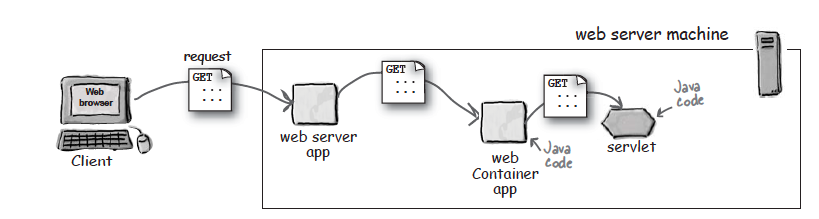
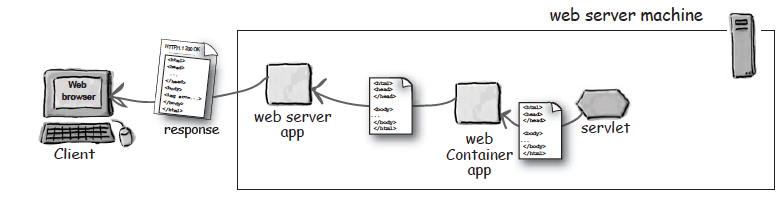
**Chapter 2**

Servlets need help!! When a request comes in, somebody has to instantiate the servlet or at least make a new thread to handle the request. Somebody has to call the servlet's doPost() and doGet() method and those methods have crucial arguments HTTP request and HTTP response objects. Somebody has to get the request and the response to the servlet. Somebody has to manage the life, death and resources of the servlet. That somebody is the WEB Container.

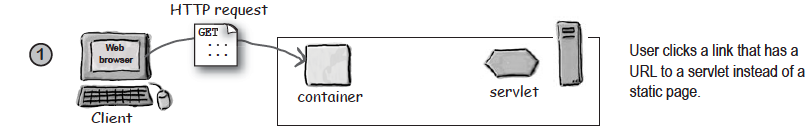
So what is a Container?

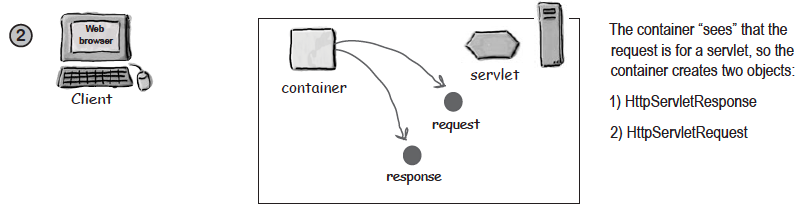
Servlets don't have a main method. They are under the control of another Java application called a Container. Tomcat is an example of a Container. When your web server application (like Apache) gets a request for a *servlet page), the server hands the request not to the servlet itself, but to the Container in which the servlet is deployed*. It’s the Container that gives the servlet the HTTP request and response, and it’s the Container that calls the servlet’s methods (like doPost() or doGet()).

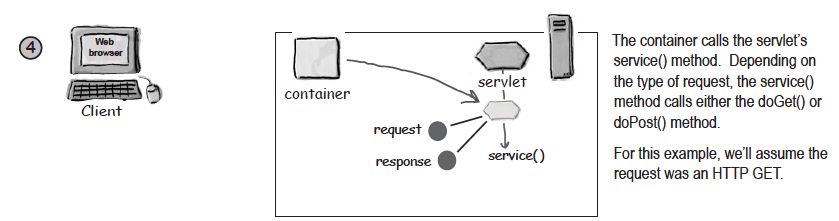
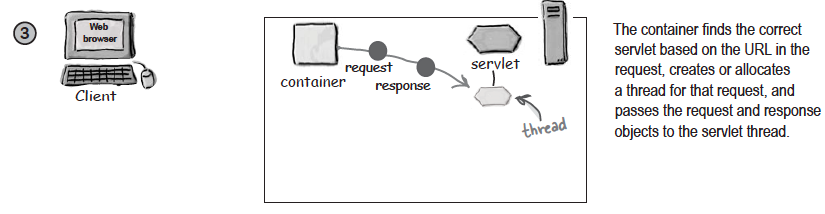


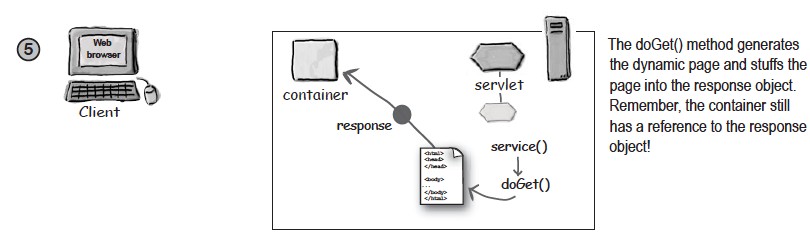


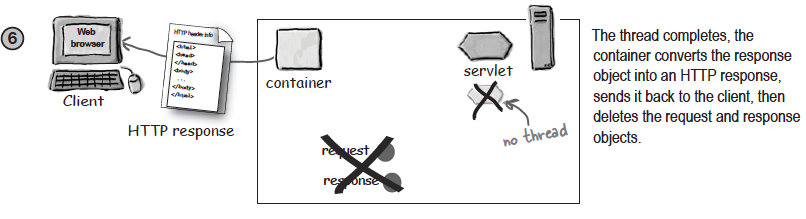
**How the Container handles a request:**

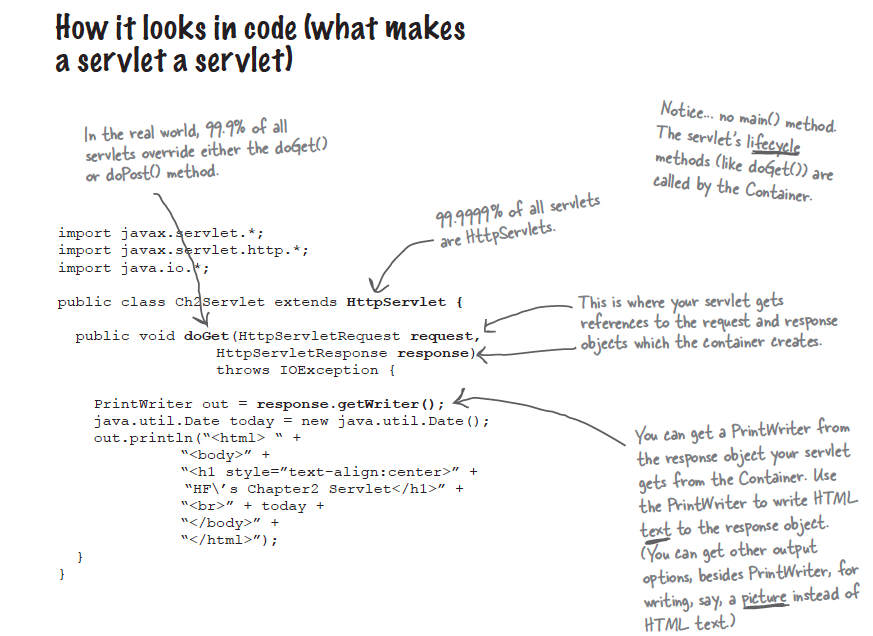












Important to ask, How the container found the Servlet??

Somehow, the URL that comes in as part of the request from the client is *mapped* to a specific servlet on the server. This mapping of URLs to servlets might be handled in a number of different ways, and it’s one of the most fundamental issues you’ll face as a web app developer. The user request must map to a particular servlet, and it’s up to you to understand and (usually) *configure* that mapping.

A Servlet can have THREE names!! OHHH

A servlet has a file path name, obviously, like classes/registration/ SignUpServlet.class (a path to an actual class file). The original developer of the servlet class chose the class name (and the package name that defines part of the directory structure), and the location on the server defines the full path name. But anyone who deploys the servlet can also give it a special deployment name.

A deployment name is simply a secret internal name that doesn’t have to be the same as the class or file name. It can be the same as the servlet class name (registration.SignUpServlet) or the relative path to the class file(classes/registration/SignUpServlet.class), but it can also be something completely different (like EnrollServlet). Finally, the servlet has a public URL name—the name the client knows about. In other words, the name coded into the HTML so that when the user clicks a link that’s supposed to go to that servlet, this public URL name is sent to the server in the HTTP request.

One should think, why the hell we need this extra overhead??

Answer : Mapping Servlet names improves your app's flexibility and security.

BUT HOW??

Think about it.

So you’ve hard-coded the real path and file name into all the JSPs and other HTML pages that use that servlet. Great. Now what happens when you need to reorganize your application, and possibly move things into different directory structures? *Do you really want to force everyone who uses that servlet to know (and forever follow) that same directory structure?*

By mapping the name instead of coding in the real file and path name, you have the flexibility to move things around without having the maintenance nightmare of tracking down and changing client code that refers to the old location of the servlet files.

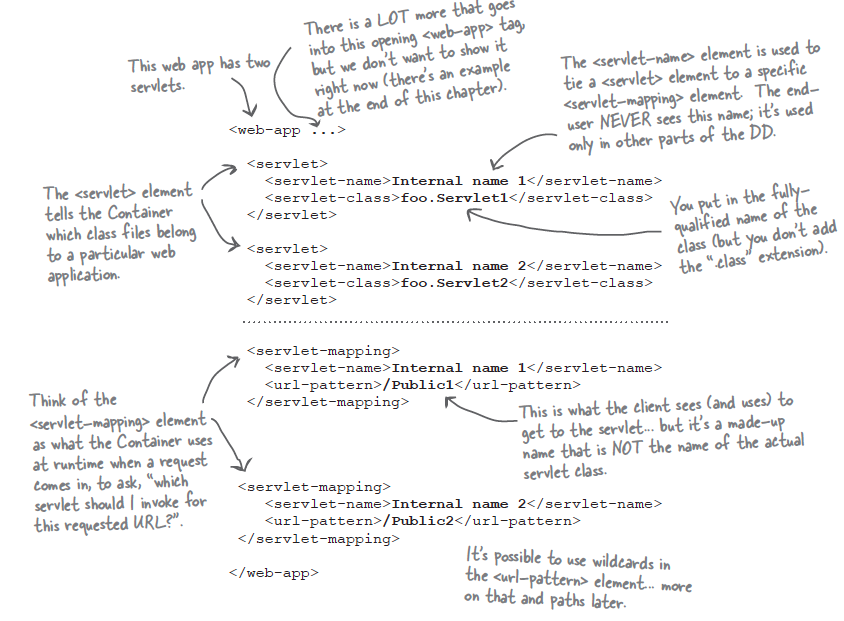
And what about security? Do you really want the client to know exactly how things are structured on your server? Do you want them to, say, attempt to navigate directly to the servlet without going through the right pages or forms? Because if the end-user can see the *real* path, she can type it into her browser and try to access it directly.

**Deployment Descriptor**

When you deploy your servlet into your web Container, you’ll create a fairly simple XML document called the Deployment Descriptor (DD) to tell the Container how to run your servlets and JSPs. Although you’ll use the DD for more than just mapping names, you’ll use two XML elements to map URLs to servlets—one to map the client-known public URL name to your own internal name, and the other to map your own internal name to a fully-qualified class name.

1. <servlet> maps internal name to fully qualified class name.

2. <servlet-mapping> maps internal name to public url.



But wait, There is more you can do with the DD !!!!

Besides mapping URLs to actual servlets, you can use the DD to customize other aspects of your web application including security roles, error pages, tag libraries, initial configuration information etc etc.

Don't worry about the complete details yet. The crucial point to know that the DD gives you a way to declaratively modify your application without changing the source code. COOL NAA !!!