Head.First.Servlet.Jsp

02 **Web App Architecture**

Table of Contents

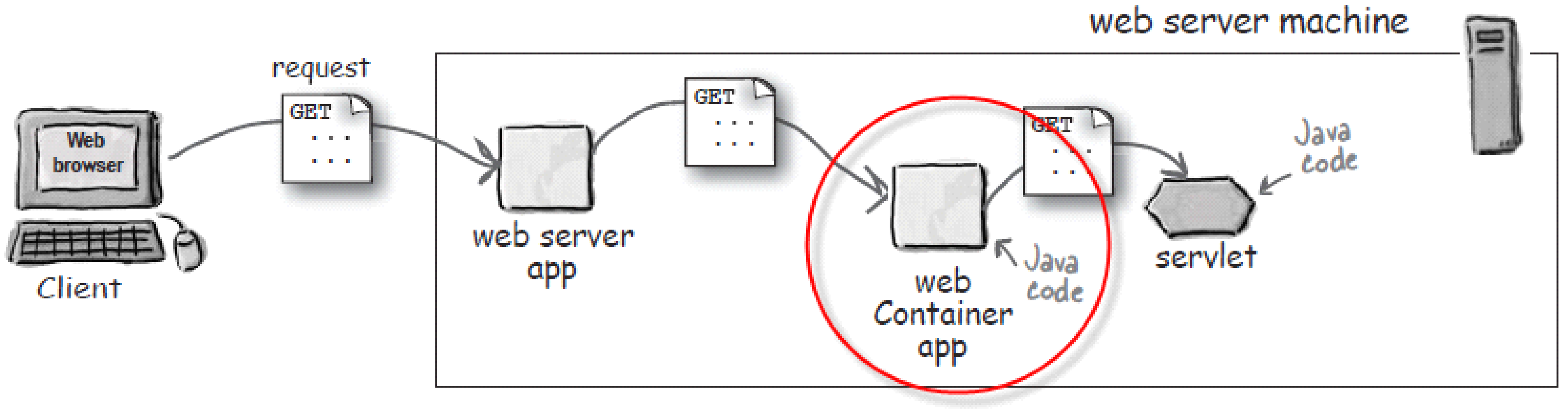
Table of Contents

Web App Architecture

- What is a Container and what does it give you?
- How it looks in code (and what makes a servlet)
- Naming servlets and mapping them to URLs using the DD
- First Web App
- How J2EE fits into all this

What is a Container ?

- **Servlets don’t have a main() method. They’re under the control of another Java application called a Container.**



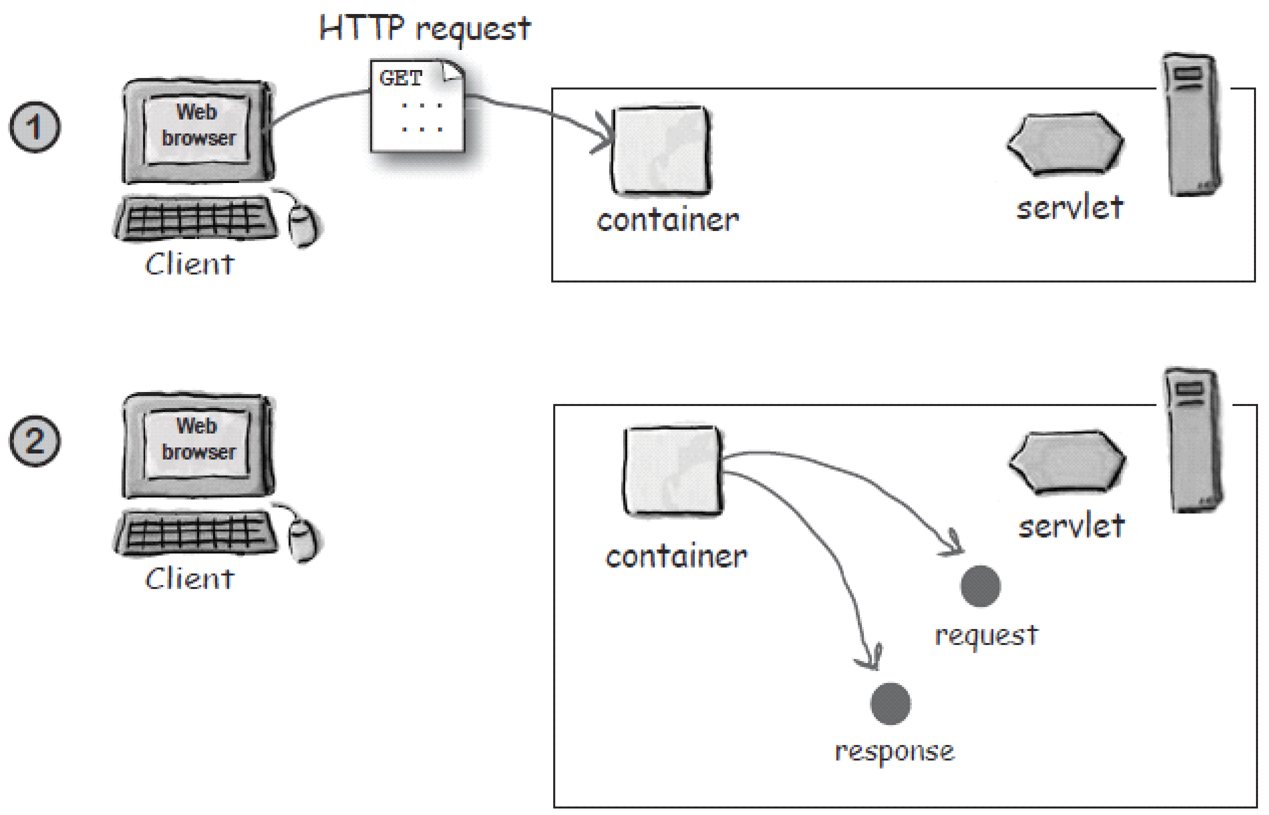
What does the Container give you?

- **Communications support**
  - You don’t have to build a ServerSocket, listen on a port, create streams, etc.
- **Lifecycle Management**
  - It takes care of loading the classes, instantiating and initializing the servlets, invoking the servlet methods, and making servlet instances eligible for garbage collection.

What does the Container give you?

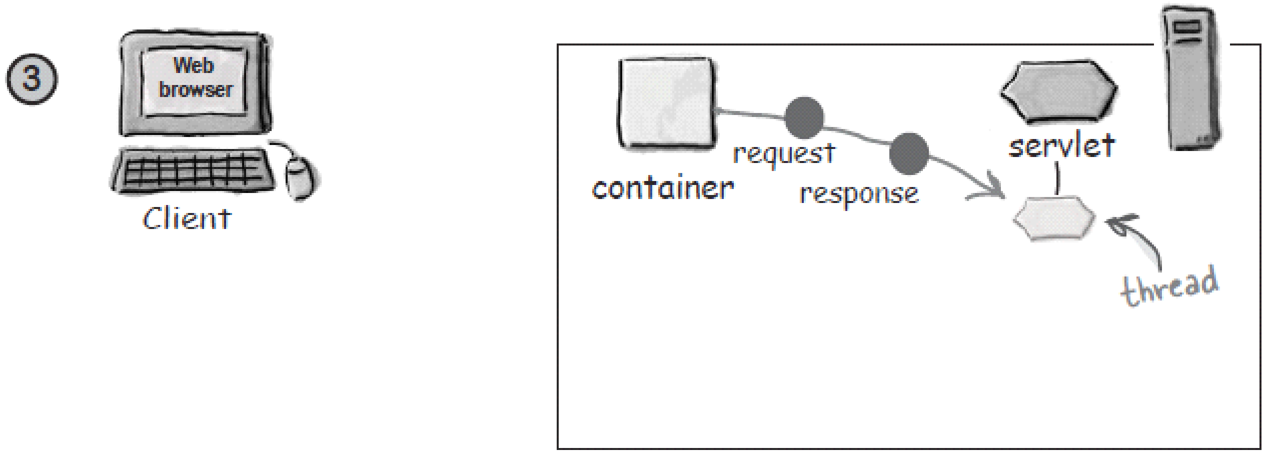
- **Multithreading Support**
  - The Container automatically creates a new Java thread for every servlet request it receives.
- **Declarative Security**
  - With a Container, you get to use an XML deployment descriptor to configure (and modify) security without having to hard-code it into your servlet (or any other) class code.
- **JSP Support**
  - You already know how cool JSPs are. Well, who do you think takes care of translating that JSP code into real Java?

How the Container handles a request 1 2/6



The container creates two objects: **1) HttpServletResponse 2) HttpServletRequest**

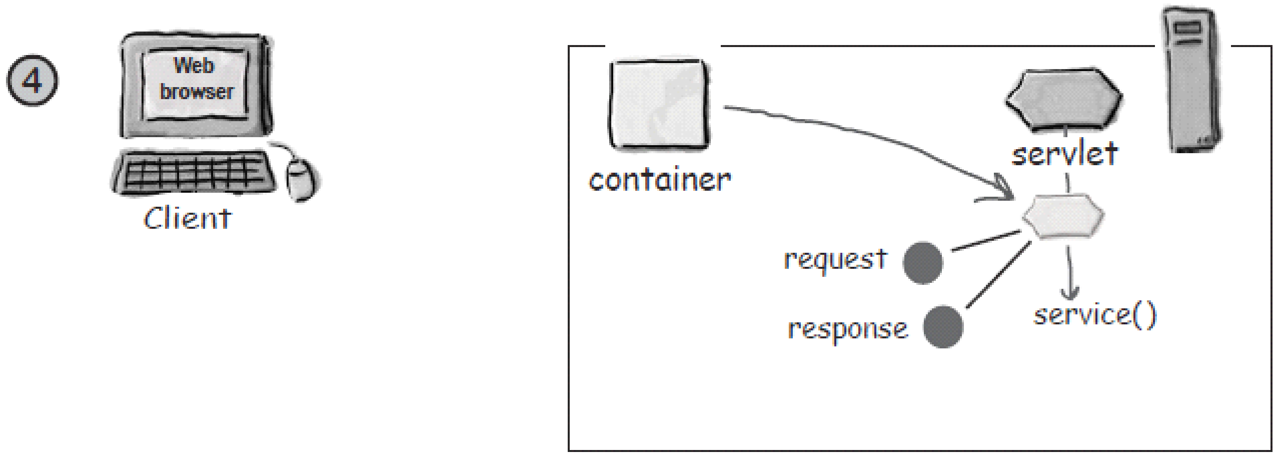
How the Container handles a request 3/6



- The container fi nds the correct servlet based on the URL in the request, creates or allocates a thread for that request, and passes the request and response objects to the servlet thread.

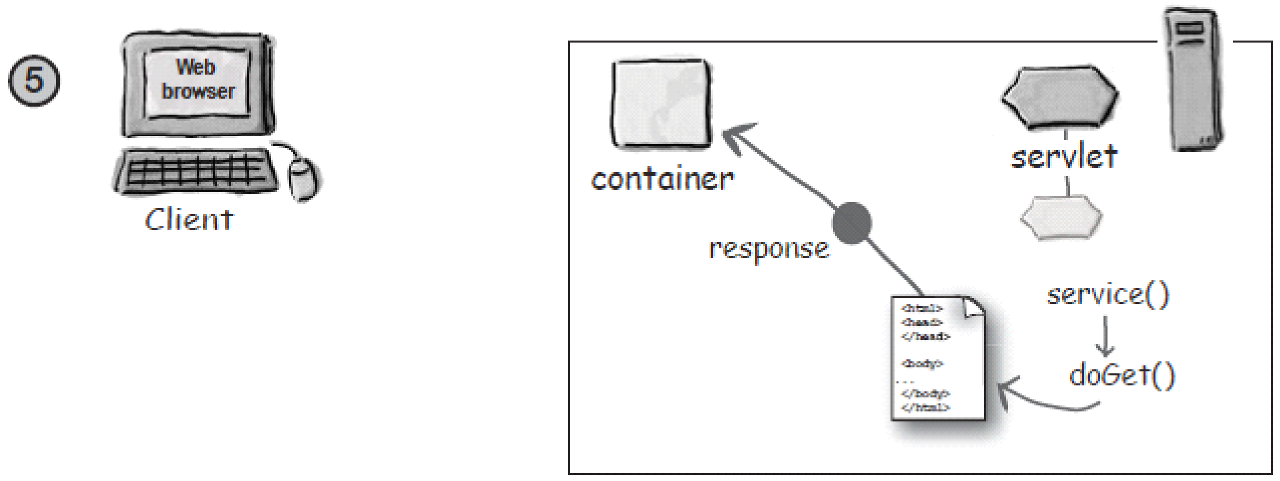
How the Container handles a request 4/6

- Call the servlet’s service() metho . Depending on the type of request,the service() method calls either th doGe () or doPost() method.For this example , w ’ll assume the request was an HTTP GET Method.



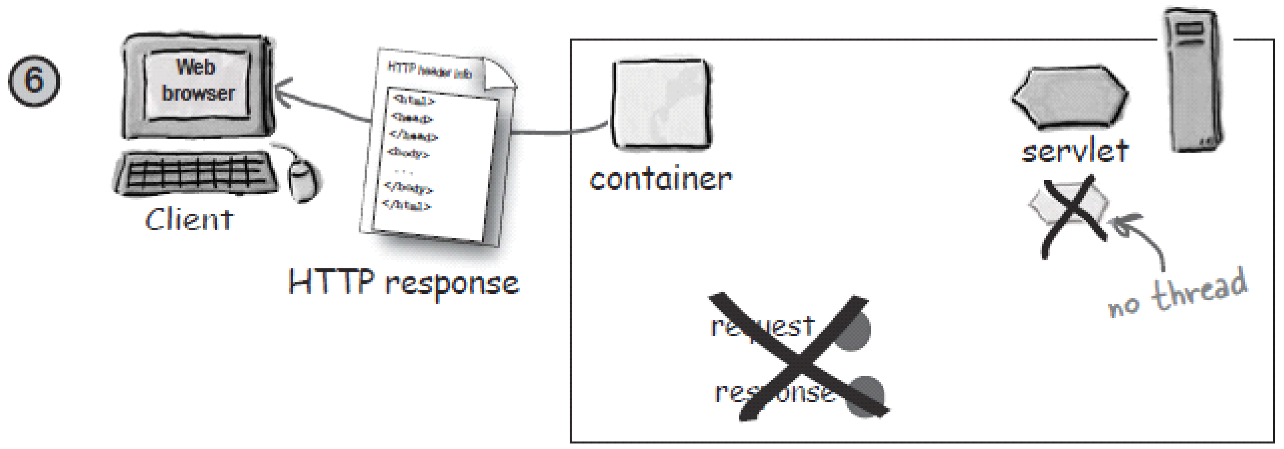
How the Container handles a request 5/6

- The doGet() method generates the dynamic page and stuffs the page into the response object. Remember, the container still has a reference to the response object!



How the Container handles a request 6/6

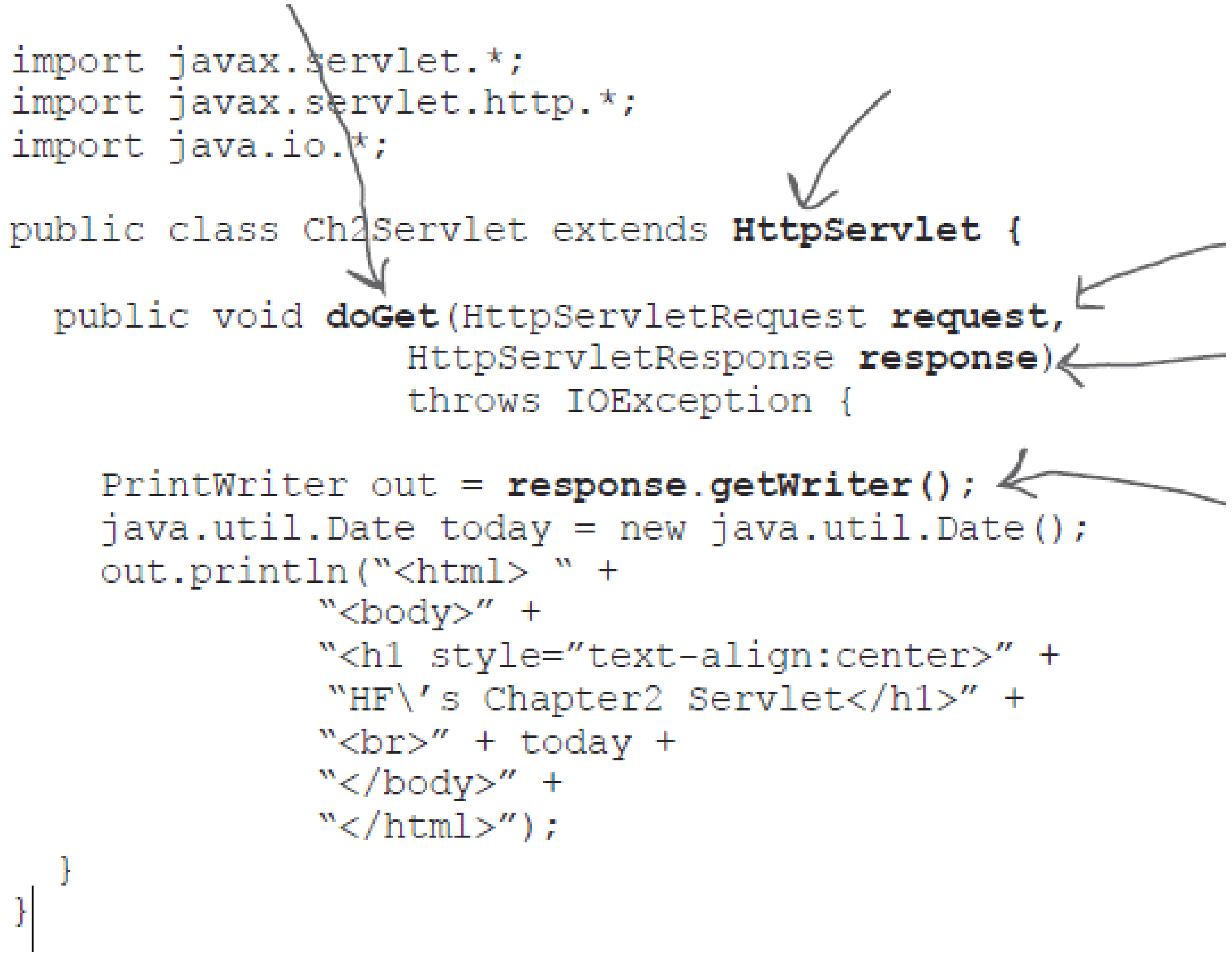
- The thread completes, the container converts the response object into an HTTP response, sends it back to the client, then deletes the request and response objects.



Web App Architecture

- What is a Container and what does it give you?
- How it looks in code (and what makes a servlet)
- Naming servlets and mapping them to URLs using the DD
- First Web App
- How J2EE fits into all this

How it looks in code



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

public class Ch2Servlet extends HttpServlet {

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

    PrintWriter out = response.getWriter();
    java.util.Date today = new java.util.Date();
    out.println(“<html> “ +
              “<body>” +
              “<h1 style=”text-align:center>” +
               “HF\’s Chapter2 Servlet</h1>” +
              “<br>” + today +
              “</body>” +
              “</html>”);
  }
}
```

Two Questions

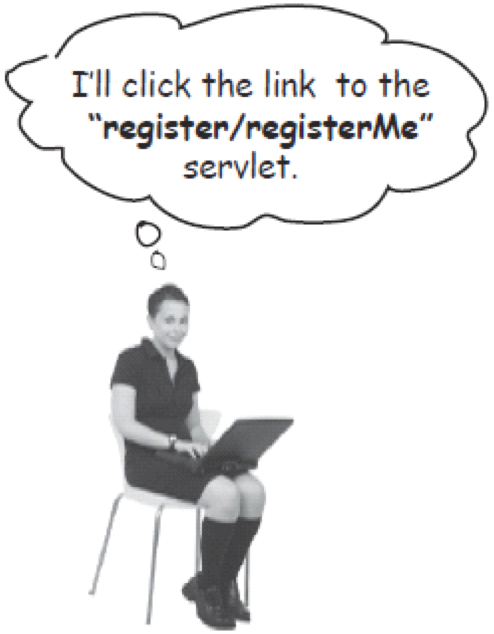
- **Where did the *service()* method come from?**
  - Your servlet inherited it from HttpServlet, which inherited it from GenericServlet which inherited it from...
- **You wimped out on explaining how the container *found* the correct servlet... like, how does a URL relate to a servlet? Does the user have to type in the exact path and class fi le name of the servlet?**
  - we'll take only a quick look on the next few pages

Web App Architecture

- What is a Container and what does it give you?
- How it looks in code (and what makes a servlet)
- Naming servlets and mapping them to URLs using the DD
- First Web App
- How J2EE fits into all this

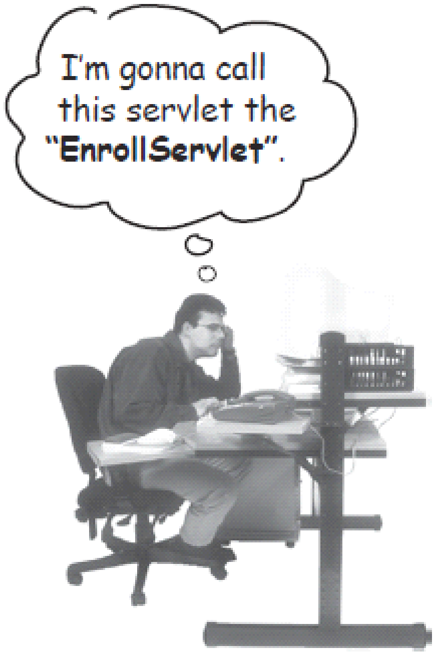
A servlet can have THREE names

1



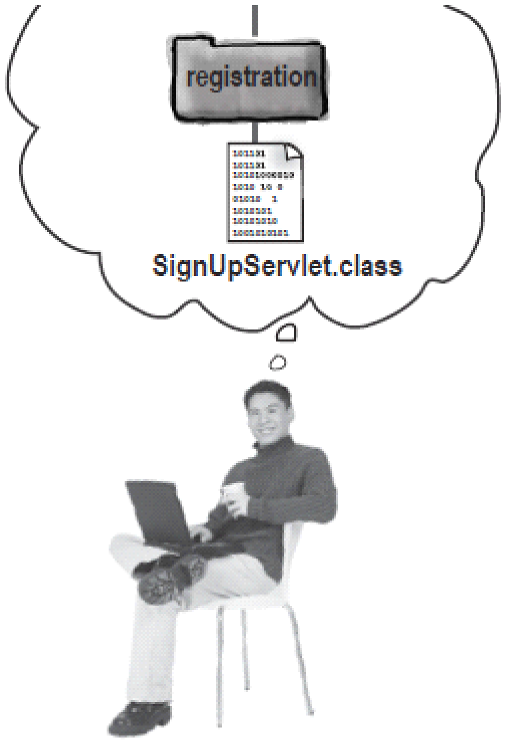
Client-known *URL* name

2



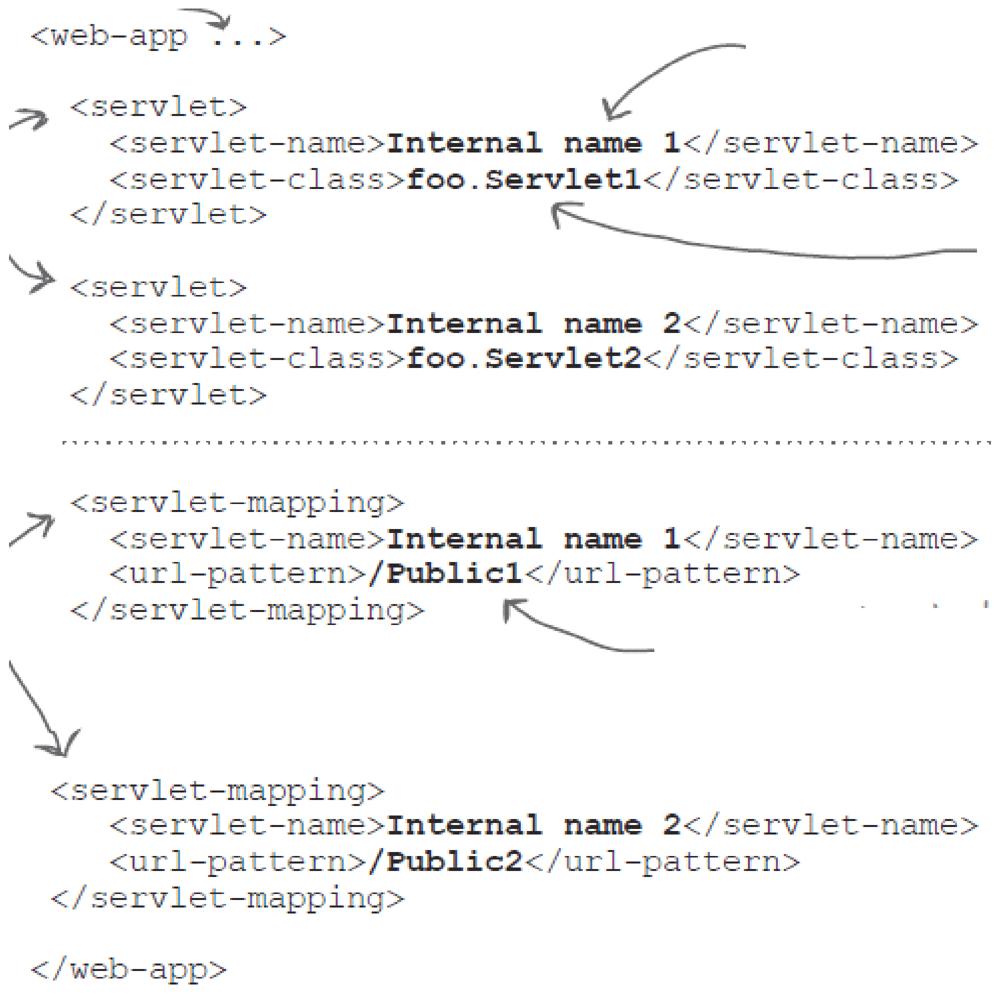
Deployer-known *secret internal* name

3



Actual *file* name

Using the Deployment Descriptor to map URLs to servlets



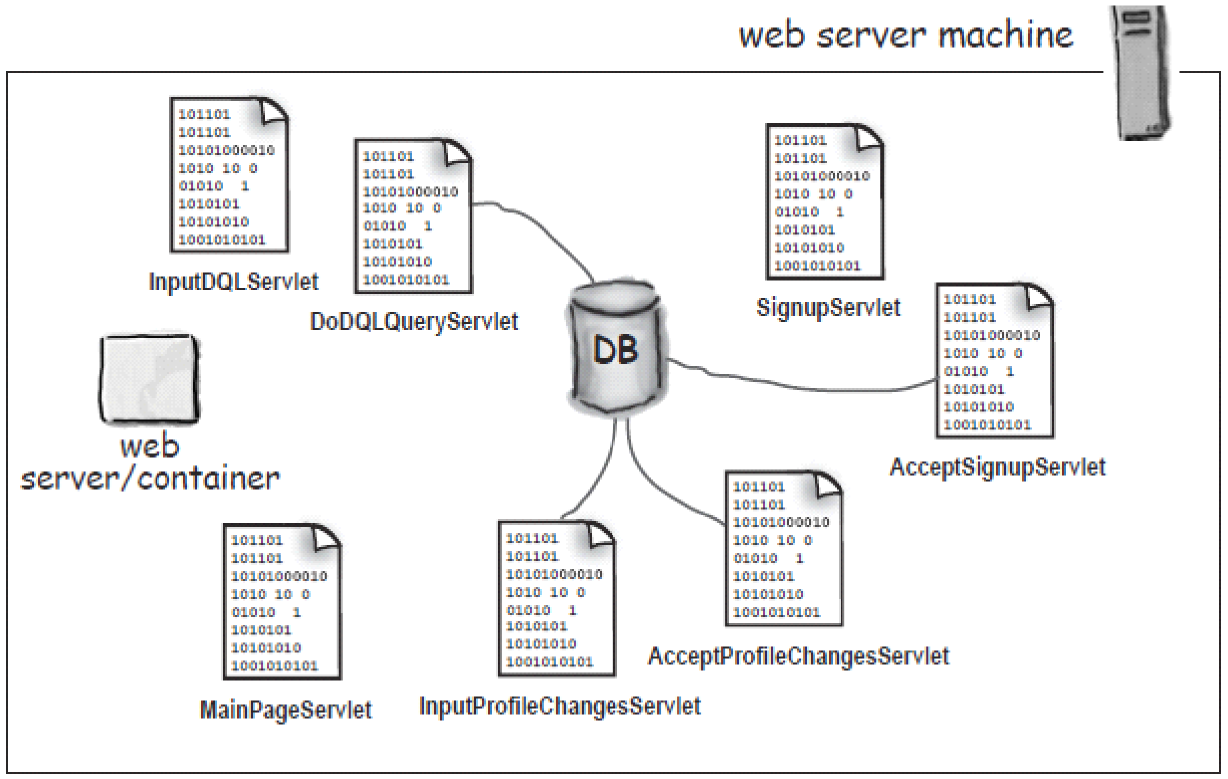
```
<web-app ...>

  <servlet>
    <servlet-name>Internal name 1</servlet-name>
    <servlet-class>foo.Servlet1</servlet-class>
  </servlet>

  <servlet>
    <servlet-name>Internal name 2</servlet-name>
    <servlet-class>foo.Servlet2</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>Internal name 1</servlet-name>
    <url-pattern>/Public1</url-pattern>
  </servlet-mapping>


 <servlet-mapping>
    <servlet-name>Internal name 2</servlet-name>
    <url-pattern>/Public2</url-pattern>
 </servlet-mapping>

</web-app>
```

Web App Architecture

- What is a Container and what does it give you?
- How it looks in code (and what makes a servlet)
- Naming servlets and mapping them to URLs using the DD
- First Web App
- How J2EE fits into all this

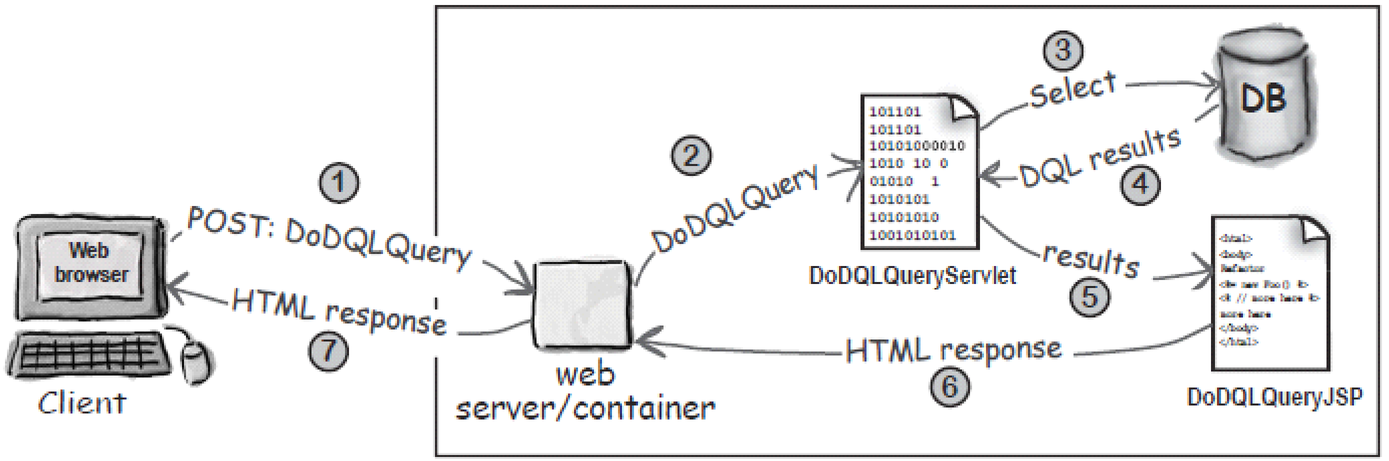
Build a site

- One Servlet for each page



separate out the presentation from the business logic

- Add JSPs



Web App Architecture

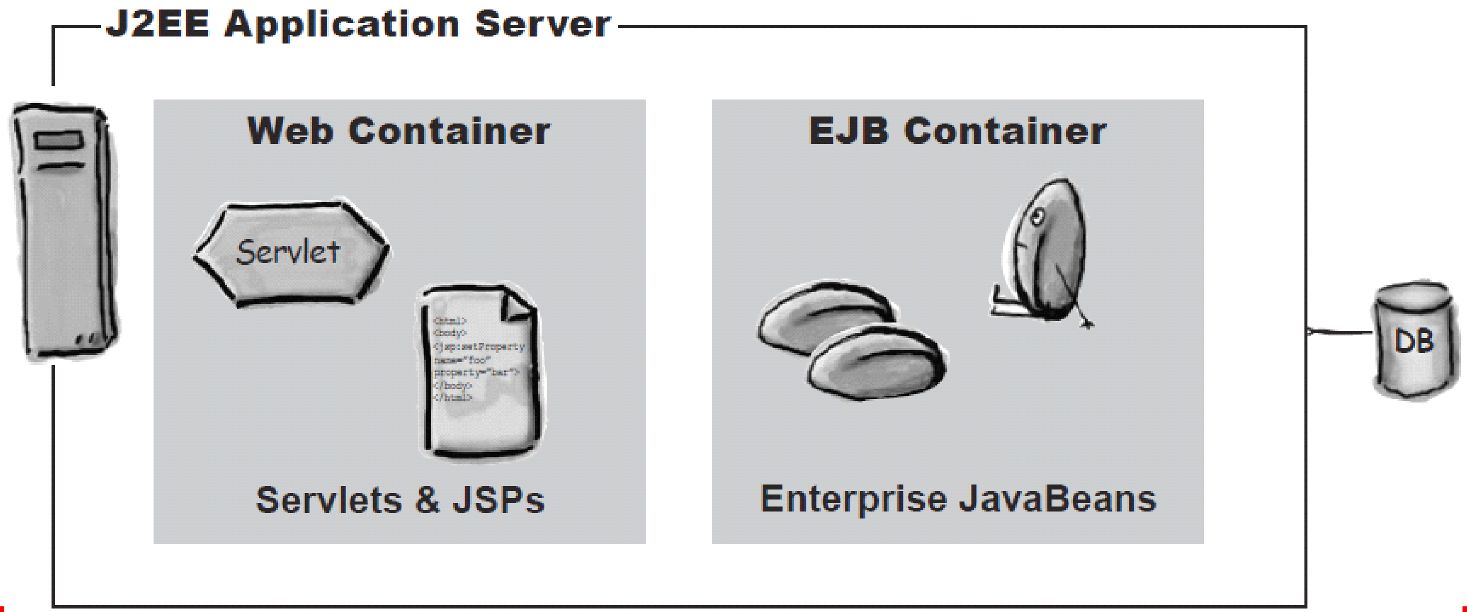
- What is a Container and what does it give you?
- How it looks in code (and what makes a servlet)
- Naming servlets and mapping them to URLs using the DD
- First Web App
- How J2EE fits into all this

How J2EE fits into all this

- The Java 2 Enterprise Edition is kind of a super-spec, including the Servlets 2.4 spec and the JSP 2.0 spec. That’s for the web Container.
- But the J2EE 1.4 spec also includes the Enterprise JavaBean 2.1 specification, for the EJB Container.
- In other words, the web Container is for *web* components (Servlets and JSPs), and the EJB Container is for *business* components.
- A fully-compliant J2EE application server must have *both* a web Container and an EJB Container

How J2EE fits into all this

- Tomcat is a web Container, but NOT a full J2EE application server.
- Some of the most common J2EE servers are BEA's WebLogic, the open source JBoss AS, and IBM's WebSphere.



BULLET POINTS

- The Container gives your web app communications support, lifecycle management, multithreading support, declarative security, and support for JSPs, so that you can concentrate on your own business logic.
- The Container creates a request and response object that servlets (and other parts of the web app) can use to get information about the request and send information to the client.
- A typical servlet is a class that extends HttpServlet and overrides one or more service methods that correspond to HTTP methods invoked by the browser (doGet() doPost(), etc.).

BULLET POINTS

- The deployer can map a servlet class to a URL that the client can use to request that servlet. The name may have nothing to do with the actual class *file* name.
- A fully-compliant J2EE application server must have *both* a web Container and an EJB Container