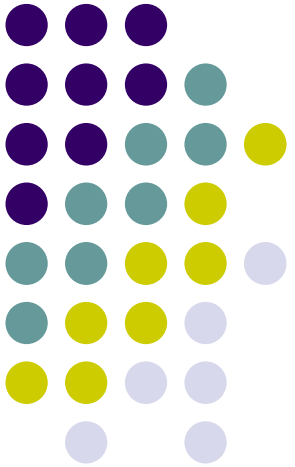
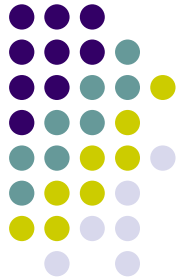
Java



Servlets

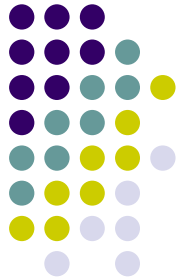
Lic. Claudio Zamoszczyk
claudio@honou.com.ar

Contenidos



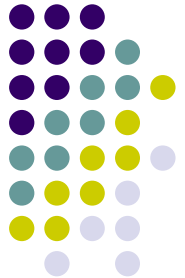
- Estructura y despliegue de una aplicación web java.
- Introducción
- Servlets

Estructura y despliegue de aplicaciones Web



Una APLICACIÓN WEB consiste en un conjunto de servlets, páginas jsp, archivos html, clases Java empaquetadas o no en archivos .jar y otro tipo de recursos tales como imágenes, archivos javascript, textos, etc.

Estructura y despliegue de aplicaciones Web

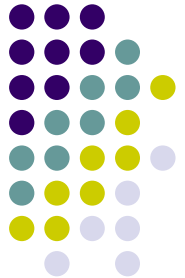


Una aplicación web puede existir de dos modos:

- Mediante una estructura de directorios basada en la especificación definida por Sun para los Servlets. Dentro de esta estructura deben ubicarse de forma adecuada los componentes de la aplicación.

- Mediante un archivo de extensión war (Web Application Resource, a veces también se le suele llamar Web ARchive) que engloba a todo su contenido. Se crea del mismo modo que un archivo jar. Este empaquetamiento se produce en la etapa de puesta en producción, es decir, cuando la aplicación ha sido testeada y depurada para su utilización final. Cuenta con una estructura similar al punto anterior.

Estructura y despliegue de aplicaciones Web



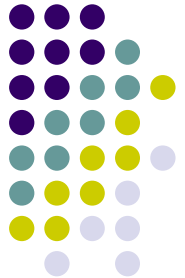
Estructura de directorio

/nombre-app-web/
/nombre-app-web/WEB-INF/
/nombre-app-web/WEB-INF/classes/
/nombre-app-web/WEB-INF/lib/

/nombre-app-web: será el directorio raíz de nuestra aplicación, este deberá existir en el directorio de despliegue del servidor. Podrá contener archivos html, jsp, js, css, directorios, imágenes, etc.

/nombre-app-web/WEB-INF: este directorio contendrá el archivo web.xml de configuración de nuestra aplicación web.

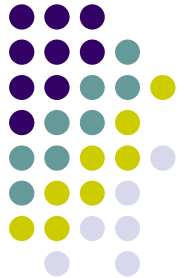
Estructura y despliegue de aplicaciones Web



/nombre-app-web/WEB-INF/classes/: aquí copiaremos todas nuestras clases con sus paquetes correpondiente (clases de negocio, servlets, utilidades, etc)

/nombre-app-web/WEB-INF/lib/: contiene todos los archivos .jar que sean útiles para el desarrollo de nuestra aplicación web (drivers jdbc, frameworks, etc).

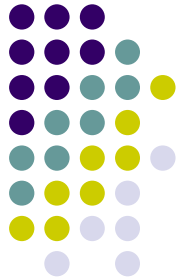
El archivo web.xml



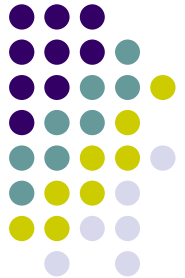
El archivo web.xml proporciona información sobre configuración y despliegue de los componentes Web que componen una aplicación Web. Los ejemplos de los componentes Web son los parámetros de servlets, las definiciones de servlets y JSP, reglas de seguridad, manejo de sesiones, etc.

El archivo web.xml debe residir en el directorio WEB-INF bajo el contexto de la jerarquía de directorios que existe para una aplicación Web. Por ejemplo, si la aplicación es client.war, entonces el archivo web.xml se coloca en el directorio /client/WEB-INF.

¿Preguntas?



Servlets: Introducción

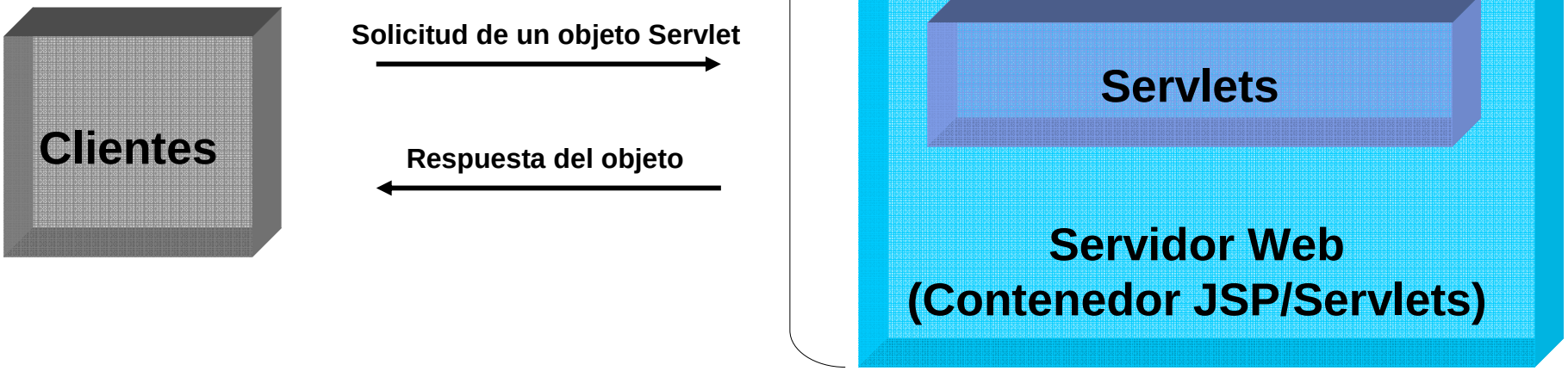


Los Servlets son componentes que se ejecutan en el contenedor web. Estos componentes pueden ser ejecutados en cualquier plataforma o en cualquier servidor debido a la tecnología Java.

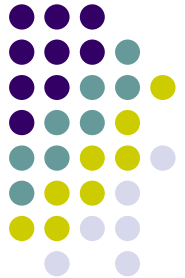
Cuando se recibe una petición del cliente, el contenedor web inicia el servlet requerido. El Servlet procesa la petición del cliente y envía la respuesta de vuelta al servidor, que es enrutada al cliente.

A diferencia de las paginas JSP, donde el modelo esta basado en la inclusión de tags especiales sobre un documento html, un servlet plantea una manera mas orientada a objetos, en la cual se crea una una clase java, que implementa la interface HTTPServlet, permitiendo la generación de contenido dinámico.

Introducción



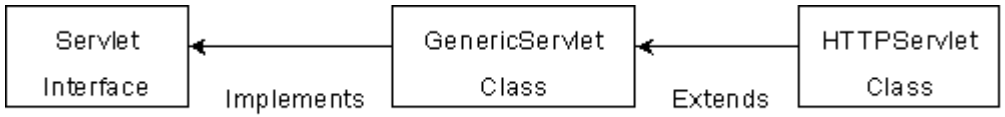
Estructura del Servlet y Ciclo de Vida



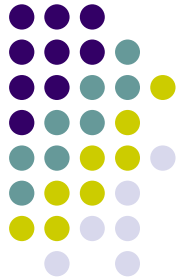
Antes de meternos en el ciclo de vida de los servlets, necesitamos comprender las clases básicas y las interfaces usados en la implementación del servlet.

Public Interface Servlet

Todo servlet debe directa o indirectamente implementar este interfaz. Como cualquier otro interfaz de Java, este es también una colección de declaraciones vacías de métodos. Los siguientes métodos están declarados en el interfaz Servlet.



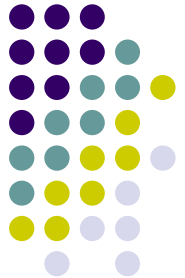
Estructura del Servlet y Ciclo de Vida



public abstract void init (ServletConfig config) throws ServletException.

El método init se usa para inicializar los parámetros proporcionados por el objeto ServletConfig. Se invoca sólo una vez, a menos que el servlet sea reiniciado si se destruye y se vuelve a cargar. El método init es el lugar en el que inicializar los parámetros de configuración como la conexión con una base de datos, la inicialización de archivos y otros valores de entorno. Ninguno método del servlet puede ser invocado a no ser que el servlet esté inicializado mediante el uso del método init().

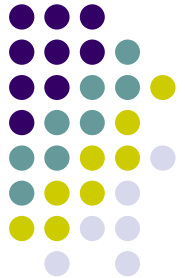
Estructura del Servlet y Ciclo de Vida



public abstract ServletConfig getServletConfig ()

Este método proporciona el objeto ServletConfig para la inicialización de los parámetros del servlet. Una vez que hayan sido especificados en el archivo web.xml, se puede acceder a ellos usando el objeto ServletConfig.

Estructura del Servlet y Ciclo de Vida



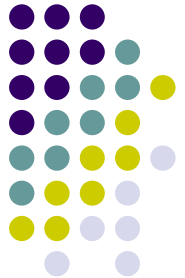
public abstract void service (ServletRequest req, ServletResponse res) throws ServletException, IOException.

El método *service* es el punto esencial del modelo petición respuesta del protocolo HTTP. Recibe una petición del cliente en forma de objeto ServletRequest. Los parámetros del cliente son pasados junto al objeto de petición (aunque existen otras formas de enviar parámetros desde el cliente al servlet, por ejemplo, usando cookies o por medio de una reescritura del URL). La respuesta resultante se envia al cliente usando el objeto ServletResponse.

Además de contar con el método service, también existen métodos mas específicos que permiten procesar la información dependiendo del tipo de llamada http.

doGet (HttpServletRequest, HttpServletResponse)
doPost (HttpServletRequest, HttpServletResponse)

Estructura del Servlet y Ciclo de Vida



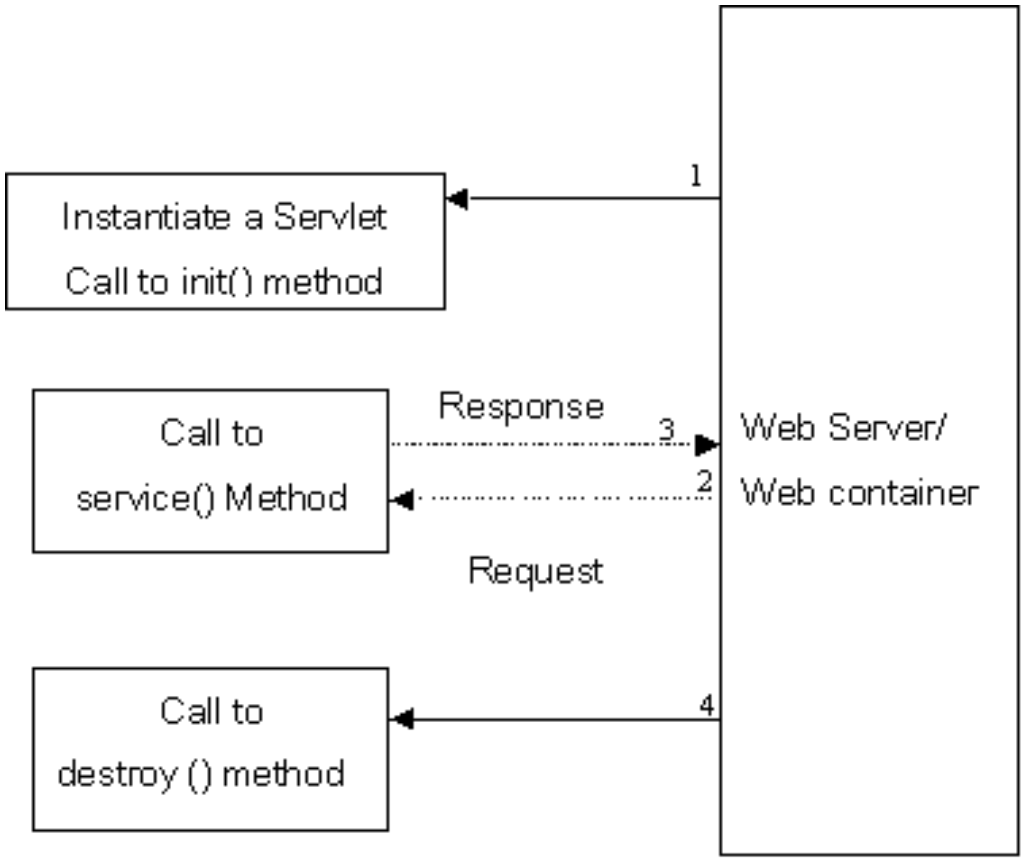
public abstract String getServletInfo()

Este método se usa para la extracción de metadatos del servlet, como por ejemplo el autor, la versión del servlet, y otra información concerniente al copyright. El método tendrá que ser redefinido dentro del servlet para que devuelva esta información.

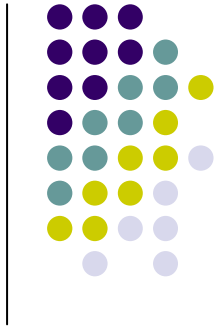
public abstract void destroy ()

El método destroy se invoca para liberar todos los recursos solicitados como la base de datos, y otros recursos del servidor. También se encarga de la sincronización de cualquier hilo pendiente. Este método se llama una sola vez, automáticamente, como el método init.

Estructura del Servlet y Ciclo de Vida

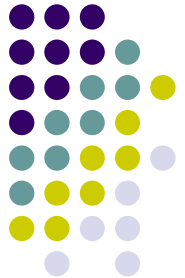


Primer Servlet



```
package edu.cursoweb;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class PrimerServlet extends HttpServlet  {

    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    String title = "Hola mundo Servlet";
    out.println("<HTML><HEAD><TITLE>");
    out.println(title);
    out.println("</TITLE></HEAD><BODY>");
    out.println("<H1>" + title + "</H1>");
    out.println("<P>Hola Mundo.");
    out.println("</BODY></HTML>");
    out.close();
  }

}
```

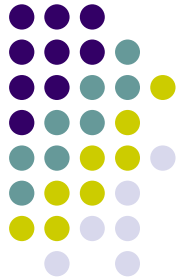
Primer Servlet



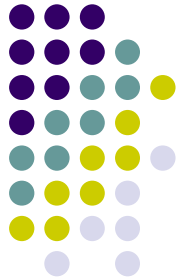
Archivo de despliegue web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
  <servlet>
    <servlet-name>PrimerServlet</servlet-name>
    <servlet-class>edu.cursoweb.PrimerServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>PrimerServlet</servlet-name>
    <url-pattern>/hola/PrimerServlet</url-pattern>
  </servlet-mapping>
</web-app>
```

¿Preguntas?



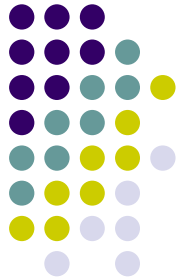
Objetos implícitos



Del mismo modo que el motor JSP nos ofrece instancias de un conjunto de clases para el manejo de información y demás temas; cuando trabajamos con Servlets estos objetos también están presentes y los podemos utilizar de la misma forma:

HttpServletRequest
HttpServletResponse
HttpSession
ServletContext
ServletConfig

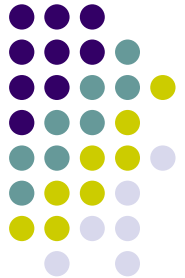
Objetos implícitos



HttpServletRequest: contiene información de datos enviados al servidor a través de una página web por parte del cliente. En el se pueden encontrar cookies, parámetros existentes en el queryString, información de formularios, etc.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)….
        ….
        String nombre =  request.getParameter("nombre");
        ….

}
```

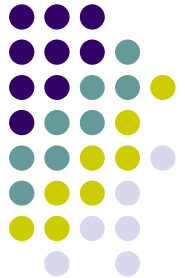
Objetos implícitos



HttpServletResponse: permite trabajar con la respuesta que es enviada al cliente, en ella podemos manejar el tipo de salida, contenido, manejo de cabeceras, códigos de estado, cookies, entre otras cosas.

```
public void doGet(HttpServletRequest request, HttpServletResponse response) ….

    response.setContentType("text/html");
    PrintWriter out = response.getWriter();

    out.println("<HTML><HEAD><TITLE>");
    ……..


    Cookie miCookie = new Cookie("miCookie","1234567");
    response.addCookie(miCookie);
    …..
```

Objetos implicitos

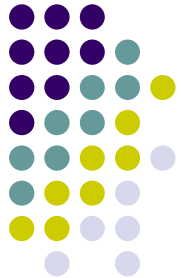


ServletContext:Este objeto es común para toda la aplicación web y, entre otras cosas, nos permite almacenar información que será accesible desde todas las páginas de la aplicación web (similar a una variable global).

Para guardar y recuperar valores:

```
ServletContext context = this.getServletContext();
String texto = (String) context.getAttribute("clave");
context.setAttribute("clave", “12454555”);
```

Objetos implícitos

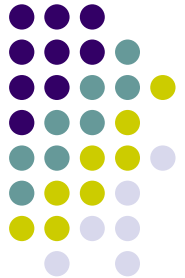


Otra utilidad del objeto ServletContext es la de recuperar información global para todos los servlets y paginas jsp existe en el archivo web.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>

   <context-param>
     <param-name>ParmetroGoblal</param-name>
     <param-value>Hola Mundo</param-value>
   </context-param>……
```

```
String parmetroGoblalValue = getServletContext().getInitParameter("ParmetroGoblal");
```

Objetos implícitos



HttpSession: Nos permite acceder a la sesión asociada a la petición. A través de este objeto podemos, entre otras cosas, guardar objetos que serán accesibles desde cualquier JSP/Servlet de la sesión o invalidarla.

Para guardar y recuperar información usaremos:

```
public void doGet(HttpServletRequest request, HttpServletResponse response)…

        HttpSession session = request.getSession();
        session.setAttribute("clave", “12454555”);
        String texto = (String) session.getAttribute("clave");
```

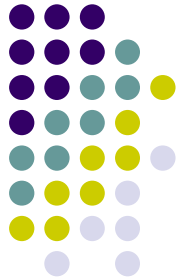
Y para invalidar la sesión:

```
session.invalidate();
```

web.xml

```
<session-config>
    <session-timeout>30</session-timeout>
</session-config>
```

Objetos implícitos



ServletConfig: Permite acceder a parámetros de inicialización de un servlet especifico existente en el archivo web.xml.

```
<servlet>
   <servlet-name>ServletContextInfo</servlet-name>
   <servlet-class>edu.cursoweb.ServletContextInfo</servlet-class>
   <init-param>
      <param-name>EjemploVar</param-name>
      <param-value>123456</param-value>
   </init-param>
</servlet>
```

```
String directorioImgValue = getServletConfig().getInitParameter("directorioImg");
String ejemploVarValue = getServletConfig().getInitParameter("EjemploVar");
```

¿Preguntas?