**setMensaje**(java.lang.String mensaje) Método "set" para establecer el mensaje.

Clase `ParserNeutro`

utilidades

Class ParserNeutro

```
java.lang.Object
  └ utilidades.ParserNeutro
```

```
public class ParserNeutro
extends java.lang.Object
```

Clase que contiene métodos estáticos para escribir con formato XML cadenas que ya tienen algún formato XHTML y comprobar que sean cadenas XML válidas.

Author:
David Domínguez

Constructor Summary

ParserNeutro()	

Method Summary	
static MensajeEstado	**compruebaInstruccionesXHTML**(java.lang.String fragmento) Comprueba un campo de instrucciones de un test o un ítem.
static void	**escribeInstruccionesXML**(java.lang.String xhtml, XMLStreamWriter xsw) Método que escribe un String con formato XHTML en un XMLStreamWriter .
static void	**escribeTextoXML**(java.lang.String texto, XMLStreamWriter xsw) Escribe todos los caracteres con el método writeCharacters(String), excepto los nueva línea, que los sustituye por el equivalente XHTML y los escribe mediante el método writeEmptyElement en el XMLStreamWriter.

Clase ResumenItem

utilidades

Class ResumenItem

java.lang.Object
  └ **utilidades.ResumenItem**

All Implemented Interfaces:

java.lang.Comparable

```
public class ResumenItem
extends java.lang.Object
implements java.lang.Comparable
```

Clase que representa un resumen de los ítems (archivos XML independientes que representan al ítem). El resumen del ítem contiene el nombre del ítem (nombre del archivo XML) en un String, el título del ítem en otro String, y el objeto File que representa al archivo XML del ítem. Contiene un constructor que inicializa todos sus valores, métodos "getters" para obtenerlos, e implementa la interfaz Comparable para permitir que se ordene una colección de objetos ResumenItem.

Author:

David Domínguez

See Also:

Comparable

Field Summary	
private java.io.File	**archivo** File al archivo XML del ítem.
private java.lang.String	**nombre** Nombre del ítem (nombre del archivo XML).
private java.lang.String	**titulo** Título del ítem.

Constructor Summary

ResumenItem(java.lang.String nombre, java.lang.String titulo, java.io.File archivo) Constructor de la clase que simplemente da a las variables de clase los valores recibidos como parámetros.

Method Summary

int	**compareTo**(java.lang.Object o) Método definido por la interfaz `Comparable` para permitir la comparación entre los objetos que implementen la interfaz.
java.io.File	**getArchivo**() Método "get" que devuelve el objeto `File` que representa al archivo XML del ítem.
java.lang.String	**getNombre**() Método "get" que devuelve el `String` con el nombre del ítem.
java.lang.String	**getTitulo**() Método "get" que devuelve el `String` con el título del ítem.

Clase `UrlUtils`

utilidades

Class UrlUtils

```
java.lang.Object
  └utilidades.UrlUtils
```

```
public class UrlUtils
extends java.lang.Object
```

Clase con métodos para codificar URLs.
---------- URL Utils - UrlUtils.java Author: C. Enrique Ortiz Copyright (c) 2004-2005 C. Enrique Ortiz This is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. Usage & redistributions of source code must retain the above copyright notice. This software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details. You should get a copy of the GNU Lesser General Public License from the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA ----------
---
Modificado por David Domínguez.
La clase contiene un método estático para codificar todos los caracteres, excepto los de

tipo `mark` y los permitidos, de un `String` que representa una URL. Se ha añadido un método similar pero que no codifica tampoco los caracteres reservados.

Author:
C. Enrique Ortiz, David Domínguez

Field Summary	
private static java.lang.String	**mark** Caracteres marcas no reservadas
private static java.lang.String	**reservados** Caracteres reservados

Constructor Summary
UrlUtils()

Method Summary	
static java.lang.String	**encodeURL**(java.lang.String url) Encodes a URL - This method assumes UTF-8 Codifica todos los caracteres excepto los permitidos, los reservados y los `mark` .
static java.lang.String	**encodeURLReserved**(java.lang.String url) Encodes a URL - This method assumes UTF-8 Codifica todos los caracteres especiales, incluidos los caracteres reservados.
private static char	**toHexChar**(int digitValue) Converts Hex digit to a UTF-8 "Hex" character.

2.4 Paquete `utilidades.etiquetas.util`

Package utilidades.etiquetas.util

Contiene dos clases que implementan dos etiquetas personalizadas de la biblioteca `util`.

See:
Description

Class Summary	
AdminMailTag	Etiqueta de página JSP para imprimir en pantalla la dirección de correo electrónico del administrador de la Aplicación.
RequestedURITag	Etiqueta de página JSP para imprimir en pantalla la dirección de la página Web que causó el error.

Clase `AdminMailTag`

utilidades.etiquetas.util
Class AdminMailTag

```
java.lang.Object
  └SimpleTagSupport
      └utilidades.etiquetas.util.AdminMailTag
```

```
public class AdminMailTag
extends SimpleTagSupport
```

Etiqueta de página JSP para imprimir en pantalla la dirección de correo electrónico del administrador de la Aplicación. La clase extiende a la clase `SimpleTagSupport` que representa a una Etiqueta Personalizada de JSP Simple. La etiqueta se encarga de devolver por la salida de la página JSP la dirección de correo electrónico de contacto del administrador, parámetro definido en el descriptor de despliegue de la Aplicación, "web.xml". El proceso se realiza en el método sobreescrito `doTag`, que se ejecuta al llamar a la etiqueta en una página JSP.

Author:
David Domínguez

Constructor Summary

AdminMailTag()

Method Summary

void	**doTag**() Imprime por pantalla la dirección de correo electrónico del administrador de la Aplicación.

Clase `RequestedURITag`

utilidades.etiquetas.util
Class RequestedURITag

```
java.lang.Object
  └SimpleTagSupport
      └utilidades.etiquetas.util.RequestedURITag
```

```
public class RequestedURITag
extends SimpleTagSupport
```

Etiqueta de página JSP para imprimir en pantalla la dirección de la página Web que causó el error. La clase extiende a la clase `SimpleTagSupport` que representa a una Etiqueta Personalizada de JSP Simple. La etiqueta se encarga de devolver por la salida de la página JSP, que debe ser una página de error, la URI que provocó el error. El

proceso se realiza en el método sobreescrito `doTag`, que se ejecuta al llamar a la etiqueta en una página JSP.

Author:
David Domínguez

Constructor Summary
RequestedURITag()

Method Summary	
void	**doTag**() Imprime por pantalla la dirección de la Web que provocó el error.

2.5 Paquete `utilidades.etiquetas.util.logging`

Package utilidades.etiquetas.util.logging

Proporciona todas las clases necesarias para el correcto funcionamiento de la etiqueta definida con el nombre de `logger`, encargada del registro de los errores que se producen en la Aplicación Web.

See:
Description

Class Summary	
AplicacionLogger	Arranca y detiene el `Logger` de la Aplicación Web cuando ésta se arranca y se detiene.
ErrorManagerPersonalizado	Manejador de los errores que se produzcan en el `Logger` de la Aplicación Web.
FormatterPersonalizado	`Formater` personalizado para realizar los registros de la Aplicación Web con el formato personalizado.
LoggerTag	Etiqueta de página JSP para realizar el registro de algún error o evento que se requiera registrar de la Aplicación Web.

Clase `AplicacionLogger`

utilidades.etiquetas.util.logging

Class AplicacionLogger

```
java.lang.Object
  └utilidades.etiquetas.util.logging.AplicacionLogger
```

```
public class AplicacionLogger
```

```
extends java.lang.Object
```

Arranca y detiene el `Logger` de la Aplicación Web cuando ésta se arranca y se detiene. La clase implementa la interfaz `ServletContextListener` y ejecutará sus métodos siempre que el contexto de la Aplicación Web se cree o se destruya (normalmente al arrancar o parar la Aplicación). Se encarga de preparar el objeto que se ocupa de las acciones relacionadas con el registro de los errores que se produzcan en la Aplicación, es decir, el "logging". Contiene una variable de clase `String` con el nombre del logger de la Aplicación para acceder a él desde otros puntos de la Aplicación por su nombre, y otra de tipo `Level` donde se define el nivel mínimo de importancia de registro de los errores de la Aplicación.

Author:
David Domínguez
See Also:
`ServletContextListener`

Field Summary

(package private) static java.util.logging.Level	**nivelLogger** Nivel mínimo de importancia de los errores que serán guardados por el logger
(package private) static java.lang.String	**nombreLogger** Nombre del logger de la Aplicación para acceder a él por su nombre en cualquier punto de la Aplicación que se requiera

Constructor Summary

AplicacionLogger()

Method Summary

void	**contextDestroyed**(ServletContextEvent sce) Método que detiene el `Logger` de la Aplicación Web.
void	**contextInitialized**(ServletContextEvent sce) Método que arranca el `Logger` de la Aplicación Web.

Clase `ErrorManagerPersonalizado`

utilidades.etiquetas.util.logging
Class ErrorManagerPersonalizado

```
java.lang.Object
  └java.util.logging.ErrorManager
      └utilidades.etiquetas.util.logging.ErrorManagerPersonalizado
```

```
public class ErrorManagerPersonalizado
```

```
extends java.util.logging.ErrorManager
```

Manejador de los errores que se produzcan en el `Logger` de la Aplicación Web. La clase se encarga de informar de que se ha producido algún error en el proceso del registro de errores. Sobreescribe el método `error` que será el llamado en caso de error al intentar registrar algo con el logger de la Aplicación.

Author:
David Domínguez
See Also:
`ErrorManager`

Field Summary

Fields inherited from class java.util.logging.ErrorManager
CLOSE_FAILURE, FLUSH_FAILURE, FORMAT_FAILURE, GENERIC_FAILURE, OPEN_FAILURE, WRITE_FAILURE

Constructor Summary

ErrorManagerPersonalizado() Constructor de la clase que simplemente llama al constructor de la superclase.

Method Summary

void	**error**(java.lang.String msg, java.lang.Exception ex, int code) Informa de que se ha producido un error al registrar algún evento de la Aplicación Web.

Clase `FormatterPersonalizado`

utilidades.etiquetas.util.logging
Class FormatterPersonalizado

```
java.lang.Object
  └java.util.logging.Formatter
      └utilidades.etiquetas.util.logging.FormatterPersonalizado
```

```
public class FormatterPersonalizado
extends java.util.logging.Formatter
```

`Formater` personalizado para realizar los registros de la Aplicación Web con el formato personalizado. La clase extiende a la clase abstracta `Formatter`, sobreescribiendo sólo su método `format`, aplicando un formato personalizado al registro de los errores de la Aplicación.

Author:
David Domínguez
See Also:

Formatter

Constructor Summary

FormatterPersonalizado()

Method Summary

java.lang.String	**format**(java.util.logging.LogRecord logRecord) Aplica un formato personalizado a la información sobre el error a registrar recibida en el LogRecord.

Methods inherited from class java.util.logging.Formatter
formatMessage, getHead, getTail

Clase LoggerTag

utilidades.etiquetas.util.logging

Class LoggerTag

java.lang.Object
  └ SimpleTagSupport
    └ **utilidades.etiquetas.util.logging.LoggerTag**

public class **LoggerTag**
extends SimpleTagSupport

Etiqueta de página JSP para realizar el registro de algún error o evento que se requiera registrar de la Aplicación Web. La clase extiende a la clase SimpleTagSupport que representa a una Etiqueta Personalizada de JSP Simple. La etiqueta se encarga de registrar algún error o información en el Logger que se creó al iniciar la Aplicación Web. Tiene dos variables de clase que son los dos atributos que se le pasan a la etiqueta, con sus dos métodos "setters" correspondientes: el primero de ellos es un String con el nivel de importancia del registro a realizar, y el segundo el mensaje del registro. Todo el proceso de registro se realiza en el método sobreescrito doTag, que se ejecuta al llamar a la etiqueta en una página JSP. Contiene también un método estático para poner en castellano los nombres cortos de los días de la semana y de los meses del año de un objeto de formato de los símbolos de fechas, DateFormatSymbols.

Author:
David Domínguez
See Also:
javax.servlet.jsp.tagext.SimpleTagSupport

Field Summary

private java.lang.String	**mensaje** String con el mensaje a registrar.

private java.lang.String	**nivel** String con el nombre del nivel de importancia del registro.

Constructor Summary

LoggerTag()

Method Summary

void	**doTag**() Realiza el registro de la información requerida en el `Logger` de la Aplicación.
void	**setMensaje**(java.lang.String mensaje) Método "set" que establece el mensaje a registrar con el `Logger`.
void	**setNivel**(java.lang.String nivel) Método "set" que establece el nombre del nivel de importancia del registro a realizar.
static void	**setNombresCortosCastellano**(java.text.DateFormatSymbols dfs) Personaliza un objeto `DateFormatSymbols` poniendo algunos nombres cortos en castellano.

2.6 Paquete `xml.items`

Package xml.items

Proporciona las clases necesarias para almacenar la información relativa a todos los tipos de ítems y escribirlos en disco.

See:

Description

Class Summary

AssessmentItem	Representación de un ítem XML de la norma QTI del IMS general.
ChoiceXML	Clase contenedora del ítem tipo "Choice".
GapmatchXML	Clase contenedora del ítem tipo "Gap Match".
HottextXML	Clase contenedora del ítem tipo "Hot Text".
InlinechoiceXML	Clase contenedora del ítem tipo "Inline Choice".
MatchXML	Clase contenedora del ítem tipo "Match".
TextentryXML	Clase contenedora del ítem tipo "Text Entry".

Clase `AssessmentItem`

xml.items

Class AssessmentItem

```
java.lang.Object
  └ xml.items.AssessmentItem
```

Direct Known Subclasses:

ChoiceXML, GapmatchXML, HottextXML, InlinechoiceXML, MatchXML, TextentryXML

```
public abstract class AssessmentItem
extends java.lang.Object
```

Representación de un ítem XML de la norma QTI del IMS general. Es una clase abstracta, superclase de todas las clases contenedoras de los ítems XML. Esta clase representa un ítem XML general, y como tal, contiene los tres campos que tienen todos los ítems: `identificador`, `título`, e `instrucciones` del ítem. Contiene un constructor que inicializa esos tres campos. También un método abstracto para escribir el ítem XML en disco y un método para escribir la cabecera común de todos los ítems y otro para el pie.

Author:

David Domínguez

Field Summary

protected java.lang.String	**identificador** El identificador del ítem, que será también el nombre del archivo.
protected java.lang.String	**instrucciones** Instrucciones para responder el ítem.
protected java.lang.String	**titulo** Título del ítem.

Constructor Summary

AssessmentItem(java.lang.String identificador, java.lang.String titulo, java.lang.String instrucciones) Constructor que le da valor a las tres variables de clase.

Method Summary

abstract MensajeEstado	**creaItemXML**(java.lang.String path, ServletContext sc, java.lang.String req) Escribe en disco el ítem XML al que representa esta clase.
java.lang.String	**getIdentificador**() Método "get" para obtener el identificador actual de ítem.
void	**setIdentificador**(java.lang.String identificador) Método "set" para establecer el identificador del ítem.

protected void	**writeCabecera**(XMLStreamWriter xsw, ServletContext sc) Escribe en un writer específico para escribir archivos XML la cabecera común a todos los tipos de ítem XML.
protected void	**writePie**(XMLStreamWriter xsw, ServletContext sc, java.lang.String req) Escribe en un writer específico para escribir archivos XML el pie común a todos los tipos de ítem XML.

Clase ChoiceXML

xml.items

Class ChoiceXML

java.lang.Object
  └ xml.items.AssessmentItem
    └ **xml.items.ChoiceXML**

---

public class **ChoiceXML**
extends AssessmentItem

Clase contenedora del ítem tipo "Choice". Contiene un campo para cada parámetro del ítem necesario para construir el archivo del ítem XML. Con los métodos que incluye permite crear un ítem tipo "Choice" completo y crear el archivo XML en disco.

Author:
David Domínguez
See Also:
Identificador

Field Summary

protected java.lang.String	**cardinalidad** Cardinalidad de la variable respuesta.
protected boolean[]	**correcto** Array de boolean de respuesta correcta.
protected boolean[]	**fija** Array de boolean de respuesta fija.
protected Identificador[]	**identificadores** Identificador de cada una de las respuestas.
protected int	**maxChoices** Número máximo de respuestas seleccionables.
protected java.lang.String	**pregunta** Pregunta del ítem.
protected java.lang.String[]	**respuestas** Texto de cada una de las respuestas.
protected boolean	**shuffle** Mezclar las respuestas del ítem sí o no.

Fields inherited from class xml.items.AssessmentItem
identificador, instrucciones, titulo

Constructor Summary

ChoiceXML(java.lang.String identificador, java.lang.String titulo, java.lang.String instrucciones, java.lang.String pregunta, boolean shuffle) Constructor que permite inicializar el objeto con un primer grupo de parámetros.

Method Summary

MensajeEstado	**creaItemXML**(java.lang.String path, ServletContext sc, java.lang.String req) Método que construye el ítem XML tipo "Choice" en la dirección que recibe como parámetro con los campos que tiene almacenados.
void	**setCorrecto**(boolean[] correcto) Método que inicializa el array de boolean que indica si la respuesta en esa misma posición es correcta.
void	**setFija**(boolean[] fija) Método que inicializa el array de boolean que indica si la respuesta en esa misma posición debe quedar fija en el caso de que se tengan que mezclar aleatoriamente.
void	**setIdentificadores**(Identificador[] identificadores) Método que inicializa el array de objetos Identificador que contiene en cada posición el identificador de la respuesta en esa posición.
void	**setMaxChoices**(int maxChoices) Método que inicializa el número máximo de respuestas que se pueden elegir.
void	**setRespuestas**(java.lang.String[] respuestas) Método que inicializa el array con las respuestas posibles a elegir.

Methods inherited from class xml.items.AssessmentItem
getIdentificador, setIdentificador, writeCabecera, writePie

Clase GapmatchXML

xml.items

Class GapmatchXML

java.lang.Object
  └ xml.items.AssessmentItem
    └ **xml.items.GapmatchXML**

---

public class **GapmatchXML**

extends AssessmentItem

Clase contenedora del ítem tipo "Gap Match". Contiene un campo para cada parámetro del ítem necesario para construir el archivo del ítem XML. Con los métodos que incluye permite crear un ítem tipo "Gap Match" completo y crear el archivo XML en disco.

Author:
David Domínguez
See Also:
Identificador, EnteroPositivo

Field Summary

protected java.lang.String	**cardinalidad** Cardinalidad de la variable respuesta.
protected boolean[]	**fija** Array con las opciones que deben ser fijas.
protected Identificador[]	**identHuecos** Identificadores de los huecos.
protected Identificador[]	**identOpciones** Identificadores de las opciones.
protected EnteroPositivo[]	**maxOpciones** Máximo de asociaciones de cada opción.
protected java.lang.String[]	**opciones** Valores posibles que pueden tomar los huecos.
protected java.lang.String	**pregunta** Pregunta del ítem.
protected int[]	**respCorrecta** Tabla con el número de la opción correcta de cada hueco.
protected boolean	**shuffle** Mezclar las opciones aleatoriamente o no.
protected java.lang.String[]	**textos** Textos en los que están los huecos imbuidos.

Fields inherited from class xml.items.AssessmentItem

identificador, instrucciones, titulo

Constructor Summary

GapmatchXML(java.lang.String identificador, java.lang.String titulo, java.lang.String instrucciones, java.lang.String pregunta, boolean shuffle)
Constructor que permite inicializar el objeto con un primer grupo de parámetros.

Method Summary	
MensajeEstado	**creaItemXML**(java.lang.String path, ServletContext sc, java.lang.String req) Método que construye el ítem XML tipo "Gap Match" con los campos que tiene almacenados en la dirección que recibe como parámetro.
void	**setFija**(boolean[] fija) Método que inicializa el array de boolean indicando si esa opción debe quedarse en esa posición o mezclarse si se deben mezclar aleatoriamente.
void	**setIdentHuecos**(Identificador[] identHuecos) Método que inicializa el array de objetos Identificador que contiene en cada posición el identificador del hueco en esa misma posición.
void	**setIdentOpciones**(Identificador[] identOpciones) Método que inicializa el array de objetos Identificador que contiene en cada posición el identificador de la opción en esa misma posición.
void	**setMaxOpciones**(EnteroPositivo[] maxOpciones) Método que inicializa el array de EnteroPositivo conteniendo en cada elemento el número máximo de huecos con los que esa opción se puede asociar.
void	**setOpciones**(java.lang.String[] opciones) Método que inicializa el array de String conteniendo en cada elemento una de las opciones con las que se pueden asociar los huecos.
void	**setRespCorrecta**(java.lang.String[] respCorrecta) Método que inicializa el array de enteros que indica la posición de la opción que es correcta para cada hueco.
void	**setTextos**(java.lang.String[] textos) Método que inicializa el array de String con los textos que rodean a los huecos.

Methods inherited from class xml.items.AssessmentItem
getIdentificador, setIdentificador, writeCabecera, writePie

Clase HottextXML

xml.items

Class HottextXML

java.lang.Object
  └ xml.items.AssessmentItem
    └ **xml.items.HottextXML**

---

public class **HottextXML**
extends AssessmentItem

Clase contenedora del ítem tipo "Hot Text". Contiene un campo para cada parámetro del ítem necesario para construir el archivo del ítem XML. Con los métodos que incluye permite crear un ítem tipo "Hot Text" completo y crear el archivo XML en disco.

Author:
David Domínguez
See Also:
Identificador

Field Summary

protected java.lang.String	**cardinalidad** Cardinalidad de la variable de respuesta.
protected boolean[]	**correcto** Array de boolean con los hottexts correctos.
protected java.lang.String[]	**hottexts** Hot texts a incluir en el texto.
protected Identificador[]	**identHottexts** Identificadores de los hottexts.
protected int	**maxChoices** Máximo número de hot texts seleccionables.
protected java.lang.String	**pregunta** Pregunta del ítem.
protected java.lang.String[]	**textos** Textos que rodean a los hot texts.

Fields inherited from class xml.items.AssessmentItem

identificador, instrucciones, titulo

Constructor Summary

HottextXML(java.lang.String identificador, java.lang.String titulo, java.lang.String instrucciones, java.lang.String pregunta)
Constructor que permite inicializar el objeto con un primer grupo de parámetros.

Method Summary

MensajeEstado	**creaItemXML**(java.lang.String path, ServletContext sc, java.lang.String req) Método que construye el ítem XML tipo "Hot Text" con los campos que tiene almacenados en la dirección que recibe como parámetro.
void	**setCorrecto**(boolean[] correcto) Método que inicializa el array de boolean que indica si el hot text en esa posición es correcto o no.
void	**setHottexts**(java.lang.String[] hottexts) Método que inicializa el array de String que contiene cada

	uno de los hot texts a incluir en el texto.
void	**setIdentHottexts**(Identificador[] identHottexts) Método que inicializa el array de objetos Identificador que contiene en cada posición el identificador del hot text en esa misma posición.
void	**setMaxChoices**(int maxChoices) Método que inicializa el número máximo de hot texts que se pueden seleccionar como correctos.
void	**setTextos**(java.lang.String[] textos) Método que inicializa el array de String con los textos que rodean a los huecos.

Methods inherited from class xml.items.AssessmentItem
getIdentificador, setIdentificador, writeCabecera, writePie

Clase InlinechoiceXML

xml.items

Class InlinechoiceXML

java.lang.Object
  └ xml.items.AssessmentItem
    └ **xml.items.InlinechoiceXML**

---

public class **InlinechoiceXML**
extends AssessmentItem

Clase contenedora del ítem tipo "Inline Choice". Contiene un campo para cada parámetro del ítem necesario para construir el archivo del ítem XML. Con los métodos que incluye permite crear un ítem tipo "Inline Choice" completo y crear el archivo XML en disco.

Author:
David Domínguez
See Also:
Identificador

Field Summary

(package private) java.lang.String	**cardinalidad** Cardinalidad de la variable respuesta que en este caso siempre es "single".
protected int	**correcto** Posición de la respuesta correcta.
protected boolean[]	**fija** Array con las respuestas que deben ser fijas.
protected Identificador[]	**identificadores** Identificador de cada una de las respuestas.

protected java.lang.String[]	**respuestas** Textos de cada una de las respuestas.
protected boolean	**shuffle** Mezclar las respuestas aleatoriamente sí o no.
protected java.lang.String	**texto1** Texto antes de las respuestas.
protected java.lang.String	**texto2** Texto tras las respuestas.

Fields inherited from class xml.items.AssessmentItem
identificador, instrucciones, titulo

Constructor Summary

InlinechoiceXML(java.lang.String identificador, java.lang.String titulo, java.lang.String instrucciones, boolean shuffle) Constructor que permite inicializar el objeto con un primer grupo de parámetros.

Method Summary

MensajeEstado	**creaItemXML**(java.lang.String path, ServletContext sc, java.lang.String req) Método que construye el ítem XML tipo "Inline Choice" en la dirección que recibe como parámetro con los campos que tiene almacenados.
void	**setCorrecto**(int correcto) Método que inicializa el entero que indica la posición de la respuesta que es correcta.
void	**setFija**(boolean[] fija) Método que inicializa el `array` de `boolean` que indica si la respuesta en esa misma posición debe quedar fija en el caso de que se tengan que mezclar aleatoriamente.
void	**setIdentificadores**(Identificador[] identificadores) Método que inicializa el `array` de objetos `Identificador` que contiene en cada posición el identificador de la respuesta en esa posición.
void	**setRespuestas**(java.lang.String[] respuestas) Método que inicializa el `array` con las respuestas posibles a elegir.
void	**setTextos**(java.lang.String texto1, java.lang.String texto2) Método que inicializa el `array` con los textos antes y después de las opciones.

Methods inherited from class xml.items.AssessmentItem
getIdentificador, setIdentificador, writeCabecera, writePie

Clase MatchXML

xml.items

Class MatchXML

```
java.lang.Object
  └xml.items.AssessmentItem
      └xml.items.MatchXML
```

```
public class MatchXML
extends AssessmentItem
```

Clase contenedora del ítem tipo "Match". Contiene un campo para cada parámetro del ítem necesario para construir el archivo del ítem XML. Con los métodos que incluye permite crear un ítem tipo "Match" completo y crear el archivo XML en disco.

Author:
David Domínguez
See Also:
Identificador, EnteroPositivo

Field Summary

protected java.lang.String	**cardinalidad** Cardinalidad de la variable de respuesta.
protected boolean[]	**fija1** Posiciones fijas de las respuestas del grupo 1.
protected boolean[]	**fija2** Posiciones fijas de las respuestas del grupo 2.
protected Identificador[]	**identificadores1** Identificadores de las respuestas del grupo 1.
protected Identificador[]	**identificadores2** Identificadores de las respuestas del grupo 2.
protected EnteroPositivo[]	**max1** Número máximo de elecciones de cada respuesta del grupo 1.
protected EnteroPositivo[]	**max2** Número máximo de elecciones de cada respuesta del grupo 2.
protected int	**maxChoices** Número total máximo de elecciones.
protected java.lang.String	**pregunta** Pregunta del ítem.
protected	**respCorrecta**

boolean[][]	Combinaciones correctas entre los dos grupos.
protected java.lang.String[]	**respuestas1** Respuestas del grupo 1.
protected java.lang.String[]	**respuestas2** Respuestas del grupo 2.
protected boolean	**shuffle** Mezclar las respuestas aleatoriamente sí o no.

Fields inherited from class xml.items.AssessmentItem
identificador, instrucciones, titulo

Constructor Summary

MatchXML(java.lang.String identificador, java.lang.String titulo, java.lang.String instrucciones, java.lang.String pregunta, boolean shuffle) Constructor que permite inicializar el objeto con un primer grupo de parámetros.

Method Summary

MensajeEstado	**creaItemXML**(java.lang.String path, ServletContext sc, java.lang.String req) Método que construye el ítem XML tipo "Match" en la dirección que recibe como parámetro con los campos que tiene almacenados.
void	**setFija1**(boolean[] fija1) Método que inicializa el array de boolean que indica si la respuesta del grupo 1 en esa misma posición debe quedar fija en el caso de que se tengan que mezclar aleatoriamente.
void	**setFija2**(boolean[] fija2) Método que inicializa el array de boolean que indica si la respuesta del grupo 2 en esa misma posición debe quedar fija en el caso de que se tengan que mezclar aleatoriamente.
void	**setIdentificadores1**(Identificador[] identificadores1) Método que inicializa el array de objetos Identificador que contiene en cada posición el identificador de la opción del grupo 1 en esa posición.
void	**setIdentificadores2**(Identificador[] identificadores2) Método que inicializa el array de objetos Identificador que contiene en cada posición el identificador de la opción del grupo 2 en esa posición.
void	**setMax1**(EnteroPositivo[] max1) Método que inicializa el array de EnteroPositivo conteniendo en cada elemento el número máximo de opciones del grupo 2 con las que esa opción del grupo 1 se puede asociar.
void	**setMax2**(EnteroPositivo[] max2)

	Método que inicializa el `array` de `EnteroPositivo` conteniendo en cada elemento el número máximo de opciones del grupo 1 con las que esa opción del grupo 2 se puede asociar.
`void`	**`setMaxChoices`**`(int maxChoices)` Método que inicializa el número máximo de asociaciones en total que se pueden hacer entre elementos del grupo 1 y del grupo 2.
`void`	**`setRespCorrecta`**`(boolean[][] respCorrecta)` Método que inicializa el `array` de dos dimensiones de `boolean` que indica si la combinación de la opción "i" del grupo 1 con la "j" del grupo 2 es correcta o no.
`void`	**`setRespuestas1`**`(java.lang.String[] respuestas1)` Método que inicializa el `array` con las opciones posibles del grupo 1.
`void`	**`setRespuestas2`**`(java.lang.String[] respuestas2)` Método que inicializa el `array` con las opciones posibles del grupo 2.

Methods inherited from class xml.items.AssessmentItem
`getIdentificador`, `setIdentificador`, `writeCabecera`, `writePie`

Clase `TextentryXML`

xml.items

Class TextentryXML

```
java.lang.Object
  └ xml.items.AssessmentItem
      └ xml.items.TextentryXML
```

```
public class TextentryXML
extends AssessmentItem
```

Clase contenedora del ítem tipo "Text Entry". Contiene un campo para cada parámetro del ítem necesario para construir el archivo del ítem XML. Con los métodos que incluye permite crear un ítem tipo "Text Entry" completo y crear el archivo XML en disco.

Author:
David Domínguez

Field Summary

protected java.lang.String	**`cardinalidad`** Cardinalidad de la variable de respuesta.
protected int	**`longitud`** Longitud esperada de la respuesta; inicializada a cero si no se debe incluir el atributo.
protected java.lang.String	**`respuesta`**

	Respuesta correcta del ítem.
protected java.lang.String	**texto1** Texto antes de la respuesta.
protected java.lang.String	**texto2** Texto después de la respuesta.

Fields inherited from class xml.items.AssessmentItem
identificador, instrucciones, titulo

Constructor Summary

TextentryXML(java.lang.String identificador, java.lang.String titulo, java.lang.String instrucciones) Constructor que permite inicializar el objeto con un primer grupo de parámetros.

Method Summary

MensajeEstado	**creaItemXML**(java.lang.String path, ServletContext sc, java.lang.String req) Método que construye el ítem XML tipo "Text Entry" en la dirección que recibe como parámetro con los campos que tiene almacenados.
void	**setLongitud**(int longitud) Método que inicializa el valor de `longitud`, la longitud esperada de la respuesta.
void	**setRespuesta**(java.lang.String respuesta) Método que inicializa el `String` con la respuesta correcta.
void	**setTextos**(java.lang.String texto1, java.lang.String texto2) Método que inicializa el `array` con los textos antes y después del hueco a rellenar por el usuario.

Methods inherited from class xml.items.AssessmentItem
getIdentificador, setIdentificador, writeCabecera, writePie

2.7 Paquete `xml.test`

Package xml.test

Proporciona todas las clases que definen un test, correspondiéndose con la estructura de clases que define la norma QTI para un test.

See:

Description

Class Summary	
AssessmentItemRef	Clase que representa las referencias a los ítems individuales dentro de un Test según la norma QTI del IMS.
AssessmentSection	Clase que representa a una sección del Test según la norma QTI del IMS.
AssessmentTest	Clase que representa a un Test según la norma QTI del IMS.
ItemSessionControl	Clase que representa un objeto de control de la sesión al realizar el Test según la norma QTI del IMS.
Ordering	Clase que representa el criterio de ordenación de los ítems de una sección de un Test según la norma QTI del IMS.
SectionPart	Clase que representa a la clase abstracta del mismo nombre según la norma QTI del IMS.
Selection	Clase que representa el criterio de selección de los ítems de una sección de un Test según la norma QTI del IMS.
TestPart	Clase que representa una Parte de un Test según la norma QTI del IMS.
Weight	Clase que representa un Peso de un ítem según la norma QTI del IMS.

Clase `AssessmentItemRef`

xml.test

Class AssessmentItemRef

```
java.lang.Object
  └xml.test.SectionPart
      └xml.test.AssessmentItemRef
```

```
public class AssessmentItemRef
extends SectionPart
```

Clase que representa las referencias a los ítems individuales dentro de un Test según la norma QTI del IMS. Se utiliza para incorporar cada una de las cuestiones individuales dentro de un Test. Contiene varios atributos que se corresponden con los atributos definidos en la norma, más una variable tipo `File` del archivo del ítem a incluir en el test para ayudar a calcular su URI relativa. Tal y como define la norma, esta clase hereda de la clase "SectionPart", por lo tanto también contendrá sus variables de clase.

Author:
David Domínguez

Field Summary	
private java.net.URI	**href** URI relativa desde el test con la referencia al ítem XML que se

	debe incluir en el Test.
private java.io.File	**itemFile** Objeto File apuntando al ítem XML al que hace referencia esta clase.
private Weight	**noOk** Objeto de tipo Weight con el peso que aplicar a la pregunta en caso de que sea incorrecta
private Weight	**ok** Objeto de tipo Weight con el peso que aplicar a la pregunta en caso de que sea correcta

Fields inherited from class xml.test.SectionPart
fijo, identificador, requerido

Constructor Summary

AssessmentItemRef(Identificador identificador, boolean requerido, boolean fijo, java.io.File itemFile, Weight ok, Weight noOk) Constructor de la clase referencia a un ítem.

Method Summary

void	**creaAssessmentItemRef**(XMLStreamWriter xsw) Método que escribe este objeto AssessmentItemRef en el XMLStreamWriter recibido como parámetro, según la norma QTI definida.
void	**creaItemRef**(java.io.File test) Calcula la URI de referencia del ítem a partir de la localización del propio ítem y de la del test que se está creando, hallando la dirección relativa del ítem respecto al test.

Methods inherited from class xml.test.SectionPart
getIdentificador

Clase AssessmentSection

xml.test

Class AssessmentSection

```
java.lang.Object
  └ xml.test.SectionPart
      └ xml.test.AssessmentSection
```

```
public class AssessmentSection
extends SectionPart
```

Clase que representa a una sección del Test según la norma QTI del IMS. Se utiliza para agrupar varios ítems dentro de un Test. Contiene varios atributos que se corresponden

con los atributos definidos en la norma, como son el título de la sección, si es visible, si se sigue algún criterio de selección y ordenación, y el array de objetos `AssessmentItemRef` con las referencias de los ítems incluidos en la sección. Tal y como define la norma esta clase hereda de la clase "SectionPart", por lo tanto también contendrá sus variables de clase.

Author:
David Domínguez

Field Summary

private java.lang.String	**instrucciones** Instrucciones del ítem.
private AssessmentItemRef[]	**itemRefArray** Array de referencias a los ítems que se incluyen en la sección.
private Ordering	**orden** Criterio de ordenación de los ítems del test.
private Selection	**seleccion** Criterio de selección de los ítems posibles a seguir.
private java.lang.String	**titulo** Título de la sección.
private boolean	**visible** Si la sección es visible para el candidato que realice el test.

Fields inherited from class xml.test.SectionPart

fijo, identificador, requerido

Constructor Summary

AssessmentSection(Identificador identificador, java.lang.String titulo, boolean visible, Selection seleccion, Ordering orden, java.lang.String instrucciones)
Constructor que inicializa variables de clase.

Method Summary

void	**creaAssessmentSection**(XMLStreamWriter xsw) Método que escribe este objeto `AssessmentSection` en el `XMLStreamWriter` recibido como parámetro, según la norma QTI definida.
void	**creaRefItems**(java.io.File referencia) Calcula la URI relativa de todos los ítems de la sección.
AssessmentItemRef[]	**getItemRefArray**() Método "get" para obtener el array de objetos de referencia a los ítems, `AssessmentItemRef`, de la sección.

Ordering	**getOrden**() Método "get" para obtener el criterio de ordenación de la sección.
Selection	**getSeleccion**() Método "get" para obtener el criterio de selección de los ítems de la sección.
void	**setItemRefArray**(AssessmentItemRef[] itemRefArray) Método "set" para establecer el array de objetos de referencia a los ítems, AssessmentItemRef, de la sección.

Methods inherited from class xml.test.SectionPart
getIdentificador

Clase AssessmentTest

xml.test

Class AssessmentTest

java.lang.Object
  └ **xml.test.AssessmentTest**

---

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

Clase que representa a un Test según la norma QTI del IMS. Contiene varios atributos que se corresponden con los atributos definidos en la norma, como son el título, el identificador del test, y las Partes del Test, TestPart, que incluye (en esta implementación sólo puede incluir una Parte de Test en el Test).

Author:
David Domínguez

Field Summary

private java.lang.String	**identificador** Identificador del Test, que será también el nombre del archivo XML que lo representa.
private TestPart	**testPart** Objeto TestPart que contiene el Test, donde está la sección con las referencias a los ítems que incluye el Test.
private java.lang.String	**titulo** Título del Test.

Constructor Summary

AssessmentTest(java.lang.String identificador, java.lang.String titulo, TestPart testPart) Constructor de la clase que le da los valores iniciales al identificador, al título y

a la Parte de Test que incluye el Test.

Method Summary

MensajeEstado	**creaAssessmentTest**(java.lang.String path, ServletContext sc) Método que escribe este objeto AssessmentTest en el directorio indicado por el String path.
java.lang.String	**getIdentificador**() Método "get" para obtener el identificador del Test.
TestPart	**getTestPart**() Método "get" para obtener la Parte del Test que incluye el Test.
java.lang.String	**getTitulo**() Método "get" para obtener el título del Test.
void	**setIdentificador**(java.lang.String identificador) Método "set" para establecer un nuevo identificador para el test.

Clase ItemSessionControl

xml.test

Class ItemSessionControl

```
java.lang.Object
  └ xml.test.ItemSessionControl
```

```
public class ItemSessionControl
extends java.lang.Object
```

Clase que representa un objeto de control de la sesión al realizar el Test según la norma QTI del IMS. Contiene sólo una variable, que se corresponde con el atributo "allowReview" definido en la norma.

Author:
David Domínguez

Field Summary

(package private) boolean	**allowReview** Permitir al candidato revisar la sección tras responderla con las respuestas que ha dado.

Constructor Summary

ItemSessionControl(boolean allowReview) Constructor que establece el valor del único campo de la clase, allowReview.

Method Summary	
void	**creaItemSessionControl**(XMLStreamWriter xsw) Método que escribe este objeto ItemSessionControl en el XMLStreamWriter recibido como parámetro, según la norma QTI definida.

Clase `Ordering`

xml.test

Class Ordering

```
java.lang.Object
  └ xml.test.Ordering
```

```
public class Ordering
extends java.lang.Object
```

Clase que representa el criterio de ordenación de los ítems de una sección de un Test según la norma QTI del IMS. El único criterio definido por la norma es permitir mezclar los ítems aleatoriamente o dejarlos en la posición definida en el Test. Contiene un atributo que se corresponde con el definido en la norma.

Author:
David Domínguez

Field Summary	
private boolean	**shuffle** Variable indicando si se deben mezclar aleatoriamente los ítems o no.

Constructor Summary
Ordering(boolean shuffle) Constructor que simplemente inicializa la variable de la clase boolean.

Method Summary	
void	**creaOrdering**(XMLStreamWriter xsw) Método que escribe este objeto Ordering en el XMLStreamWriter recibido como parámetro, según la norma QTI definida.

Clase `SectionPart`

xml.test

Class SectionPart

```
java.lang.Object
  └ xml.test.SectionPart
```

Direct Known Subclasses:
AssessmentItemRef, AssessmentSection

```
public abstract class SectionPart
extends java.lang.Object
```

Clase que representa a la clase abstracta del mismo nombre según la norma QTI del IMS. Es la superclase de otras clases que componen un Test. Contiene los atributos que se corresponden con los definidos en la norma.

Author:
David Domínguez

Field Summary

protected boolean	**fijo** boolean indicando si la subclase está fija en caso de que se use algún criterio de ordenación.
protected Identificador	**identificador** Identificador del elemento.
protected boolean	**requerido** boolean indicando si la subclase está requerida obligatoriamente en caso de que se use algún criterio de selección.

Constructor Summary

SectionPart(Identificador identificador, boolean requerido, boolean fijo) Constructor de la clase que simplemente le da valores iniciales a las variables de clase.

Method Summary

Identificador	**getIdentificador**() Método "get" para obtener el identificador de la clase.

Clase `Selection`

xml.test

Class Selection

```
java.lang.Object
  └xml.test.Selection
```

```
public class Selection
extends java.lang.Object
```

Clase que representa el criterio de selección de los ítems de una sección de un Test según la norma QTI del IMS. El único criterio definido por la norma es seleccionar un subconjunto del conjunto total de ítems de la sección. Contiene unos atributos que se correspondes con los definidos en la norma.

Author:
David Domínguez

Field Summary

private boolean	**conReemplazamiento** Si se pueden repetir los ítems escogiéndolos más de una vez o no.
private int	**seleccion** Número de ítems a seleccionar del número total de ítems.

Constructor Summary

Selection(int seleccion, boolean conReemplazamiento) Constructor de la clase que simplemente le da valores iniciales a las variables de la clase.

Method Summary

void	**creaSelection**(XMLStreamWriter xsw) Método que escribe este objeto Selection en el XMLStreamWriter recibido como parámetro, según la norma QTI definida.
int	**getSeleccion**() Método "get" que devuelve el número de ítems a seleccionar del conjunto de los ítems de la sección.

Clase TestPart

xml.test

Class TestPart

java.lang.Object
  └ **xml.test.TestPart**

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

Clase que representa una Parte de un Test según la norma QTI del IMS. Se utiliza para agrupar Secciones. Contiene unos atributos que se corresponden con los definidos en la norma.

Author:
David Domínguez

Field Summary

private ItemSessionControl	**control** Objeto de control de la Parte del Test.
private Identificador	**identificador** Identificador de la Parte del Test.

private java.lang.String	**navegacion** Modo de navegación por la Parte del Test.
private java.lang.String	**presentacion** Modo de presentación de la Parte del Test.
private AssessmentSection	**seccion** Sección que incluye esta Parte del Test.

Constructor Summary

TestPart(Identificador identificador, java.lang.String navegacion, java.lang.String presentacion, ItemSessionControl control, AssessmentSection seccion) Constructor que simplemente inicializa las variables de la clase.

Method Summary

void	**creaTestPart**(XMLStreamWriter xsw) Método que escribe este objeto `TestPart` en el `XMLStreamWriter` recibido como parámetro, según la norma QTI definida.
AssessmentSection	**getSeccion**() Método "get" para obtener la Sección, el objeto `AssessmentSection` que incluye esta Parte del Test.

Clase `Weight`

xml.test

Class Weight

```
java.lang.Object
  └ xml.test.Weight
```

```
public class Weight
extends java.lang.Object
```

Clase que representa un Peso de un ítem según la norma QTI del IMS. Se utiliza para aplicar pesos distintos a los distintos ítems que componen el Test a la hora de evaluarlos. Contiene unos atributos que se corresponden con los definidos en la norma.

Author:
David Domínguez

Field Summary

private Identificador	**identificador** Identificador válido del peso.
private double	**peso** Valor del peso que tiene el ítem dentro del Test.

Constructor Summary

Weight(Identificador identificador, double peso) Constructor que simplemente le da valores iniciales a las variables de la clase.

Method Summary

void	**creaWeight**(XMLStreamWriter xsw) Método que escribe este objeto `Weight` en el `XMLStreamWriter` recibido como parámetro, según la norma QTI definida.