

Servlets, Conceptos Básicos

Servlets, Conceptos Básicos

- Procesamiento en Java del lado del servidor.
- Extensiones al modelo cliente-servidor.

Servlets

- Los servlets son:
 - Programas capaces de extender las capacidades de un servidor web.
 - Nueva opción a la programación en CGI-BIN.
 - Aplicaciones útiles para la programación de portales y sitios adaptativos.
 - Aplicaciones en Java.

Servlets

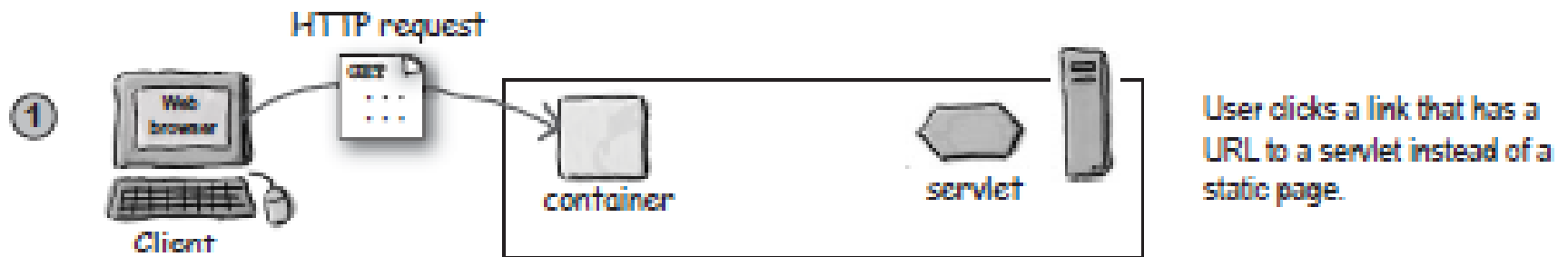
- Mejoras introducidas por la tecnología Servlets:
 - Creación de procesos con múltiples hilos dentro de un servidor.
 - Creación de procesos dentro de un hilo dentro de un servidor.
 - Creación de procesos con múltiples hilos dentro de múltiples servidores.

Servlets

- Mejoras introducidas por la tecnología Servlets (cont...)
 - Un servlet funciona en la mayor parte de las plataformas comerciales.
 - Un programa CGI sólo opera bajo ambientes restringidos (Unix).

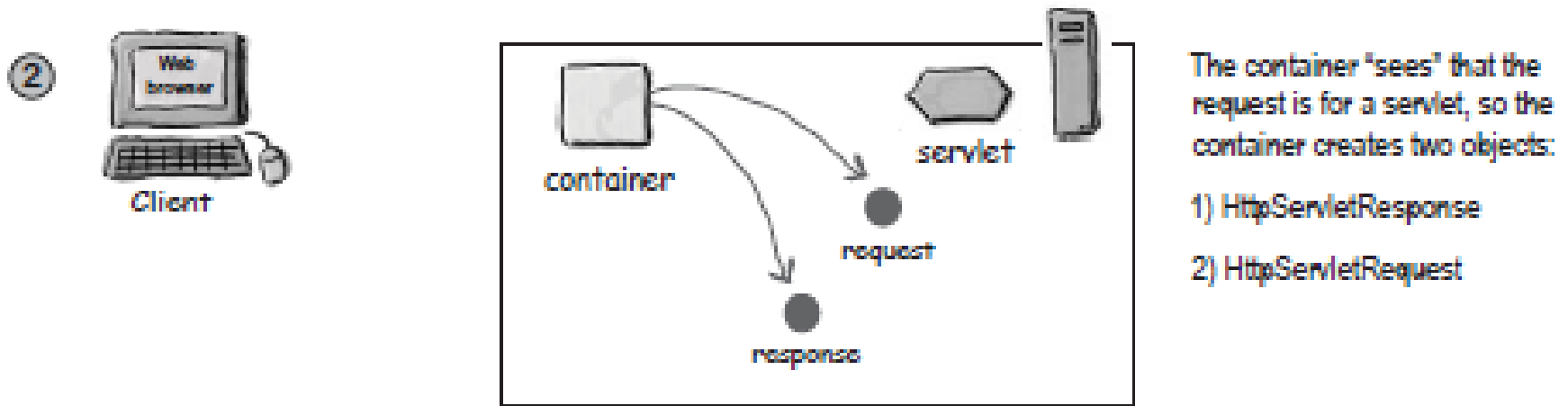
Creación y ejecución de Servlets

Manejo de un request por el contenedor



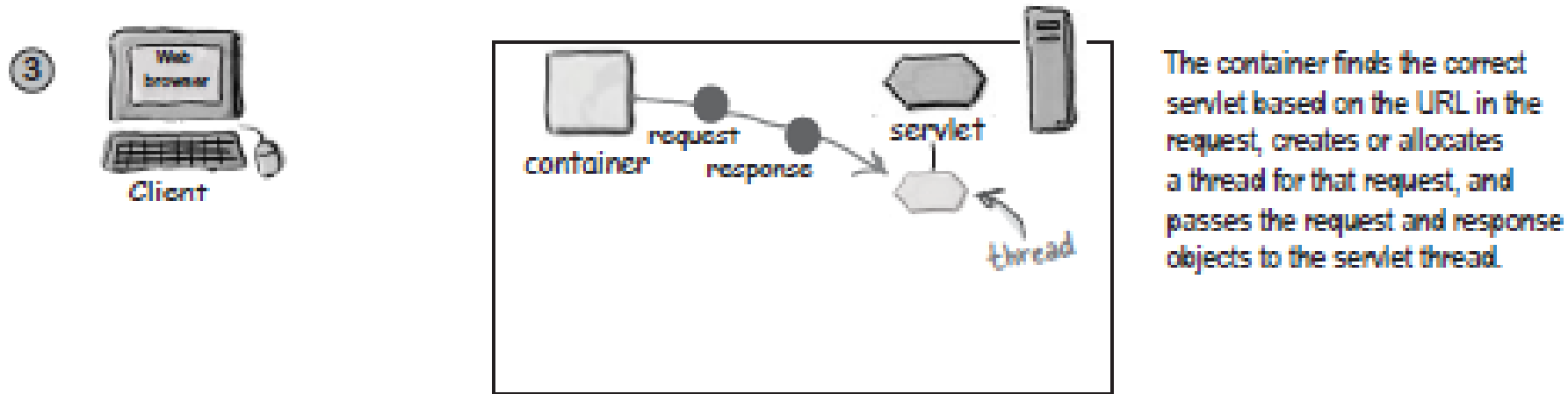
Head First Servlets & JSP, Bryan Basham et al. O'Reilly

Manejo de un request por el contenedor



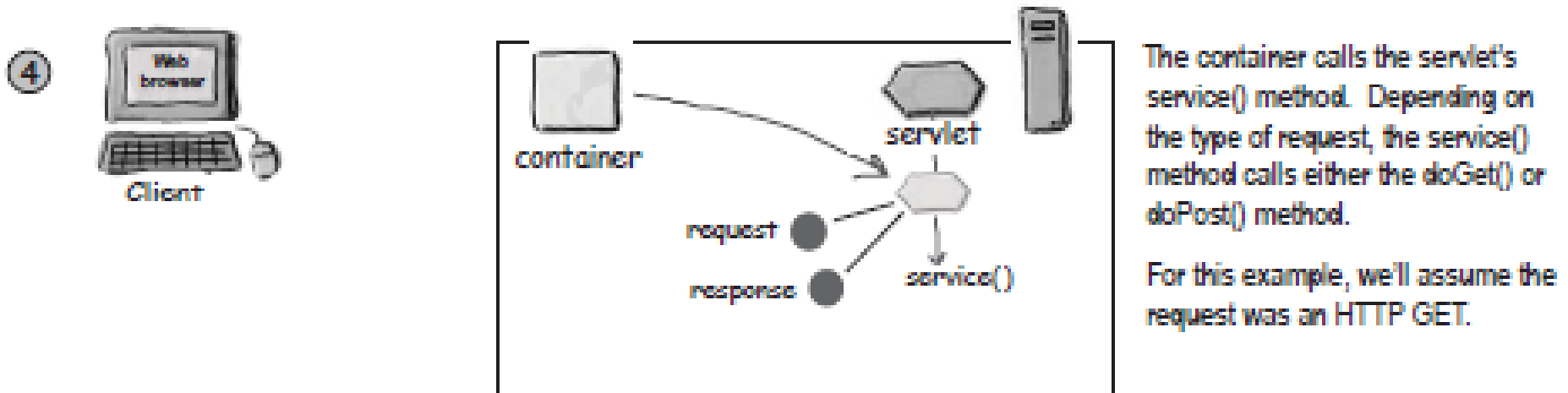
Head First Servlets & JSP, Bryan Basham et al. O'Reilly

Manejo de un request por el contenedor



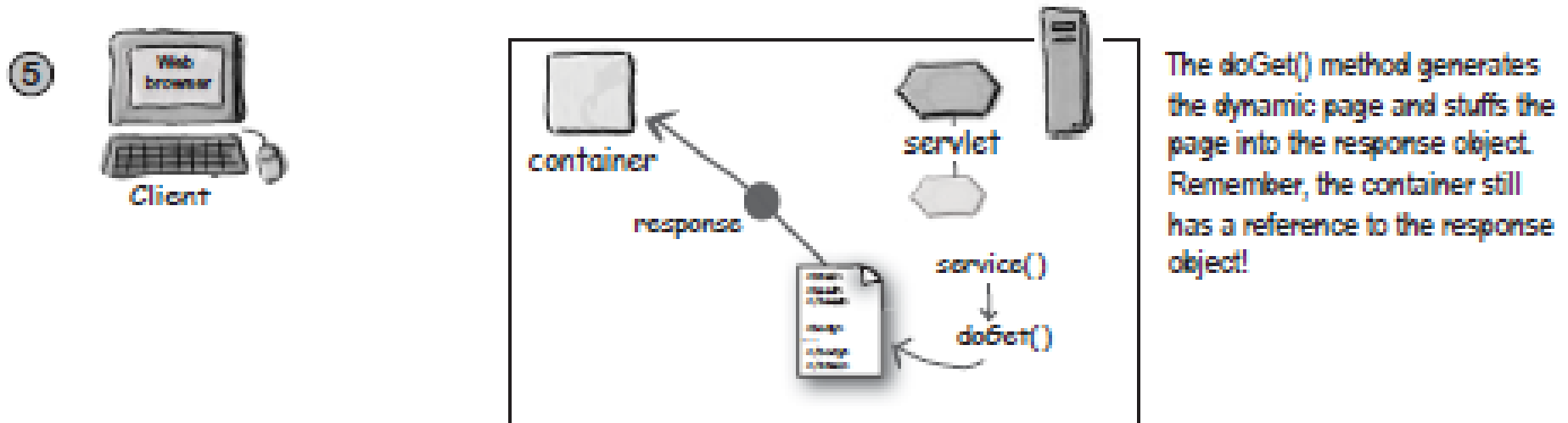
Head First Servlets & JSP, Bryan Basham et al. O'Reilly

Manejo de un request por el contenedor



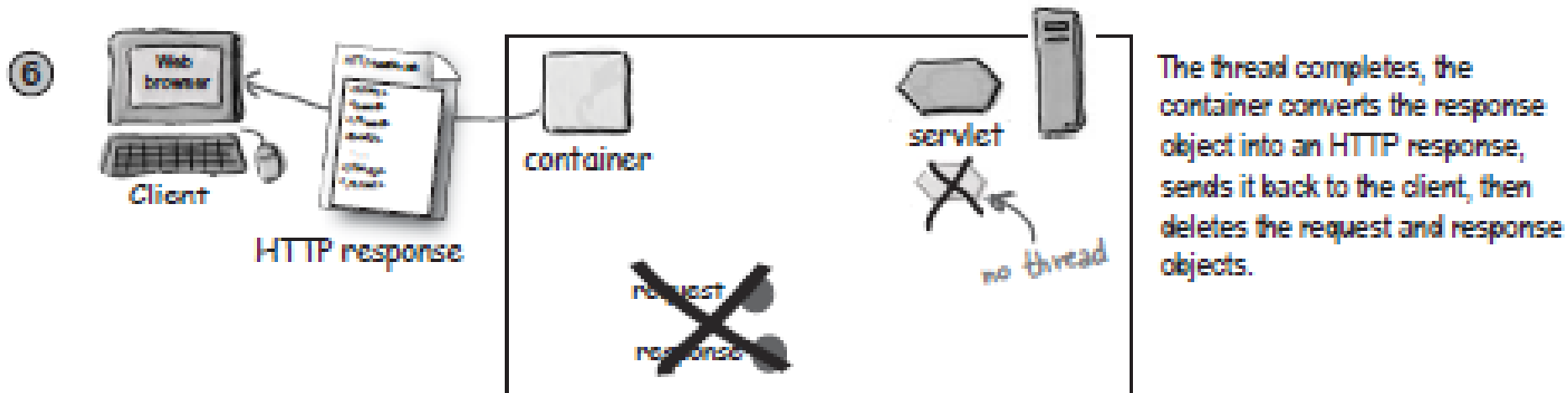
Head First Servlets & JSP, Bryan Basham et al. O'Reilly

Manejo de un request por el contenedor



Head First Servlets & JSP, Bryan Basham et al. O'Reilly

Manejo de un request por el contenedor



Head First Servlets & JSP, Bryan Basham et al. O'Reilly

Ciclo de vida de un servlet_[2]

- Cliente hace la solicitud.
- El contenedor verifica que es una llamada al servlet.

[2] Head First Servlets & JSP, Bryan Basham et al. O'Reilly

Ciclo de vida de un servlet

- El contenedor crea dos objetos del tipo:
 - HttpServletResponse
 - HttpServletRequest
- El contenedor, basado en el URL de la solicitud, localiza el servlet

Ciclo de vida de un servlet

- El contenedor crea o asigna un thread para esa solicitud y transfiere a ese thread los objetos
- El contenedor llama al método `service()`
 - Dependiendo el tipo de solicitud, el método llama al método `doGet()` o `doPost()`

Ciclo de vida de un servlet

- El método llamado por el método `service()` genera una página dinámica y la agrega el objeto de la respuesta.
- El thread termina.

Ciclo de vida de un servlet

- El contenedor convierte el objeto de la respuesta en un respuesta HTTP y la envía de regreso al cliente.
- Finalmente el contenedor borra los objetos de la solicitud y respuesta.

Estructura de un código de un servlet

- Importar bibliotecas

```
import javax.servlet.*;
```

```
import javax.servlet.http.*;
```

```
import java.io.*;
```

- Definir una clase que herede de HttpServlet


```
public class ServApp1 extends HttpServlet {...}
```

Estructura de un código de un servlet

- Sobre-escribir el método dependiendo del tipo de solicitud (doGet() o doPost()).

Por ejemplo:

```
public void doGet(HttpServletRequest peticion,  
                  HttpServletResponse respuesta)
```



Aquí es donde el servlet obtiene la referencia a los objetos de petición y respuesta que son creados por el contenedor.

Estructura de un código de un servlet

- Se indica tipo de respuesta (en este caso texto)

```
respuesta.setContentType("text/html");
```

- Se obtiene un 'escritor' del objeto de respuesta del servlet.

```
PrintWriter out = respuesta.getWriter();
```

Estructura de un código de un servlet

- Entonces ya podemos ‘escribir’ al objeto de respuesta con el siguiente formato:

```
out.println([HTML estándar]);
```

– Por ejemplo:

```
out.println("<html>");
```

```
out.println("<table border=0>");
```

```
out.println("<li>");
```

Estructura de un código de un servlet

- Ahora un ejemplo de una sección de código HTML que muestra un mensaje formateado:

```
out.println("<h1>");  
out.println("Bienvenidos y bienvenidas!");  
out.println("</h1>");
```

Publicación al contenedor

- Para publicar un servlet es necesario mapear el URL público.
- Esto se hace en un archivo xml de propósito específico llamado 'Deployment Descriptor' (DD).
- Este archivo tiene el nombre de web.xml.

Publicación al contenedor

- Existen dos elementos dentro del descriptor de publicación que se utilizan para publicar el servlet

`<servlet>`

y

`<servlet-mapping>`

<servlet>

- Este elemento mapea el nombre interno al nombre de la clase completamente calificado, esto es, toda la trayectoria en la que se encuentra el servlet.

<servlet-mapping>

- Este elemento mapeará al nombre interno al nombre del URL público.
- Este URL público es el que se utiliza para llamar a la clase.

Ejemplo

Nombre interno

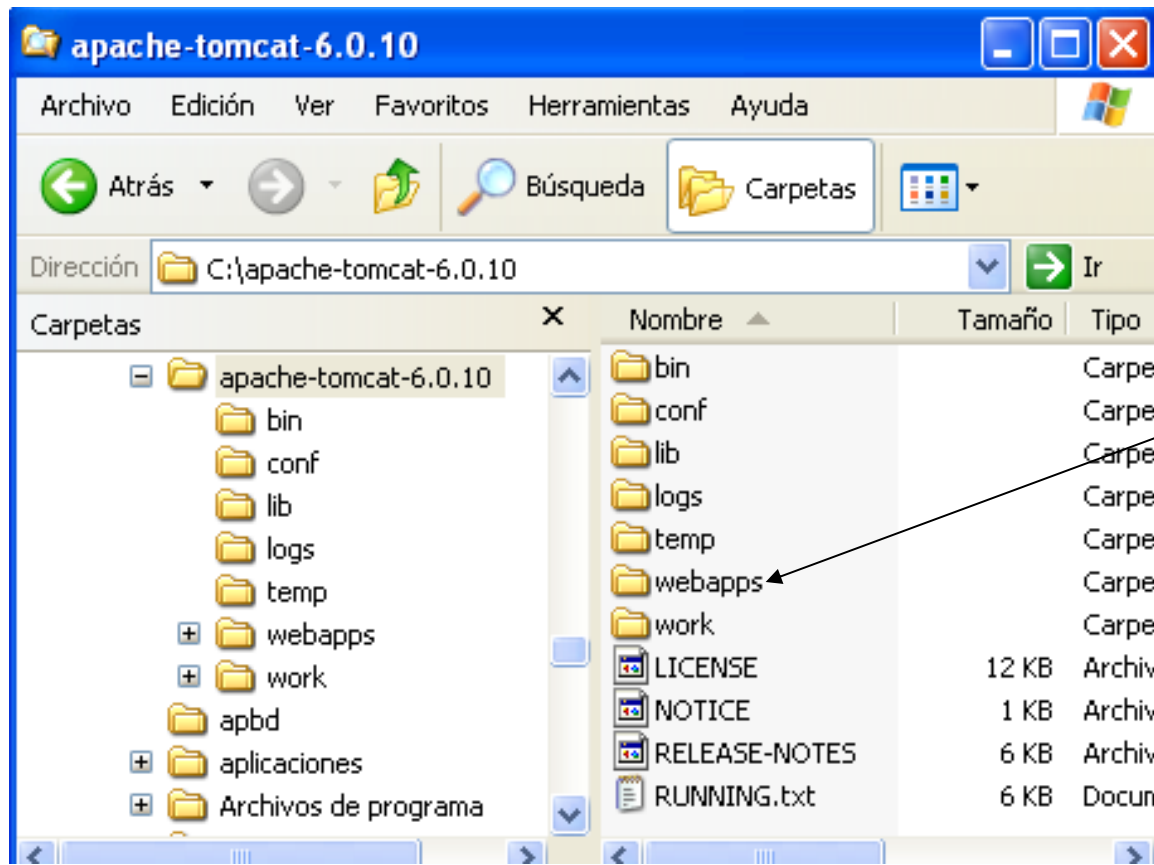
```
<servlet>  
  <servlet-name>ServAppInterno</servlet-name>  
  <servlet-class>edu.ipn.upiicsa.servlet.ServApp</servlet-class>  
</servlet>
```

Nombre completamente calificado de la clase

```
<servlet-mapping>  
  <servlet-name>ServAppInterno</servlet-name>  
  <url-pattern>/ServAppPublico</url-pattern>  
</servlet-mapping>
```

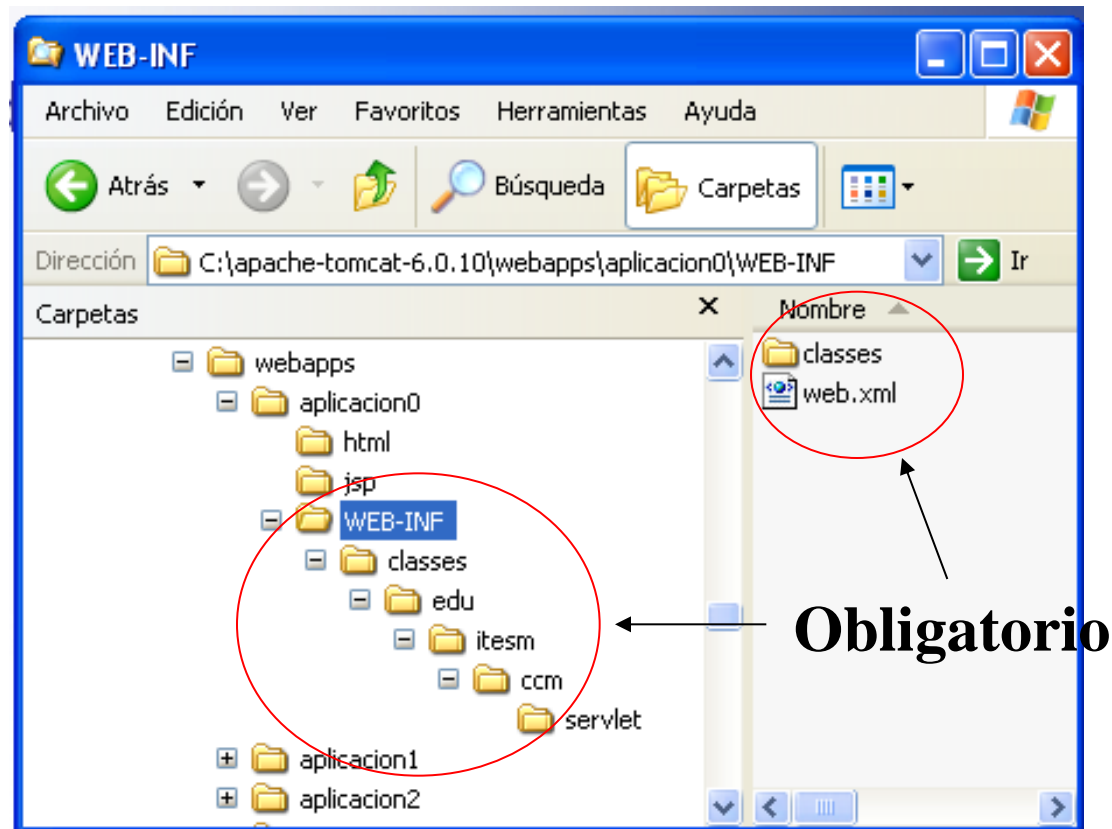
URL publico

Publicando el servlet en el contenedor

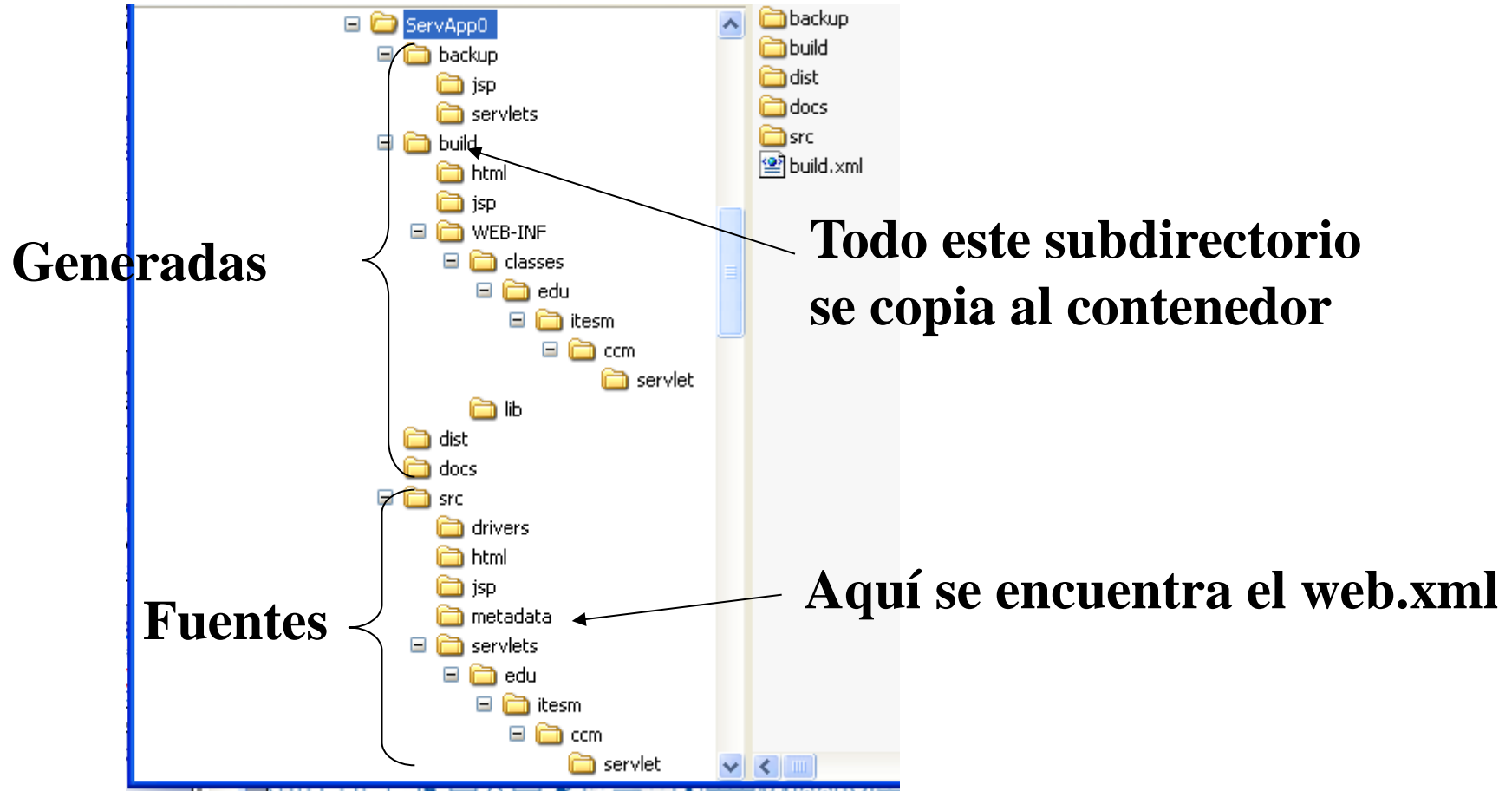


**En este subdirectorio
se instalará**

La distribución típica de una aplicación de web en el contenedor de servlets

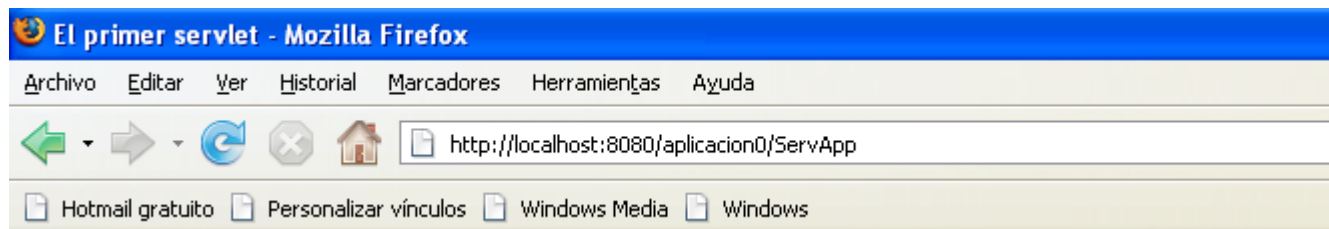


Una distribución propuesta del ambiente de desarrollo



El primer servlet publicado (ServApp0)

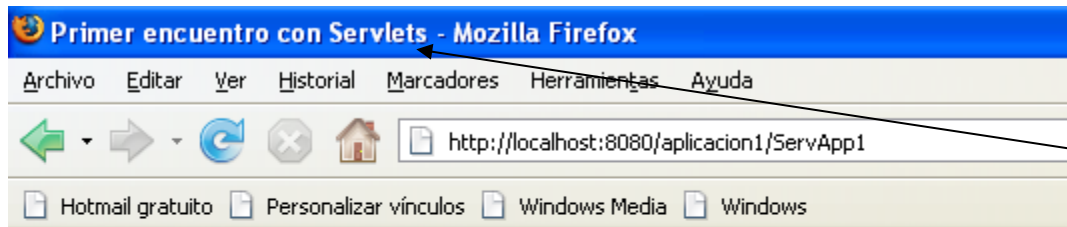
`http://localhost:8080/aplicacion0/ServApp`



Bienvenidos y bienvenidas a mi primer servlet!

Servlet parametrizado (ServApp1)

<http://localhost:8080/aplicacion1/ServApp1>



Titulo parametrizado

Bienvenidos y bienvenidas ServApp1!

Obteniendo información sobre la conexión (ServApp2)



Creada con:



`http://localhost:8080/aplicacion2/ServApp2`

Notar el uso de imágenes

Ejemplo #2: Obteniendo información sobre la conexión.

Método llamado: GET
URI de procedencia: /aplicacion2/ServApp2
Protocolo utilizado: HTTP/1.1
Dirección remota: 127.0.0.1