DESPLIEGUE DE APLICACIONES WEB UD-3

CONCEPTOS Y EJEMPLOS

ÍNDICE

[CGI (COMMON GATEWAY INTERFACE) 3](#_Toc150525777)

[JAVA EE (JAVA ENTERPRISE EDITION) 4](#_Toc150525778)

[A. SERVLETS 5](#_Toc150525779)

[B. JSP (JAVASERVER PAGES) 6](#_Toc150525780)

[C. SERVLET CONTAINER 6](#_Toc150525781)

[D. DESCRIPTORES DE APLICACIÓN (WEB.XML) 7](#_Toc150525782)

[E. RECURSOS ESTÁTICOS 9](#_Toc150525783)

[F. FILTROS 9](#_Toc150525784)

[G. RECURSOS DE DATOS 9](#_Toc150525785)

[H. PÁGINAS DE ERROR 9](#_Toc150525786)

[COMPRESIÓN DE PROYECTOS JAVA. ARCHIVOS .WAR 10](#_Toc150525787)

[A. CREACIÓN DE UNA APLICACIÓN 11](#_Toc150525788)

[JSON (JAVASCRIPT OBJECT NOTATION) 12](#_Toc150525789)

# CGI (COMMON GATEWAY INTERFACE)

Es un tipo de implementación que permite al usuario ejecutar aplicaciones de servidor desde su navegador a través del protocolo HTTP.

En este caso los scripts se encuentran en el servidor y son llamados por el cliente a través del navegador, son ejecutados en el mismo servidor y reenvían la información de vuelta al usuario.

Diagrama

Descripción generada automáticamente

Para llamar a un programa CGI debemos escribir una URL tipo:

http://www.servidor.com/cgi-bin/programa1

Además, se puede enviar información por medio de un método GET o POST siendo el primero más recomendable para las solicitudes cortas, simples y rápidas y el POST para aquellas que requieran una mayor seguridad o un mayor envío de información.

Por ejemplo:

http://www.servidor.com/cgi-bin/programa1?param1=valor1&param2=valor2

En este caso:

1. La parte cgi-bin es fija y está definida en el fichero de configuración del servidor web.
2. Esta parte le dice al servidor que lo que viene después es una llamada a un ejecutable (programa1) el cual recibe dos parámetros por método GET (param1 y param2).
3. El servidor ejecuta dicho programa y devuelve el resultado al navegador. Este resultado se devuelve en un formato entendible por el navegador (HTML) y debe ser el propio programa el que se encargue de generar dicho formato.

Este tipo de implementación la usan lenguajes como PHP o Pearl:

Texto

Descripción generada automáticamente

# JAVA EE (JAVA ENTERPRISE EDITION)

Diagrama

Descripción generada automáticamenteSi creamos un software por ejemplo de facturación o de inventarios aparte de escribir el código que realiza la facturación o los inventarios hay que agregarle otras cosas igual de importantes y necesarias. Estas otras cosas que agregamos son funcionalidades de seguridad, comunicación de red, logging, integridad de datos, etc.

Cada aplicación que creamos debe tener todas estas funcionalidades, así que la idea detrás de Java EE es reunirlas y ofrecerlas en un conjunto con el objetivo de que el desarrollador se concentre en lo que realmente le interesa que sería la lógica de negocios de su aplicación de facturación, inventarios, recursos humanos, etc.

Eso es Java EE, un soporte para desarrollar aplicaciones empresariales que ofrece todos estos servicios de seguridad, comunicación, logging, integridad, etc.

Un proyecto sencillo de Java EE puede constar de varios componentes esenciales que permiten crear una aplicación web básica.

Estos componentes clave son los siguientes:

## SERVLETS

Un servlet en Java es un programa informático que se ejecuta en un servidor web y se utiliza para procesar las solicitudes de los clientes (navegadores web). Los servlets son especialmente útiles para generar contenido dinámico en páginas web, como formularios, datos de bases de datos o interacciones en tiempo real, y se escriben en Java.

Por ejemplo:

1. Servlet que muestra una página web con el texto “Hola mundo”.

Texto

Descripción generada automáticamente

1. Servlet que genera un formulario HTML que solicita un nombre por pantalla.

Texto

Descripción generada automáticamente

1. Servlet que realiza una redirección a otra página.

Texto

Descripción generada automáticamente

## JSP (JAVASERVER PAGES)

JSP es una tecnología de Java EE que permite la creación de páginas web dinámicas. Los JSP combinan contenido estático (HTML) con código Java embebido para generar páginas web dinámicas y permitir la presentación de datos.

Por ejemplo, un JSP de una página que muestre un “Hola mundo” sería el siguiente:

Texto

Descripción generada automáticamente

En este ejemplo:

1. La página JSP está delimitada por las etiquetas `<%` y `%>`, que permiten incluir código Java embebido en la página.
2. El código Java dentro de las etiquetas `<%` y `%>` se ejecuta en el servidor cuando se solicita la página JSP. En este caso, se utiliza el objeto `out` para imprimir el mensaje "Hola Mundo desde JSP" en la página.
3. El resultado de la ejecución del código Java se incrusta en el HTML de la página.

En este ejemplo, el mensaje "Hola Mundo desde JSP" se mostrará en un encabezado (`<h1>`) cuando se acceda a la página.

1. El resto del código HTML se trata como contenido estático y se renderizará tal como está en el navegador.

Cuando un usuario accede a esta página JSP a través de un navegador web, verá el mensaje "Hola Mundo desde JSP" en el encabezado de la página.

## SERVLET CONTAINER

El servlet container (contenedor de servlets) es el entorno en el que se ejecutan los servlets. Proporciona una infraestructura para gestionar ciclos de vida de servlets, administración de solicitudes y respuestas, y control de hilos.

## DESCRIPTORES DE APLICACIÓN (WEB.XML)

El descriptor de aplicación, generalmente llamado "web.xml," es un archivo de configuración XML que contiene información sobre la aplicación web. Define la configuración de servlets, filtros, páginas de error, seguridad y otras características de la aplicación.

Un descriptor de aplicación Java, también conocido como "archivo de descriptores de implementación" o "deployment descriptor", es un archivo XML utilizado en el desarrollo de aplicaciones web en Java.

Este archivo proporciona información de configuración y metadatos esenciales que describen cómo se debe implementar y ejecutar una aplicación Java en un contenedor de servlets, como Apache Tomcat. El descriptor de aplicación Java es una parte clave de la especificación Java EE (Enterprise Edition) y se utiliza para configurar componentes de la aplicación, como servlets, filtros, escuchadores y recursos.

El descriptor de aplicación Java (generalmente llamado "web.xml" en proyectos web) contiene información como:

1. **Definición de servlets y mapeo de URL**: Especifica los servlets en la aplicación y cómo se asocian a las URL. Esto incluye el nombre de la clase Java del servlet, la URL de asignación y otros parámetros relacionados.
2. **Filtros**: Define filtros que se utilizan para preprocesar o posprocesar las solicitudes y respuestas web antes de que lleguen a los servlets.
3. **Escuchadores**: Configura escuchadores que se utilizan para detectar eventos de la aplicación, como la inicialización y destrucción de la aplicación.
4. **Recursos y contextos de recursos**: Describe los recursos de la aplicación, como bases de datos o archivos JNDI, que pueden ser accedidos por la aplicación.
5. **Configuraciones de autenticación y autorización**: Especifica cómo se gestionan la autenticación y la autorización de los usuarios en la aplicación.

El descriptor de aplicación Java es esencial para la correcta configuración y despliegue de aplicaciones web Java. Proporciona información que el contenedor de servlets utiliza para comprender cómo debe manejar las solicitudes web y cómo se comunican los componentes de la aplicación.

Cabe señalar que, en las versiones más recientes de Java EE, se han introducido configuraciones que permiten simplificar la configuración de la aplicación, reduciendo la necesidad de un descriptor de aplicación Java en algunos casos.

Sin embargo, el descriptor de la aplicación Java sigue siendo una parte importante de la especificación y se utiliza en muchas aplicaciones web Java tradicionales.

Texto

Descripción generada automáticamente

En este ejemplo se establece el espacio de nombres XML xmlns para la versión 4.0 de Java EE. Se define un servlet llamado "MiServlet" utilizando la etiqueta <servlet>. Se especifica el nombre del servlet y la clase Java que implementa el servlet en las etiquetas <servletname> y <servlet-class>, respectivamente.

Luego, se configura el mapeo de URL para el servlet mediante la etiqueta <servletmapping>. Especificamos el nombre del servlet que definimos anteriormente y la URL a la que responderá el servlet en las etiquetas <servlet-name> y <url-pattern>.

En este caso, el servlet "MiServlet" responderá a las solicitudes que coincidan con la URL "/miurl". Puedes agregar más servlets y mapeos de URL según las necesidades de tu aplicación.

Cuando hablamos de mapeo de URL para el servlet nos referimos a la configuración que asocia una URL específica con un servlet en una aplicación web Java. Permite que el servidor web (como Apache Tomcat) sepa qué servlet debe manejar una solicitud HTTP en función de la URL a la que se accede.

Por ejemplo, supongamos que tenemos un servlet llamado "MiServlet" y se quiere que maneje solicitudes cuando los usuarios accedan a una URL específica, como "/miurl".

Para lograrlo, se debe configurar el mapeo de URL para el servlet de la siguiente manera en el descriptor de aplicación Java (web.xml):

Texto

Descripción generada automáticamente

En este ejemplo, se establece el mapeo de URL para el servlet "MiServlet". El servlet "MiServlet" se asocia a la URL "/miurl". Cuando un cliente accede a esta URL, el servidor web consultará el descriptor de la aplicación Java (web.xml) para determinar qué servlet debe manejar la solicitud, y luego dirigirá la solicitud al servlet "MiServlet" para su procesamiento.

## RECURSOS ESTÁTICOS

Los recursos estáticos, como archivos HTML, CSS, imágenes y scripts JavaScript, son parte de la interfaz de usuario de la aplicación web. Estos archivos se sirven al cliente tal como están y no requieren procesamiento dinámico.

## FILTROS

Los filtros son componentes que permiten la manipulación de solicitudes y respuestas en una aplicación web. Se utilizan para realizar tareas comunes, como autenticación, autorización, compresión y registro de solicitudes.

## RECURSOS DE DATOS

En un proyecto más completo, podrías tener recursos de datos, como conexiones a bases de datos o servicios web, para acceder y manipular datos en la aplicación.

## PÁGINAS DE ERROR

Las páginas de error son páginas especiales que se muestran cuando ocurren errores durante la ejecución de la aplicación web. Pueden personalizarse para brindar información útil al usuario en caso de problemas.

# COMPRESIÓN DE PROYECTOS JAVA. ARCHIVOS .WAR

El proceso de empaquetar un proyecto Java en un archivo WAR. Se puede utilizar una herramienta de construcción como Maven para crear el archivo WAR, o hacerlo manualmente. Aquí se muestra cómo hacerlo manualmente: Supongamos que tenemos una estructura de proyecto web llamada "MiProyectoWeb" con la siguiente estructura de directorios:

Pantalla de computadora con letras

Descripción generada automáticamente con confianza media

Para empaquetar este proyecto en un archivo WAR, se seguirán estos pasos:

1. Crear una carpeta llamada "WEB-INF" en la raíz de tu proyecto si aún no existe.
2. Crear un archivo llamado "web.xml" dentro de la carpeta "WEB-INF". Este archivo es el descriptor de implementación de la aplicación web y define cómo se configuran los servlets y otras configuraciones de la aplicación. Este es un ejemplo simple de un archivo "web.xml":

Texto

Descripción generada automáticamente

1. Mover todas las clases Java compiladas y otros recursos necesarios, como JSP, al directorio raíz de tu proyecto.
2. Abrir una terminal o línea de comandos en el directorio raíz de tu proyecto.
3. Mediante el comando "jar" o una herramienta similar para crear un archivo WAR con todos los archivos y directorios en tu proyecto.
4. Por ejemplo, en un sistema Linux, se puede utilizar el siguiente comando para crear el fichero MiProyectoWeb.war:

jar -cvf MiProyectoWeb.war \*

1. A continuación, se puede desplegar en un servidor web compatible, como Apache Tomcat copiando el archivo WAR en el directorio "webapps" del servidor Tomcat, y se desplegará automáticamente.

En un entorno de desarrollo real, es común utilizar herramientas de construcción como Maven o Gradle para gestionar y empaquetar proyectos web de manera más eficiente.

Estas herramientas también facilitan la gestión de dependencias y la compilación de proyectos Java.

## CREACIÓN DE UNA APLICACIÓN

Veamos un ejemplo en un contexto. Supongamos que tenemos una estructura de proyecto web llamada "MiProyectoWeb". Dentro de esta estructura, crea un paquete llamado "miservlets" y, a continuación, crea un archivo Java llamado "HolaMundoServlet.java" con el siguiente contenido:

Texto

Descripción generada automáticamente

Luego, se compila este servlet y se empaqueta en un archivo WAR. Asegúrate de que el archivo WAR contenga la estructura de paquetes adecuada y el archivo "web.xml" para definir el servlet. Por ejemplo, el archivo "web.xml" podría verse así:

Captura de pantalla de computadora

Descripción generada automáticamente

Finalmente, se despliega el archivo WAR en el servidor web Apache Tomcat 9.

Una vez desplegado, se puede acceder al servlet mediante el navegador utilizando la URL http://localhost:8080/MiProyectoWeb/HolaMundo .

Nos aparecerá en el navegador un mensaje "Hola Mundo desde un Servlet".

# JSON (JAVASCRIPT OBJECT NOTATION)

Un ejemplo simple de un objeto JSON en Java utilizando la biblioteca Jackson, que es una de las bibliotecas más comunes para trabajar con JSON en Java. Para usar Jackson, primero se agregará la dependencia en tu proyecto, ya sea mediante Maven o Gradle.

Supongamos que se quiere representar información de una persona en formato JSON.

Texto

Descripción generada automáticamente

En este ejemplo:

* Creamos un objeto `ObjectMapper` de Jackson, que se utiliza para convertir objetos Java a JSON y viceversa.
* Definimos una clase interna `Persona` que representa la estructura de datos que queremos convertir en JSON. Esta clase tiene tres propiedades: `nombre`, `apellido` y `edad`.
* Creamos una instancia de la clase `Persona` con algunos datos de ejemplo.
* Utilizamos el `ObjectMapper` para convertir la instancia de `Persona` en una cadena JSON mediante el método `writeValueAsString`.
* Imprimimos el JSON resultante en la consola. El resultado será una cadena JSON que representa los datos de la persona: {"nombre":"Juan","apellido":"Pérez","edad":30}