Arquitectura de los Servlets

Introducción

Que es un Servlet

Es parte de la tecnología Java junto con JSP, y pertenece a la Edición Empresarial (J2EE)

Es una unidad de funcionalidad que se ejecuta del lado del servidor, y genera resultados que son enviados al cliente

Debe ser desplegado dentro de un Servlet Container para su correcto funcionamiento

Arquitectura

La arquitectura es la infraestructura necesaria para poder ejecutar un Servlets

Para el despliegue de un Servlets es necesario contar con un Servlets Container, o también llamado Web Container

El Servlets Container deberá colaborar con el Web Server para determinar el momento de ejecución de cada uno de los Servlets

CGI vs. Servlets

CGI = Common Gateway Interface

Los CGIs fueron una de las primeras tecnologías utilizadas para el proceso de datos en el servidor

Tiene como desventajas que son dependientes de la plataforma y difíciles de integrar en aplicaciones de gran envergadura ya que presentan problemas de escalabilidad

Los Servlets representan una forma de evolución de los CGIs

Tienen la ventaja de ser multiplataforma, integrables con el lenguaje de programación Java, altamente escalables y fácil de integrar en una arquitectura multi-capa

Arquitectura de HTTP

El Web Client

Es el consumidor principal de la arquitectura

Representa a un browser realizando requests (pedidos) al servidor, y recibiendo responses (respuestas)

El Web Server

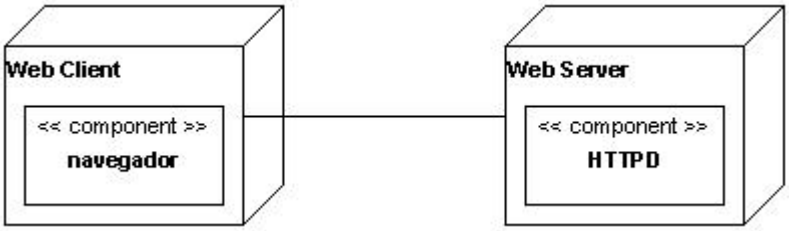
Es el servidor que provee servicios Web

Recibe pedidos de los clientes, y les brinda respuestas

En su carácter más básico, se encarga de servir páginas según la demanda

---

Diagrama



El Servlets container

Que es un ServletsContainer

Es un software que tiene como principal característica la capacidad de resolver la lógica que es necesaria para la utilización de Servlets

Provee el ambiente de despliegue y ejecución para los Servlets

No es un Web Server

Es una aplicación que se instala donde está instalado el Web Server

Se debe configurar para que actúe de forma coordinada con el Web Server

Relación con el Web Server

El ServletContainer se suele instalar “sobre” el Web Server

Los pedidos de los clientes se realizan al Web Server, quien solicita la resolución de Servlets al ServletContainer

Ambos cooperan con el objetivo de enviar una página con información al cliente

Los distintos ServletContainers

En el mercado existen distintas propuestas

El Apache Tomcat es una de las opciones más utilizadas

Existen otras propuestas de gran difusión como ser JRun

En general, los Servlet Containers se pueden agregar a los Applications Servers para que funcionen en conjunto

El Tomcat como ServletContainer

El Tomcat es uno de los ServletContainer más utilizados

Si bien es también un WebServer, no se recomienda su uso en producción

Su uso como WebServer debería ser exclusivamente para desarrollo y testing

Es común instalarlo sobre el Apache WebServer

Arquitectura de un ServletContainer

El Web Client

Ídem Web Client de Arquitectura HTTP

El Web Server

Ídem Web Server de Arquitectura HTTP

El Web Container

Se instala y configura junto con el Web Server

El Web Container tiene la responsabilidad de “conversar” con el Web Server

En conjunto, coordinan la resolución de la funcionalidad y el envío de la página al cliente

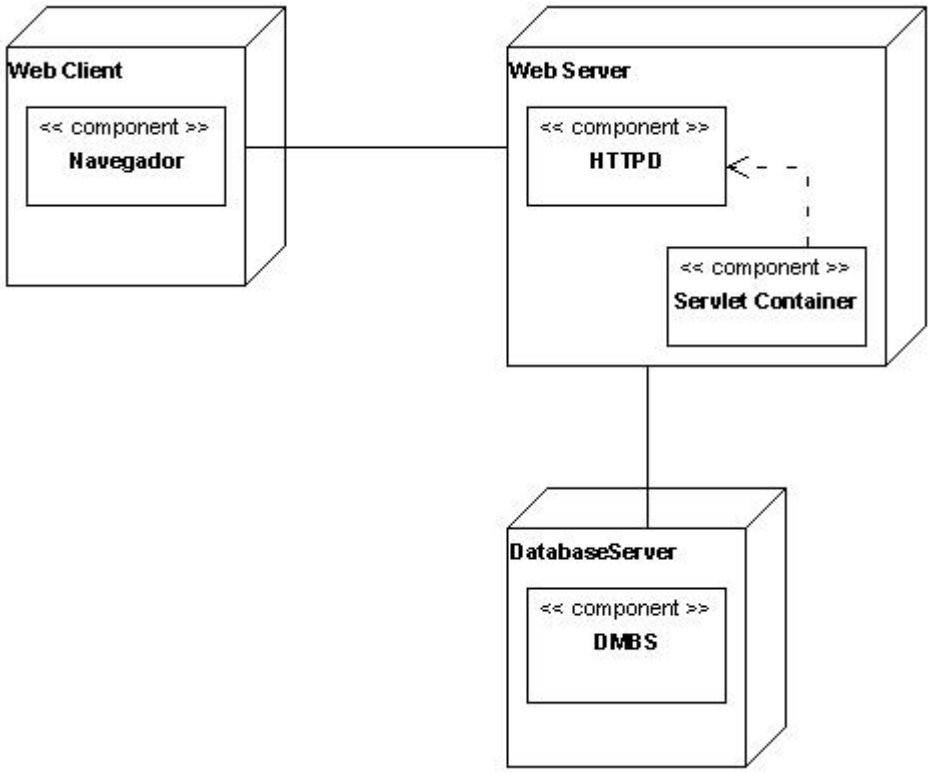
El Database Server

Es el software utilizado para la Administración de Base de Datos

Generalmente está en una máquina independiente

Provee los servicios necesarios al Web Container

Diagrama



Web components

Que es un Web component

Cualquier elemento que forme parte de un sitio Web es considerado un componente Web

Es una unidad de funcionalidad o almacenamiento dentro de un sitio Web

Puede ser un Servlets, JSP, archivos de texto, imágenes, como también otras clases que formen parte del sitio Web

Servlets

Es la propuesta inicial de Java en programación Web

Es una unidad de funcionalidad que se ejecuta del lado del servidor, y genera resultados que son enviados al cliente

Debe ser desplegado dentro de un Servlets Container para su correcto funcionamiento

Java Server Page

Es la propuesta de Java en programación Web posterior a los Servlets

Provee un mecanismo más sencillo de desarrollo

Un JSP es un archivo de texto que se convertirá en un Servlets cuando se compile

La compilación y ejecución es llevada a cabo por el Web Container

Developing Basic Servlets

Fundamentos

El paquete javax.servlet

Es el paquete donde están colocadas las clases e interfaces de la Servlet API

Contiene a los demás paquetes de Servlets más específicos

Es el punto de partida para la construcción de Servlets

La clase GenericServlet

Es la clase básica para la creación de un Servlets

Es el Servlets de carácter más genérico que existe en la API

Cuenta con un único método: service()

Es una clase abstracta, con lo cual es necesario construir una subclase y sobrescribir el método service()

Permite el uso de protocolos que no es el HTTP

Si se desea usar el protocolo HTTP, es necesario utilizar la clase HttpServlet como superclase

El método service()

Es el método a sobrescribir obligatoriamente

Deberá contener las reglas de negocio a realizar

Cuando el cliente hace una llamada al Servlets, el ServletContainer dispara el método service() perteneciente al Servlets que fue llamado

El paquete javax.servlet.http

Es el paquete donde están colocadas las clases e interfaces de la HTTP Servlets API

Contiene los elementos necesarios para la construcción de una aplicación que utiliza el protocolo HTTP

Es el punto de partida para la construcción de una aplicación Web

La clase HttpServlet

Cuando es necesario trabajar dentro de un marco que utiliza el protocolo HTTP, es necesario construir una subclase de HttpServlet

Posee los métodos service(), doGet(), doPost()

El criterio del método service() también es disparado cuando se llama al Servlets, y como valor agregado éste llama el método doPost() o doGet() según corresponda

El método doGet()

Es el método utilizado para procesar los datos que vienen a través del método GET

Debe ser sobrescrito en la subclase de HttpServlet

Si se sobrescribe el método service(), este método deja de tener validez

El método doPost()

Es el método utilizado para procesar los datos que vienen a través del método POST

Debe ser sobrescrito en la subclase de HttpServlet

Si se sobrescribe el método service(), este método deja de tener validez

La clase HolaServlet

Objetivo

Presentar la clase HolaServlet como mecanismo básico de respuesta a una consulta entre el cliente y el servidor

Pasos necesarios

La clase HolaServlet debe ser construida como una subclase de la clase HttpServlet

Se deberán sobrescribir los métodos doGet() y doPost() según corresponda

Se deberá genera una entrada en el deployment descriptor utilizado por el Servlet Container (ver punto 6.1.4)

Comenzar el Web Server + Servlets Container

Ejemplo de uso

```
import javax.servlets.*;

import javax.servlets.http.*;

public class HolaServlet extends HttpServlet
```

```
{
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    // aquí el código
}

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
    // aquí el código
}
}
```

Despliegue de una aplicación Web

Introducción

Definición

Despliegue = “Deployment”

Representa el pasaje de todos los componentes web a un ambiente de ejecución, generalmente producción (aunque podría ser de testing también)

El descriptor de despliegue web.xml

El despliegue de la aplicación queda determinado por un archivo llamado “descriptor de despliegue”

El descriptor de despliegue tiene la información necesaria de configuración y despliegue de la aplicación a utilizar

Está escrito en XML

Servlet Mappings

Definición

Es el mapeo necesario entre el nombre de un Servlets a utilizar y la forma de llamarlo

Para poder utilizar un Servlets, deberá estar presente en el deployment descriptor

Los Servlets mappings se realizan a través de XML dentro del deployment descriptor

El tag <web-app>

Representa a la aplicación Web

Es el tag raíz del descriptor de despliegue

Contiene a todos los tags que determinan la concepción de la aplicación Web

El tag <Servlets>

Se utiliza para representar a los Servlets que forman parte de la aplicación

Contiene dos tags importantes: <servlet-name> y <Servlets-class>

<Servlets-name> se utiliza para identificar el nombre “lógico” del servlet

<Servlets-class> se utiliza para identificar el nombre “físico” del servlet, es decir el nombre que tiene el archivo Servlets junto con sus paquetes

El tag <Servlets-mapping>

Se utiliza para realizar el mapeo entre el Servlets y la forma de llamarlo

Contiene dos tags importantes: <Servlets-name> y <url-pattern>

<Servlets-name> se utiliza para identificar el nombre “lógico” del Servlets

<url-pattern> se utiliza para identificar la forma de llamar al Servlets

Ejemplo de un deployment descriptor

<web-app>

<servlet>

<servlet-name>HolaServlet</servlet-name>

<servlet-class>paq.HolaServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>HolaServlet</servlet-name>

<url-pattern>/HolaServletDesdeUrl</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>

index.jsp

</welcome-file>

</welcome-file-list>

</web-app>

Creación del ambiente de despliegue

La carpeta webapps

El Servlet Container puede lidiar con más de una aplicación

webapps representa las palabras “Web Applications”, o aplicaciones Web

La carpeta webapps contiene las distintas aplicaciones Web que podrán utilizarse, cada una de ellas en una carpeta distinta

Es la carpeta de mayor nivel en la jerarquía dentro del Servlet Container

La carpeta WEB-INF

Representa el corazón de la aplicación Web en cuanto a las reglas de negocio

Contiene el archivo de configuración web.xml

Contiene el directorio correspondiente a las clases

La carpeta classes

Deberá contener todos los archivos .class

Cada clase compilada deberá estar ubicada en el directorio correspondiente al paquete que la contiene

La carpeta lib

Deberán estar ubicados todas las librerías que se utilizan en el proyecto, es decir los archivos .jar

Despliegue a través de un archivo .war

El contenido de un .war

Deberá contener todos los Web components de la aplicación Web

Entre ellos se incluyen archivos de texto, imágenes y archivos de audio, como también las clases compiladas necesarias para su correcta ejecución

Ubicación de archivos .war

War = Web Archive

El archivo .war contiene todas los Web components necesarios para ejecutar la aplicación

La ventaja que tiene es que resulta más fácil desplegar una aplicación ya que se necesita de un único archivo

Se puede ver como la versión Web del archivo ".jar"

La aplicación "MiAplicacion" podría construirse dentro de un archivo .war con todo su contenido, y copiarse al directorio webapps

Despliegue de múltiples aplicaciones

Organización

En el Servlet Container se puede desplegar más de una aplicación web, donde cada una de ellas deberá estar contenida en un directorio diferente

Todas las aplicaciones deberán estar contenidas en el directorio webapps, por ejemplo:

webapps/

ROOT/ (aplicación base)

Admin/ (cuestiones de administración)

Ecommerce/ (sitio de e-commerce)

Eportal/(otro portal)

Dentro de cada aplicación, existe también una estructura de directorios a respetar

Dicha estructura es la siguiente:

webapps/

MiAplicacion/

index.jsp (podría tener otra extensión)

WEB-INF

web.xml

classes/

lib/

Dentro de la carpeta MiAplicacion estarán ubicados los archivos html y jsp

Developing Data-Processing Servlets

El formulario y sus componentes

El tag <FORM>

Representa el formulario para registración de datos

Es el tag responsable de contener a otros tags que servirán para enviar información al servidor

Para más información, ver detalle del tag <FORM> en el capítulo de HTML

Envío de datos al servidor

Cuando se presiona un boton correspondiente al submit, la información que forma parte del formulario es enviada al servidor

Dicha información es previamente empaquetada y señalizada por el método en que se envía: método GET o método POST

La propiedad “action” del tag <FORM> representará el recurso del servidor que se encargará de procesar los datos enviados

La interfaz HttpServletRequest

Introducción

La interfaz HttpServletRequest tiene los métodos necesarios para obtener la información enviada por el cliente a través de HTTP

Tanto el método doGet() como el método doPost() reciben un objeto llamado request como uno de sus parámetros, el cual posee todos los métodos de la interfaz

El objeto request es generado por el Servlet Container, y almacena los valores recibidos del cliente

Es una forma sencilla de obtener datos del cliente, de forma independiente al método utilizado (es decir, GET o POST)

El método getParameter()

Utilizado para obtener el valor de un parámetro enviado por el cliente vía GET o POST

La firma del método es: String getParameter(String s)

Si el dato enviado está en una caja de texto denominada txtNombre, entonces la forma de obtener ese dato será:

String elDato = request.getParameter(“txtNombre”);

El método getParameterNames()

Utilizado para obtener los nombres de todos los datos enviados por el cliente vía GET o POST

La firma del método es: Enumeration getParameterNames()

La forma de utilizarlo es:

Enumeration losNombres = request.getParameterNames();

El método getParameterValues()

Utilizado para obtener los valores de un componente del formulario que puede enviar más de un dato, por ejemplo una lista con múltiples selecciones

La firma del método es: String[] getParameterValues(String s)

La forma de utilizarlo es:

String[] losValores = request. getParameterValues (“cmbItems”);