Asignatura: Programación Web

Conferencia #4 Aplicaciones empresariales. Tecnología empresarial Java Enterprise Edition. Introducción a Java Server Pages.

Objetivos:

- Conocer el concepto de aplicaciones empresariales
- Introducir Java Server Pages.

Contenido:

- Introducción.
- Protocolos y métodos de comunicación
- Aplicaciones Web con Java
- Introducción a JSP
- Conclusiones.

Bibliografía:

1. Geary, David; Horstmann Cay. "Core JavaServer™ Faces, Second Edition". Prentice Hall, 2007. ISBN: 978-0-13-173886-7. Capítulos 1.
2. Deitel, P; Deitel, H. "AJAX, Rich Internet Applications, and Web Development For Programmers". Pearson Education, Inc, 2008. Capítulo 22.
3. Dudney, B et al. "Mastering in JavaServer™ Faces". Wiley Publishing, Inc, 2004. Capítulo1.

Introducción

En la actualidad existe una tendencia cada vez más fuerte de desarrollar aplicaciones distribuidas transaccionales para las empresas y por consiguiente aumentar el rendimiento, seguridad y confiabilidad de la tecnología del lado del servidor. Cuando se aborda el término empresa nos referimos a una organización económica y las aplicaciones de la empresa son aquellas aplicaciones de software que facilitan diversas actividades dentro la misma.

HTTP

- HTTP es un protocolo simple y sin estado, usado para la comunicación entre las diferentes aplicacicones en la web, esto significa que el servidor no puede mantener información alguna sobre el cliente luego de enviar su respuesta, e inclusive no puede saber si diferentes acciones desde un mismo cliente están relacionadas.
- Cuando un cliente (por ejemplo desde el firefox) hace una llamada (pedido [request]) al servidor, este responde y la transacción es terminada.

Métodos para realizar llamadas

- Get: Se basa en el uso de una URL que incluye (opcionalmente) una cadena de parámetros en un cuerpo, o sea los parámetros son visibles al usuario en la barra de direcciones del navegador.
  http://www.google.com.cu/search?q=jsp
- Post: Es similar al get, y aunque puede enviar parámetros en la URL, estos se envían generalmente en el cuerpo de la llamada y son invisibles al usuario.

- Otros métodos usados, aunque no muy comúnmente son los siguientes: OPTIONS, HEAD, PUT, DELETE, TRACE.

Aplicaciones web con java

Se entiende por una aplicación Web con java a una colección de servlets, páginas html, clases y otros recursos (estáticos o no) que pueden ser empaquetados y ejecutados en diferentes contenedores de distintos proveedores, o sea es informalmente cualquier cosa que resida en la capa web de una aplicación.
Todas las aplicaciones web tienen solamente un contexto de ejecución de los servlets (ServletContext) controlado por contenedor de servlets garantizando que entre ellas no puede haber colisiones ni visibilidad directa.

Estructura de una aplicación web con java

Una aplicación web está usualmente contenida dentro del contenedor de servlets, a partir de un directorio raíz en el servidor (SERVER_ROOT), en tomcat es /webapps y deben tener la estructura de directorios siguiente:

Directorio	Descripción
`/miapp`	Esta es la carpeta de inicio de de la aplicación. Usualmente los archivos JSP, html, etc se colocan aqui o en carpetas que se crean en este directorio.
`/miapp/WEB-INF`	Aqui se incluyen todos los recursos que no se colocan en el directorio raíz. Aquí es donde se ubica el descriptor de desplieque de la aplicación (web.xml). Se ddebe tener en cuenta que este no es un directorio público por lo que ningun recurso que se situe dentro de esta carpeta no se puede mostrar directamente al usuario.
`/miapp/WEB-INF/classes`	Aqui se almacenan las clases de utilidades de la aplicación Tiene precedencia respecto a`WEB-INF/lib` en caso de existir alguna clase con igual nombre calificado.
`/miapp/WEB-INF/lib`	Aqui están todos los archivos y/o bibliotecas de los cuales depende la aplicación para su ejecución adecuada.

Descriptor de despliegue

El descriptor de despliegue es un documento xml llamado web.xml ubicado en la carpeta /<SERVER_ROOT>/aplicación/WEB-INF/. Este describe la información de la configuración para toda la aplicación web. Este documento incluye los siguientes elementos:

* Parámetros de inicialización de la aplicación (context-param)
* Configuración de la sesión
* Definiciones de los Servlets / JSPs
* Mapeos de los Servlets / JSPs

* Mapeos de tipos MIME (Extensiones de correo Internet Multipropósito)
* Lista de archivos de inicio
* Páginas de error
* Seguridad

Ejemplo

```
<web-app>
  <display-name>Mi App</display-name>
  <session-timeout>30</session-timeout>
  <servlet>
    <servlet-name>TestServlet</servlet-name>
    <servlet-class>com.test.TestServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
    <init-param>
      <param-name>name</param-name>
      <param-value>value</param-value>
    </init-param>
  </servlet>
</web-app>
```

Java Server Pages

Un JSP es técnicamente un servlet aunque su estructura está orientada a ser usado mediante la combinación con código html, siendo este una pieza de código que añade, extiende o mejora una funcionalidad del servidor donde se hospeda (el contenedor de servlets). Apache Tomcat por ejemplo es la implementación de referencia para las especificaciones de servlets y jsp. Esta es la alternativa más difundida que usa java a CGI, ASP, etc.

Contextos y aplicaciones web

Una aplicación web en java está compuesta por diferentes recursos: jsp, servlets, html, librerías.
Dentro del contenedor web cada una de las aplicaciones que residen en el mismo está representadas por el contexto de ejecución de los servlets que a su vez está asociados con un único prefijo URI de acceso, que no es más que el camino o la dirección del contexto.
Cada uno de los contextos existentes en un servidor web son auto contenidos, o sea tienen todos los elementos necesarios para su ejecución satisfactoria, y no tienen conocimiento alguno de las otros existentes en el mismo.

Estructura de un archivo JSP

```
<%@ page language="java" contentType="text/html; charset=UTF-8“
        pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
                                    "http://www.w3.org/TR/html4/loose.dtd">
```

```
<html>
<head></head>
<body>
        <jsp:useBean id="userName" class="ejemplo.UserNameBean">
        <jsp:setProperty name="userName" property="userName" value="nombre" />
        <jsp:setProperty name="userName" property="userId" value="ident" />
        </jsp:useBean>
<ul>
        <li>Nombre: <jsp:getProperty name="userName" property="userName" /></li>
        <li>id: <jsp:getProperty name="userName" property="userId" /></li>
</ul>
</body>
</html>
```

Nota: Un archivo JSP no está limitado a contener un documento html/xml/etc. El mismo admite contener cualquier tipo de documento para luego ser procesado.

Procesamiento de las páginas JSP

Cuando se invoca una página JSP, primeramente el servidor web llama al contenedor de JSP para que verifique si la página ha sufrido alguna modificación desde la última llamada hecha a la misma. En caso de que se haya modificado entonces se procede a la conversión de la misma en un servlet de la siguiente manera:
Toda la parte de texto estática (la plantilla) es convertida a llamadas a la función println( ) y las instrucciones JSP son convertidas en el código java que implementa la funciones dinámicas que estas describen.
Luego de realizar la conversión el contenedor de JSP compila la clase generada.
A causa de esto es posible que en la primera llamada que se hace a un archivo JSP se note alguna demora, que no sucedería en llamadas subsiguientes.

Elementos directivos	
Elemento	Descripción
<%@ page ...%>	Define los attributos que son dependientes de la página tales como la administración de las sesiones, páginas de error, etc.
<%@ include ... %>	Incluye un archivo durante la fase de traducción.
<%@ taglib ...%>	Declara una librería de etiquetas, que contiene acciones ya preestablecidas, de manera que estas pueden ser usadas en la página.

Elementos de acción	
Elemento	Descripción
<jsp:useBean>	Hace el componente JavaBeans sea accesible en la página
<jsp:getProperty>	Obtiene una propiedad del componente especificado y la añade en la respuesta.
<jsp:setProperty>	Actualiza una propiedad del componente JavaBean con un valor dado.
<jsp:include>	Incluye el resultado (respuesta) de un servlet o una página JSP durante la fase de ejecución de la página. Este es unha alternativa al uso de la directiva <%@ include ... %> , pero aqui no se incluye directament el archivo sino el resultado de su invocación, o sea aqui es visible la respuesta no el contenido del archivo.
<jsp:forward>	Pasa el control del procesamiento del pedido a otra página JSP o servlet.
<jsp:param>	Añade un nuevo parametro (con su valor) al pedido que es transferido a otro servlet vía <jsp:include> o <jsp:forward>

Código

Elemento	Descripción
<% ...%>	Se usa para inscrustar código ejecutable
<%= ...%>	Incluye el resultado de la evaluación de una expresión.
<%! ...%>	Declara variables de instancias y métodos en la página
<%-- --%>	Declara un comentario para la página JSP

```
<%
    for (int j = 0; j <5; j++) {
        out.print(j);
    }
%>
<%= 10+20 %>
<%-- Esto es un comentario --%>
<%!
  int globalCounter = 0; // Variable de instancia
%>
<%
  int localCounter = 0;   // Variable local
%>
```

Variables implícitas

Nombre	Descripción
Page	(javax.servlet.jsp.HttpJspPage) Refencia a una pagina JSP, se usa para llamar a cualquier instancia de las páginas
Config	(Javax.servlet.http.ServletConfig) Se usa para obtener información acerca de la configuración del servlet.
Application	(Javax.servlet.http.ServletContext) Usado principalmente para intercambiar datos entre todas las paginas de la aplicación.
PageContext	(Javax.servlet.jsp.pagecontext) Se usa para acceder a los atributos y espacios de nombre asociados a la página.
Out	(Javax.servlet.jsp.JspWriter) Se usa para imprimr directamente a la salida dentro de la paina JSP.
Session	(Javax.servlet.http.httpsession) Habilita mantener el estado de variables (objetos) entre diferentes llamadas a paginas (no tienen que ser las mismas) siempre y cuando se trate de la misma sesion de usario.
response	(Javax.servlet.http.httpservletresponse) Contiene la información que resulta de la ejecución de una llamada al servidor (pagina, servlet, etc). Generalmente usado en conjunto con las cookies y encabezados HTTP.
request	(Javax.servlet.http.httpservletrequest) Contiene los datos (parámetros, encabezados, cookies, etc) que son enviados al servidor en cada una de las llamadas hechas al mismo.

Visibilidad

- Página: El dato es accesible solo dentro de la página JSP y se destruye cuando esta ha terminado de generar su salida para la llamada realizada.
- Llamada: Los datos son visibles hasta que el procesamiento de la llamada se completa.
- Sesión: Los datos se pueden acceder mientras la sesión del usuario esté activa.
- Aplicación: Son visibles entre todas las sesiones abiertas por diferentes usuarios dentro de una misma aplicación.

Tratamiento de errores

Archivo JSP estándar

```
<%@ page import="java.util.*" errorPage="error.jsp"%>
<body>
        <%= 0/0 %>  <- Aquí se produce una división por cero que lanza una excepción del tipo java.lang.ArithmeticExceptiony provoca se llame al archivo definido en errorPage de la directiva page
</body>
```

Archivo JSP para tratamiento de errores

```
<%@ page language="java"  isErrorPage="true" %>
                                ^_          Tiene definido el atributo isErrorPage a true, lo que habilita que la variable predefinida exception sea
                                            accesible, para hacer el tratamiento de error.
  <%= exception.toString() %><br><br>
  <%
     StackTraceElement ste[] = exception.getStackTrace();
```

```
    for(int i = 0; i < ste.length; i++)
      out.println(ste[i] + "<br>");
  %>
```

JSTL

JSTL es una biblioteca que implementa funciones de uso frecuente en aplicaciones JSP, agrupadas en 5 bibliotecas distintas:

- Funciones comunes de iteración sobre datos, operaciones condicionales, e importación de otras páginas.
- Internacionalización y formateo de texto.
- Funciones de manipulación de cadenas.
- Procesamiento de XML.
- Acceso a bases de datos.

Además proporciona un lenguaje de expresión para referenciar objetos y sus propiedades sin necesidad de código java, y validadores de bibliotecas de etiquetas o TLV.

Conclusiones

En la conferencia se analizaron las características de las aplicaciones Java; se definió la plataforma Java EE como la solución estándar de Java para desarrollar aplicaciones de este tipo. Se introdujo la tecnología JSP como estándar de Java EE para implementar aplicaciones web empresariales.

Motivación para la próxima clase

En la próxima clase se continuará el estudio de la tecnología JSP.