Ingenieria de software 2

Primer parcial

Ultima modificación el lunes 29 de septiembre de 2008 a las 02:50:50

Aplicación web

Aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador.

Aunque existen muchas variaciones posibles, una aplicación web está normalmente estructurada como una aplicación de tres-capas. En su forma más común, el navegador web ofrece la primera capa y un motor capaz de usar alguna tecnología web dinámica (ejemplo: PHP, Java Servlets o ASP, ASP.NET, CGI, ColdFusion, embPerl, Python o Ruby on Rails) constituye la capa de enmedio. Por último, una base de datos constituye la tercera y última capa.

El navegador web manda peticiones a la capa de enmedio que ofrece servicios valiéndose de consultas y actualizaciones a la base de datos y a su vez proporciona una interfaz de usuario.

Servlets

Un servlet es un objeto que se ejecuta en un servidor o contenedor JEE, fue especialmente diseñado para ofrecer contenido dinámico desde un servidor web.

La palabra servlet deriva de otra anterior, applet, que se refería a pequeños programas escritos en Java que se ejecutan en el contexto de un navegador web. Por contraposición, un servlet es un programa que se ejecuta en un servidor.

El uso más común de los servlets es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

Los servlets forman parte de JEE (Java Enterprise Edition), que es una ampliación de JSE (Java Standard Edition).

Un servlet es un objeto Java que implementa la interfaz javax.servlet.Servlet o hereda alguna de las clases más convenientes para un protocolo específico (ej: javax.servlet.HttpServlet). Al implementar esta interfaz el servlet es capaz de interpretar los objetos de tipo HttpServletRequest y HttpServletResponse quienes contienen la información de la página que invocó al servlet.

Entre el servidor de aplicaciones (o web content) y el servlet existe un contrato que determina cómo han de interactuar. La especificación de éste se encuentra en los JSR (Java Specification Requests) del JCP (Java Community Process).

Archivo WAR (Web Application Archive)

Es un archivo JAR usado para distribuir la collecion de paginas JavaServer, servlets, clases Java, archivos XML, librerias y paginas web estaticas HTML que juntas constituyen una **aplicación web.**

Archivo EAR (Enterprise ARchive)

Un archivo sado por Java EE para empaquetar uno o mas modulos de manera que su instalacion en un servidor de aplicaciones sea simultanea y coherente. Ademas contiene archivos XML llamados descriptores de instalacion (deployment descriptors) que describen como instalar los modulos. Maven o Ant se pueden usar para construir archivos EAR.

Java Server Pages (JSP)

JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

Permite incluir codigo Java y ciertas acciones predefinidas dentro de contenido estatico.

La sintaxis JSP agrega tags estlio XML, llamadas acciones JSP, que se usan para invocar funcionalidad predefinida. Adicionalmente, se pueden crear librerias de tags JSP que sirven para extender las tags standard HTML o XML. Estas librerias proveen una manera de extender la funcionalidad de un servidor web.

Los JSPs son en realidad servlets: un JSP se compila a un programa en Java la primera vez que se invoca, y del programa en Java se crea una clase que se empieza a ejecutar en el servidor como un servlet.

La principal diferencia entre los servlets y los JSPs es el enfoque de la programación: un JSP es una página Web con etiquetas especiales y código Java incrustado, mientras que un servlet es un programa que recibe peticiones y genera a partir de ellas una página web.

JSP puede considerarse como una manera alternativa, y simplificada, de construir servlets.

Para ejecutar las páginas JSP, se necesita un servidor Web con un contenedor Web que cumpla con las especificaciones de JSP y de Servlet.

Sintaxis JSP

Una pagina JSP puede dividirse en:

- datos estaticos tales como HTML
- directivas JSP tales como include
- elementos de scripting JSP
- acciones JSP
- tags perzonalizadas

Las directivas JSP controlan como el compilador genera el servlet.

Los tags ***<%!*** ... ***%>*** permiten incluir codigo java.

Idealmente no deben haber reglas de negocio en el JSP, solo poco codigo.

Ejemplo

```
<HTML>
<BODY>
Hola! La hora actual es <%= new java.util.Date() %>
</BODY>
</HTML>
```

JSP Modelo 2 (MVC)

La nueva version de JSP incluye nuevas caracteristicas para mejorar la productividad del programador. Tales como un lenguage de expresion **EL** que permite entre otras cosas crear templates de estilo Velocity y acceder a componentes Java (JavaBeans) desde JSP, ademas de una manera mas rapida de mostrar parametros, y una manera mas clara de navegar beans anidados. Ademas, SUN propone utilizar el modelo-vista-controlador MVC (model, view, controller).

Expression Language EL

Al incorporar el lenguaje EL, se permite el fácil acceso a componentes Java (JavaBeans), desde JSP. De esta manera, el diseñador-programador solo necesita saber como hacer las llamadas apropiadas a los metodos del nucleo Java, disfrutando de la facilidad de un verdadero lenguaje de scripting.

Struts

Struts es una herramienta de soporte para el desarrollo de aplicaciones Web bajo el patrón **MVC** bajo la plataforma J2EE (Java 2, Enterprise Edition). Struts se desarrollaba como parte del proyecto Jakarta de la Apache Software Foundation, pero actualmente es un proyecto independiente conocido como Apache Struts.

El objetivo de Struts es separar el modelo (logica de la aplicación que interactúa con una base de datos) de la vista (paginas HTML presentadas al cliente), y el controlador (pasa información entre la vista y el modelo).

Nota: Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.

Struts provee el controlador (un servlet conocido como **ActionServlet**) y facilita la escritura de templates para la capa de vista o presentacion (tipicamente en JSP, pero XML/XSLT y Velocity también se pueden usar).

El programador de la aplicación web es responsable de escribir el codigo del modelo, y crear una configuracion central en el archivo **struts-config.xml** que une al modelo, la vista y el controlador.

Las peticiones de un cliente son enviadas al controlador en forma de "**Acciones**" definidas en el archivo de configuracion. El controlador llama a la clase de la acción correspondiente que interactúa con el codigo especifico del modelo.

El codigo del modelo devuelve un "**ActionForward**", un string diciendole al controlador que pagina devolver al cliente.

La información se pasa entre model y vista en forma de JavaBeans especiales.

Una librería de tags perzonalizadas permite leer y escribir el contenido de esos beans desde la capa de presentacion sin necesidad de codigo java embebido.

Struts ademas soporta internacionalizacion, y tiene validacion de datos enviados en web forms, ademas de tener un mecanismo de templates.

Ejemplo Struts

Para el manejo de formas, Struts provee una clase llamada **ActionForm** que se debe extender para darle funcionalidad. Dentro de la subclase, por cada campo dentro de la forma, se debe declarar una variable instancia con el mismo nombre que se asignó al atributo name en el código HTML, para cada variable se deben escribir métodos **get** y **set**.

Adicionalmente se pueden sobrescribir los métodos **reset** y **validate** que tienen funcionalidades específicas. El método **reset** es invocado para reinicializar las variables a un valor dado por el programador, mientras el método **validate** se usa para validar que los datos no tengan errores de captura o valores nulos.

Struts implementa el controlador mediante la clase **ActionServlet** del paquete **org.apache.struts.action,** es importante aclarar que esta clase es concreta, es el servlet principal de nuestra aplicación y no se necesita extender para poder usar el framework.

Toda aplicación Web debe tener un descriptor de despliegue (conocido en inglés como *deployment descriptor*) llamado **web.xml**, en este archivo se debe especificar el uso de Struts.

El desarrollador indica al framework las acciones, las formas y otros atributos mediante un archivo de configuración que por convención se llama **struts-config.xml.**

Para crear una acción se debe extender la clase **org.apache.struts.action.Action** y sobrescribir el método **execute.**

Para correr la aplicación se deben compilar las clases y copiar los paquetes generados al directorio **classes** dentro de **WEB-INF**, por otra parte se deben copiar también las páginas o JSPs encargadas de la vista a la carpeta raíz, una vez tenida la estructura, se procede a copiar la carpeta de la aplicación al directorio de nuestro contenedor de Servlets.

JavaScript

JavaScript es un lenguaje de programación interpretado, es decir, que no requiere compilación, utilizado principalmente en páginas web, con una sintaxis semejante a la del lenguaje Java y el lenguaje C.

JavaScript NO es una version simplificada de Java, su nombre es una movida comercial entre Sun y Netscape, pero son dos lenguajes totalmente diferentes.

JavaScript es un lenguaje orientado a objetos propiamente dicho, ya que dispone de herencia, si bien esta se realiza siguiendo el paradigma de programación basada en prototipos, ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Todos los navegadores modernos interpretan el código JavaScript integrado dentro de las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del **DOM**.

Javascript se puede incluir en cualquier documento HTML, o todo aquel que termine traduciéndose en HTML en el navegador del cliente; ya sea PHP, ASP, SVG...

Incluir código directamente en una estructura HTML es una práctica invasiva, y no recomendada. El método correcto que define la W3C es incluir javascript como un archivo externo, tanto por cuestiones de accesibilidad, como practicidad y velocidad en la navegación.

DOM

Document Object Model (DOM) es un modelo orientado a objetos a través del cual los programas y scripts pueden acceder y modificar dinámicamente el contenido, estructura y estilo de los documentos HTML y XML.

El DOM define la manera en que objetos y elementos se relacionan entre sí en el navegador y en el documento. Cualquier lenguaje de programación adecuado para el desarrollo de paginas web puede ser utilizado.

En el caso de javascript, cada objeto tiene un nombre, el cual es exclusivo y único. Cuando existen más de un objeto del mismo tipo en un documento web, estos se organizan en un vector.

Ejemplo Javascript

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
<head><title>Hola Mundo</title></head>
	<body>
		<script type="text/javascript">
		document.write('Hola Mundo!');
		</script>
	</body>
</html>
```

Ajax

AJAX, Asynchronous JavaScript And XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications).

Estas aplicaciones se **ejecutan en el cliente**, es decir, en el navegador de los usuarios mientras se mantiene la **comunicación asíncrona** con el servidor en segundo plano.

De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página.

JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales.

En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM).

AJAX es una combinación de cuatro tecnologías ya existentes:

- **XHTML** (o **HTML**) y hojas de estilos en cascada (**CSS**) para el diseño que acompaña a la información.
- Document Object Model (**DOM**) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.
- El objeto **XMLHttpRequest** para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.
- **XML** es el formato usado generalmente para la transferencia de datos solicitados al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

Ejemplo AJAX

AJAX utiliza XML para transmitir los datos del servidor al cliente.

Para que el contenido del documento XML sea reconocido como tal por el cliente es necesario que desde el servidor se especifique `Content-Type: text/xml` *como encabezado, y como los datos que el servidor manda al cliente se habrán generado de forma dinámica, se agregan encabezados especiales :* `Cache-Control: no-cache, must-revalidate` *y* `Expires: Mon, 01 Jan 2007 01:00:00 GMT.`

Encabezado XML en JSP

```
<%!
response.setHeader("Content-Type", "text/xml");
response.setHeader("Expires", "Mon, 01 Jan 2007 01:00:00 GMT");
response.setHeader("Cache-Control", "must-revalidate");
response.setHeader("Cache-Control", "no-cache");
%>
```

XMLHttpRequest

XMLHttpRequest (XHR), también referida como XMLHTTP (Extensible Markup Language / Hypertext Transfer Protocol), es una interfaz empleada para realizar peticiones HTTP y HTTPS a servidores WEB.

Para los datos transferidos se usa cualquier codificación basada en texto, incluyendo: texto plano, XML, JSON, HTML y codificaciones particulares específicas.

La interfaz se presenta como una clase de la que una aplicación cliente puede generar tantas instancias como necesite para manejar el diálogo con el servidor.

El uso más popular, si bien no el único, de esta interfaz es proporcionar contenido dinámico y actualizaciones asíncronas en páginas WEB mediante tecnologías construidas sobre ella como por ejemplo AJAX.

La interfaz se presenta encapsulada en una clase.

Para utilizarla, la aplicación cliente debe crear una nueva instancia mediante el constructor adecuado.

Es posible realizar peticiones síncronas y asíncronas al servidor; en una llamada asíncrona el flujo de proceso no se detiene a esperar la respuesta como se haría en una llamada síncrona, si no que se define una función que se ejecutará cuando se complete la petición: un manejador de evento.

JBoss

JBoss es un servidor de aplicaciones J2EE de código abierto implementado en Java puro.

Al estar basado en Java, JBoss puede ser utilizado en cualquier sistema operativo que lo soporte, e implementa todo el paquete de servicios de J2EE.

JBoss AS es el primer servidor de aplicaciones de código abierto, preparado para la producción y certificado J2EE 1.4, disponible en el mercado, ofreciendo una plataforma de alto rendimiento para aplicaciones de e-business. Combinando una arquitectura orientada a servicios revolucionaria con una licencia de código abierto, JBoss AS puede ser descargado, utilizado, incrustrado, y distribuido sin restricciones por la licencia. Por este motivo es la plataforma más popular de middleware para desarrolladores, vendedores independientes de software y, también, para grandes empresas.

Las características destacadas de JBoss incluyen :

- Producto de licencia de código abierto sin coste adicional.
- Cumple los estandares.
- Confiable a nivel de empresa
- Incrustable, orientado a arquitectura de servicios.
- Flexibilidad consistente
- Servicios del middleware para cualquier objeto de Java
- Ayuda profesional 24x7 de la fuente
- Soporte completo para JMX

JBoss IDE

Brinda una IDE Eclipse para el JBoss AS. De esta forma la depuración y otras tareas asociadas al desarrollo de aplicaciones puede ser realizadas desde el entorno de Eclipse.

Deployment en JBoss

[debug sin revisar/ terminar , investigar mejor!!]

El punto de partida de todo desarrollador de aplicaciones web es la creación de un documento root o directorio raíz de aplicación que va a contener a todos sus componentes.

Este directorio raíz colgará del directorio raíz de todas las aplicaciones web del servidor J2EE de trabajo.

En el caso de Tomcat, `tomcat_home\webapps`.

En el caso de JBoss, `jboss_home\server\default\deploy`

Hecho esto, el desarrollador deberá

- Situar los ficheros html, jsp, xml, txt, imágenes, sonidos, etc. colgando directamente del directorio raíz de su aplicación o de subcarpetas del mismo
- Crear un subdirectorio WEB-INF colgando del raíz de la aplicación y almacenar en él, el fichero descriptor de despliegue web.xml.
- Este directorio contiene los recursos privados de la aplicación. No son accesibles directamente desde el cliente, en el sentido de que no puede descargárselos y ver su código.
- Crear un subdirectorio WEB-INF\classes y almacenar en él los class de los servlets compilados y otras clases de apoyo. Debe tenerse en cuenta que si están empaquetados, los class se ubicarán en subdirectorios adecuados atendiendo a sus estructuras de paquete.
- Crear un subdirectorio WEB-INF\lib y almacenar en él librerías de clases en forma de ficheros jar que la aplicación web usa. Típicamente drivers de BDs y librerías de etiquetas.
- Además, suele crearse una carpeta de nombre src (source) colgando del raíz de la aplicación donde se ubican los códigos fuente de los servlets, en el caso de que el desarrollador quiera hacerlos públicos

Tomcat

Tomcat (también llamado Jakarta Tomcat o Apache Tomcat) es un **contenedor de servlets** utilizado como la implementación de referencia oficial para las tecnologías de JavaServer Pages y Java Servlet. **Es el contenedor por defecto que emplea JBoss.**

Tomcat es un servidor web con soporte de servlets y JSPs. Tomcat no es un servidor de aplicaciones, como JBoss o JOnAS. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java

Relación entre JBoss y Tomcat

El contenedor servlet es la tecnologia fundamental de J2EE para la web. Es considerada un contenedor debido a que los JSPs y servlets no pueden correr como aplicaciones stand-alone.

Deben implementar interfaces especificas y correr "dentro" del contenedor.

El contenedor escucha solicitudes HTTP en un puerto dado, pasa esta informacion al componente y usa la salida del mismo para crear un documento HTML que se le devuelve al cliente como respuesta.

Los desarrolladores de JBoss decidieron no reinventar la rueda y crear un nuevo contenedor servlet, y en su lugar permiten integrar un contenedor ya existente.

El contenedor por defecto de JBoss es Tomcat, y es instalado como un SAR en `$JBOSS_HOME/server/default/deploy/jbossweb-tomcat55.sar`

Gracias al diseño modular de JBoss, es sencillo cambiar Tomcat por otro contenedor web.

Enlaces relacionados

JSP

http://www.jsptut.com/Index.html#contents

Struts

http://www.myjavaserver.com/~aldosolis/tutorial_struts/struts_tutorial.html

Eclipse y JBoss

http://www.programacion.net/java/articulo/jap_eclip_3/

http://www.adrformacion.com/cursos/javaser/leccion3/tutorial2.html