ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA, UNIVERSIDAD DE VIGO

Servidor Híbrido de Documentos Estructurados

Segunda Entrega

Desarrollo de Aplicaciones para Internet
Grupo 1.2

García Limón, Jesús 34274576-E
Gutiérrez Jácome, Alberto 44473906-W
Vázquez Fernández, Pablo 44656466-B

22/12/2013

Contenido

Introducción

Se plantea el desarrollo de un sistema híbrido cliente/servidor y P2P para la gestión de documentos estructurados (HTML y XML). Dicho proyecto se engloba dentro del ámbito de la asignatura Desarrollo de Aplicaciones para Internet, y su objetivo primordial es la aplicación de los conceptos teóricos cubiertos en la asignatura.

El desarrollo del proyecto contará con dos entregas, siendo esta aquí descrita la segunda y última de ellas, centrada en la implementación de un sistema similar a un servidor web que, además de servir documentos HTML (parte de la primera entrega), sirva también ficheros XML, XSD y XSLT a través de un sistema P2P entre los servidores desplegados. A mayores, los documentos XML podrán ser transformados con los documentos XSLT y validados con los documentos XSD.

Motivación del sistema

Dada la actual necesidad de disponer de acceso a documentos estructurados, HTML y XML, el sistema proporciona una correcta gestión de los mismos, permitiendo las operaciones básicas de alta, baja y consulta en toda la red P2P de servidores. Así pues, disponer o no de un documento en el servidor local no supondrá ningún problema siempre que dicho documento esté almacenado en cualquier servidor de la red.

Con la existencia de dicha red de servidores híbridos, los usuarios podrán salvaguardar regularmente todos sus documentos, permitiendo reducir los riesgos ocasionados por la pérdida de ficheros, y poder volver a acceder a los mismos desde cualquier equipo conectado a cualquier servidor de la red. Se descentraliza así el almacenamiento de los documentos, permitiendo un acceso más cómodo para el usuario final.

Arquitectura del sistema

Descripción general

El sistema se ha planteado en base a una arquitectura en tres capas, siendo la capa inferior la responsable del acceso a la base de datos, la capa intermedia la responsable de implementar la lógica del sistema y la capa superior, encargada de presentar todos los documentos del sistema a través de un servidor HTTP. La capa de lógica contendrá todos los controladores necesarios para la implementación correcta del sistema, y la capa de acceso a datos contará con todos los objetos DAO y entidades necesarias para dar soporte a la lógica de la capa superior. A mayores de estas tres capas, existirán un conjunto de clases generales para todas las capas, entre las que se incluyen todas las excepciones definidas, utilizadas para la notificación de errores entre las distintas capas, y la clase de configuración del sistema, que almacenará en un objeto Singleton todos los parámetros necesarios para el correcto funcionamiento del sistema, según lo definido por el usuario en el fichero de configuración.

La comunicación entre dichas capas se realizará solamente desde las capas superiores a las capas inferiores, así la capa del servidor HTTP sólo se comunicará con la capa de lógica del sistema, y la capa de lógica del sistema sólo se comunicará con la capa de acceso a datos, y nunca en el sentido contrario.

Diagrama de clases

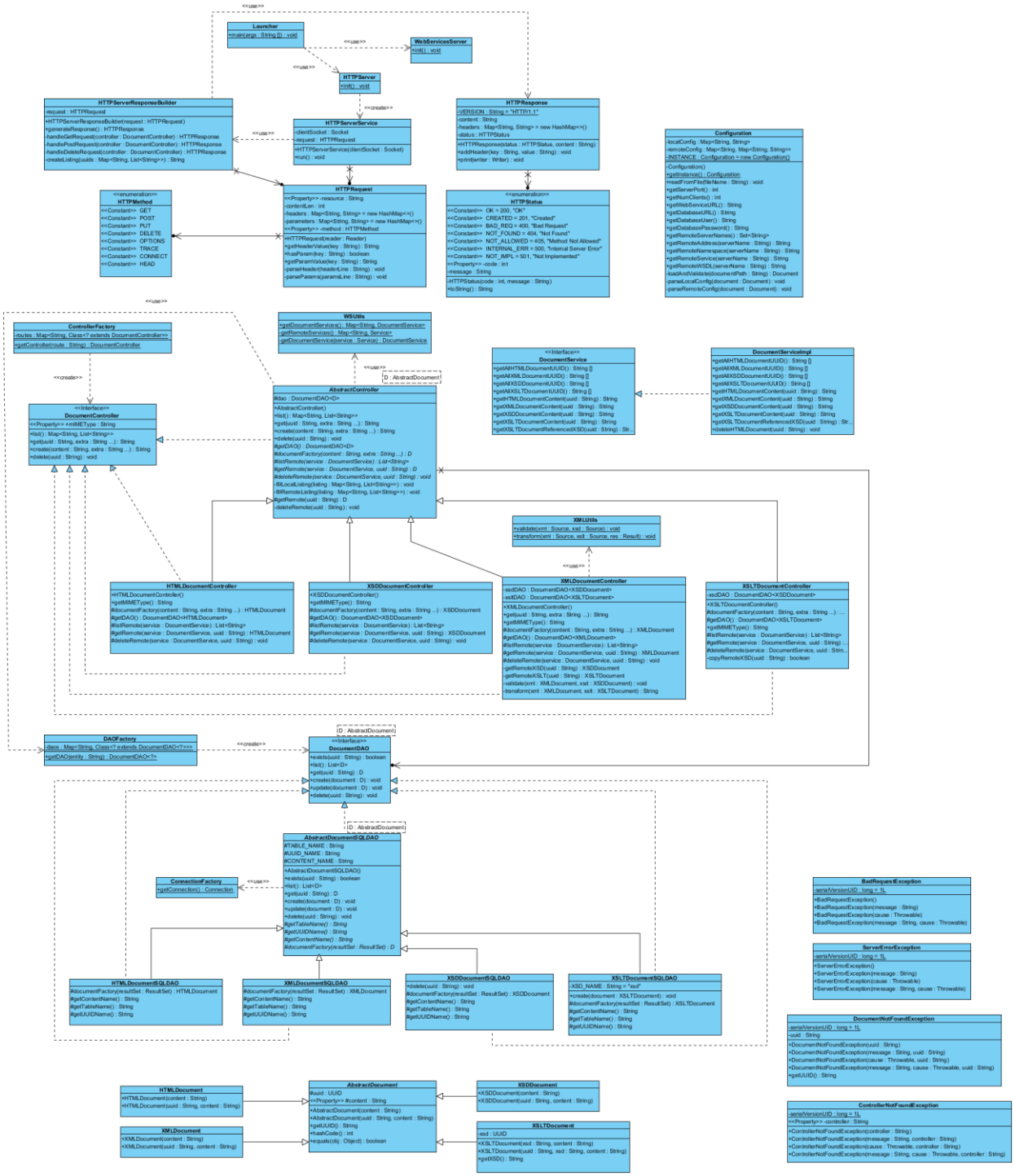

Tabla de responsabilidades

Clase	Responsabilidad
DocumentService	Interfaz para la gestión remota de documentos a través de Web Services.
DocumentServiceImpl	Implementación de la interfaz para la gestión remota de documentos.
WSUtils	Clase estática que proporciona métodos de utilidad para el trabajo con WebServices por parte de los controladores.
XMLUtils	Clase estática que proporciona métodos de utilidad para el trabajo con documentos XML por parte de los controladores.
HTMLDocumentController	Controlador para la gestión de documentos HTML.
XMLDocumentController	Controlador para la gestión de documentos XML.
XSDDocumentController	Controlador para la gestión de documentos XSD.
XSLTDocumentController	Controlador para la gestión de documentos XSLT.
AbstractController	Clase abstracta proporcionando una implementación común a todos los controladores de documentos.
DocumentController	Interfaz a implementar por todos los controladores de documentos estructurados.
ControllerFactory	Factoría de controladores.
AbstractDocumentSQLDAO	Clase abstracta proporcionando una implementación común a todos los DAO de SQL para documentos.
HTMLDocumentSQLDAO	DAO para el acceso a documentos HTML en una base de datos SQL.
XMLDocumentSQLDAO	DAO para el acceso a documentos XML en una base de datos SQL.
XSDDocumentSQLDAO	DAO para el acceso a documentos XSD en una base de datos SQL.

XSLTDocumentSQLDAO	DAO para el acceso a documentos XSLT en una base de datos SQL.
DocumentDAO	Interfaz a implementar por todos los DAO para el acceso a documentos del sistema.
AbstractDocument	Clase abstracta proporcionando una implementación común para todos los documentos del sistema.
HTMLDocument	Clase para el manejo de documentos HTML.
XMLDocument	Clase para el manejo de documentos XML.
XSDDocument	Clase para el manejo de documentos XSD.
XSLTDocument	Clase para el manejo de documentos XSLT.
ConnectionFactory	Factoría de conexiones a la base de datos configurada en el sistema.
DAOFactory	Factoría de objetos DAO para el acceso a datos según el tipo de documento solicitado.
BadRequestException	Excepción para la notificación de peticiones incorrectas al servidor.
ControllerNotFoundException	Excepción para la notificación de solicitud incorrecta de controlador (inexistencia del mismo).
DocumentNotFoundException	Excepción para la notificación de solicitud de un documento no existente.
ServerErrorException	Excepción para la notificación de un error interno por parte del servidor.
HTTPMethod	Enumerado representando todos los métodos HTTP posibles.
HTTPRequest	Clase para el parseo y almacenamiento de peticiones HTTP al servidor.
HTTPResponse	Clase para el almacenamiento y devolución de respuestas HTTP desde el servidor al cliente.

HTTPStatus	Enumerado representando todos los estados de respuesta HTTP que utiliza el servidor.
HTTPServer	Clase encargada de iniciar el servidor HTTP y esperar por conexiones a un Socket.
HTTPServerService	Clase encargada de la gestión de los clientes conectados al Socket iniciado por el servidor HTTP.
HTTPServerResponseBuilder	Clase encargada de la construcción de una respuesta HTTP adecuada según la petición recibida al servidor.
WebServicesServer	Clase encargada de la publicación del Web Service asociado al servidor.
Configuration	Clase Singleton encargada de obtener y almacenar todos los parámetros de configuración desde un fichero XML.
Launcher	Punto de entrada para la ejecución del servidor híbrido.

Manual de usuario

Se asume la previa compilación, con un compilador de código Java, de todo el código fuente dentro del árbol de directorios `src/`. Se asume, además, que todos los ficheros .class generados en la compilación cuelgan de un árbol de directorios `bin/` bajo la misma estructura que el directorio del código fuente original. Para dicha tarea se recomienda el uso de alguna herramienta de compilación de código Java, como pueden ser Ant, Maven o incluso algún IDE como Eclipse o IntelliJ, y no ser realizada manualmente con el compilador de Java (javac).

Configuración

La configuración del servidor híbrido de documentos estructurados se realiza a través de un fichero XML encontrado en la raíz del proyecto bajo el nombre configuration.xml. Dicho fichero de configuración cuenta con tres grandes bloques de parámetros, el primero de ellos, `connections`.

El elemento http hace referencia al puerto donde escuchará el servidor dentro de la máquina local que lo ejecute. El elemento `webservice` hace referencia a la URL que utilizará el servidor dentro de la máquina local para la publicación del servicio web desde el que podrán solicitar documentos el resto de los servidores de la red P2P. El último de los elementos, `numClients` especifica el número de clientes concurrentes que es capaz de soportar el servidor ejecutándose en la máquina local.

El segundo de los bloques de configuración, `database`, hace referencia a la conexión a la base de datos para el servidor local.

El elemento `user` hace referencia al nombre de usuario que el servidor local utilizará para conectarse a la base de datos. A su vez, el elemento `password` hace referencia a la contraseña de acceso a la base de datos utilizada para dicho usuario. Y por último, el elemento `url` hace referencia a la URL de conexión a la base de datos, que será utilizada por JDBC para realizar la conexión desde el programa Java.

El último bloque de configuración, `servers`, hace referencia a los distintos servidores remotos a utilizar para recuperar documentos que no se encuentren dentro del servidor local, proporcionando así una red P2P para compartir documentos dentro de todos los servidores híbridos desplegados.

El bloque `servers` puede contar con tantos elementos `server` como considere necesario según la red de servidores utilizada. Cada elemento server cuenta con un conjunto de atributos que definen la conexión realizada hacia el mismo, `name` establece el nombre que el servidor local utilizará para nombrar a dicho servidor (se mostrará en los listados de documentos), `wsdl`

hace referencia a la URL del WSDL del servidores remoto, al igual que `namespace` hace referencia al espacio de nombres que el servicio del servidor remoto utiliza. El atributo `service` hace referencia al servicio proporcionado por el servidor remoto, que utilizará el servidor local para obtener los documentos. Se tratará de una clase Java que implemente la interfaz DocumentService existente dentro del proyecto en el paquete `es.uvigo.esei.dai.hybridserver.controller.service`. Por último, el atributo `httpAddress` hace referencia a la URL del servidor HTTP remoto.

Conste que dicho fichero de configuración será validado según el esquema XSD existente en `xml/configuration.xsd`, y que es necesario proporcionar la localización de dicho fichero en el atributo `xsi:schemaLocation` dentro del elemento raíz `configuration` del fichero de configuración. El atributo schemaLocation proporcionado en el fichero de configuración de ejemplo ha sido especificado de forma correcta para que la validación pueda realizarse satisfactoriamente.

Ejecución

Para la ejecución del servidor híbrido de documentos estructurados, desde la raíz del proyecto, y asumiendo que todo el código haya sido compilado dentro del directorio "bin", puede realizarse con el siguiente comando:

```
$ java -cp bin:lib/mysql.jar es.uvigo.esei.dai.hybridserver.Launcher
configuration.xml
```

Nótese la inclusión en el Classpath del fichero `lib/mysql.jar`. dicho fichero JAR proporcionaría el driver para la conexión a una base de datos MySQL. Éstos drivers serán necesarios siempre que se desee realizar una conexión hacia una base de datos, y dependerán de la plataforma concreta. Habitualmente, la página web del sistema de bases de datos dispone de un apartado de descargas desde donde obtener dichos ficheros JAR con la implementación del driver adecuado.

Configuración de la base de datos

La base de datos hacia donde se desee conectar el servidor deberán estar disponibles las tablas `HTML`, `XMLT`, `XSD` y `XSLT`. Todas ellas debeán contar con una columna `uuid` y una columna `content`. A mayores, la tabla `XSLT` debe contar con otra columna `xsd` que haga referencia al `uuid` de la tabla `XSD`, pero sin configurarse como una clave foránea hacia la misma.

Dentro del directorio `sql/` se proporcionan dos scripts SQL para la creacion de dichas tablas, junto a la inserción de diez documentos HTML de prueba, en un sistema MySQL y en un sistema Apache Derby.

El script para Apache Derby asumira que ya se ha establecido (e.g. por consola) una conexion a una base de datos existente donde crear las tablas, puesto que el sistema no proporciona ninguna sintaxis SQL para la creacion directa. Mientras tanto, el script para MySQL se encargará de la creación de una base de datos con el nombre `HybridServer` y el usuario `dai_user` con contraseña `dai_pass`.