Servlet basics

Nguyễn Hoàng Anh
Trương Phước Lộc
Hồ Tuấn Thanh

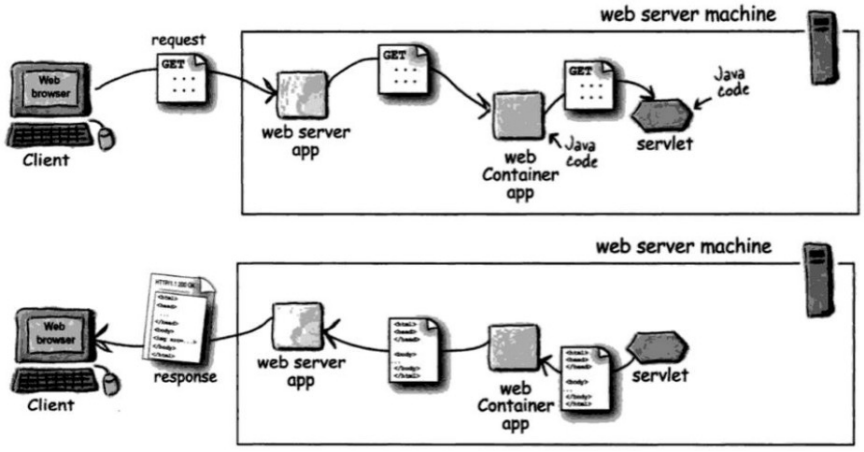
1

Agenda

1. Container
2. Servlet
3. Story: build a site
4. Demo
5. Notes

2

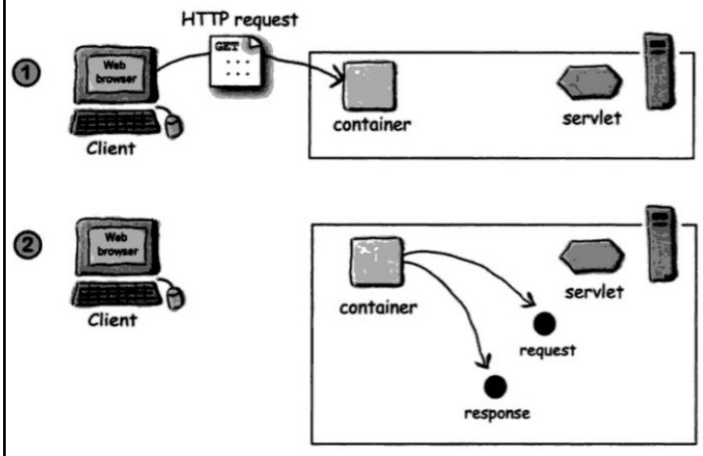
1.1 What is a container?



1.2 What does the Container give you?

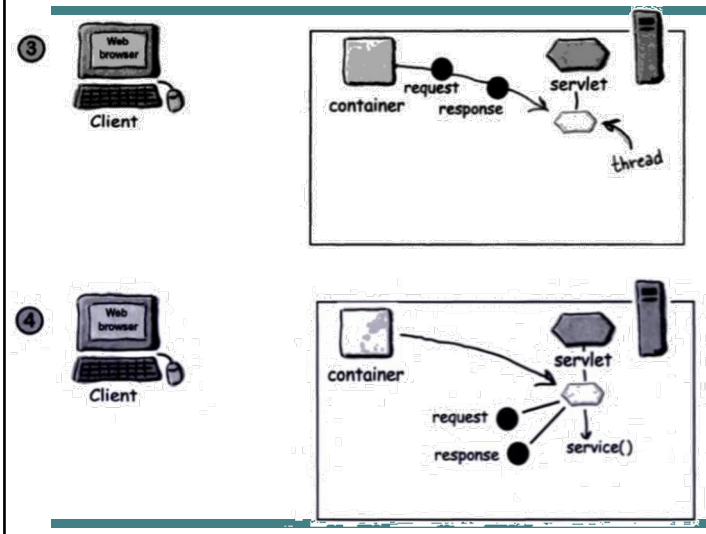
- Communicate support
- Lifecycle management
- Multithreading support
- Declarative security
- JSP support

1.3 How the Container handles a request



1. User clicks a link that has a URL to a servlet instead of a static page.
2. The container "sees" that the request is for a servlet, so the container creates two objects:
   1) HttpServletResponse
   2) HttpServletRequest

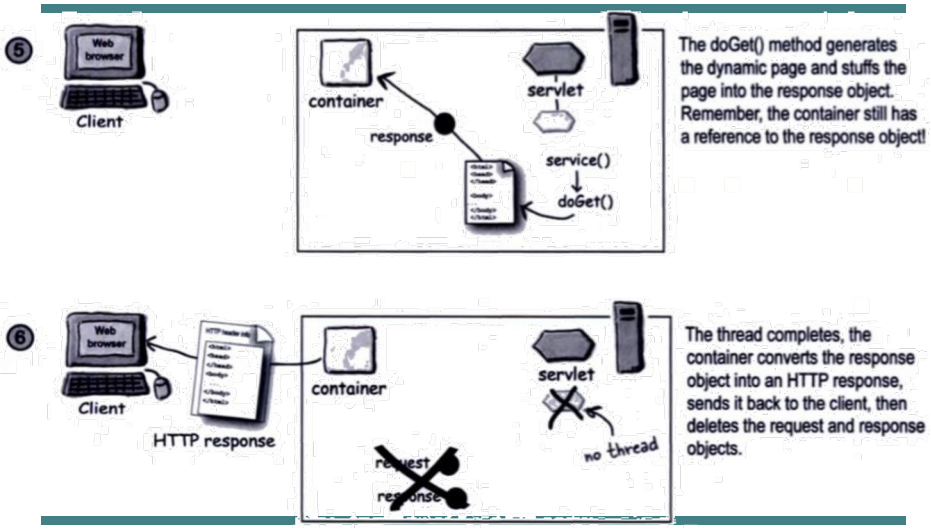
1.3 How the Container handles a request



3. The container finds 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.
4. The container calls the servlet's service() method. Depending on the type of request, the service() method calls either the doGet() or doPost() method.

   For this example, we'll assume the request was an HTTP GET.

1.3 How the Container handles a request



Agenda

1. Container
2. Servlet
3. Story: build a site
4. Demo
5. Notes

2. Servlet

- **Servlets** provide a component-based, platform-independent method for building Web-based applications.
- **Servlets** have access to the **entire family of Java APIs**, including the JDBC API to access enterprise databases

2.1 How it looks in code

In the real world, 99.9% of all servlets override either the doGet() or doPost() method.

Notice... no main() method. The servlet's lifecycle methods (like doGet()) are called by the Container.

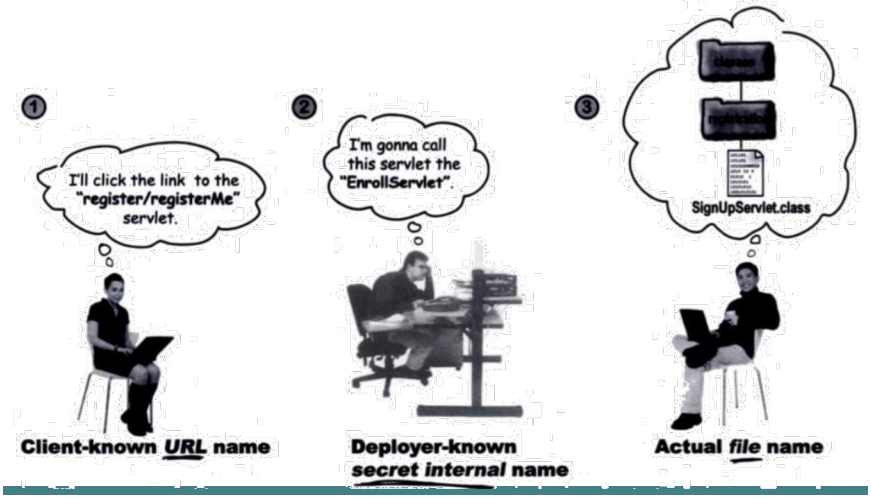
99.9999% of all servlets are HttpServlets.

```
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>");
  }
}
```

This is where your servlet gets references to the request and response objects which the container creates.

You can get a PrintWriter from the response object your servlet gets from the Container. Use the PrintWriter to write HTML text to the response object. (You can get other output options, besides PrintWriter, for writing, say, a picture instead of HTML text.)

2.2 A servlet can have THREE names



2.2 A servlet can have THREE names

1. **<servlet>**
   maps internal name to fully-qualified class name
2. **<servlet-mapping>**
   maps internal name to public URL name

This web app has two servlets.

There is a LOT more that goes into this opening <web-app> tag, but we don't want to show it right now (there's an example at the end of this chapter).

<web-app ...>

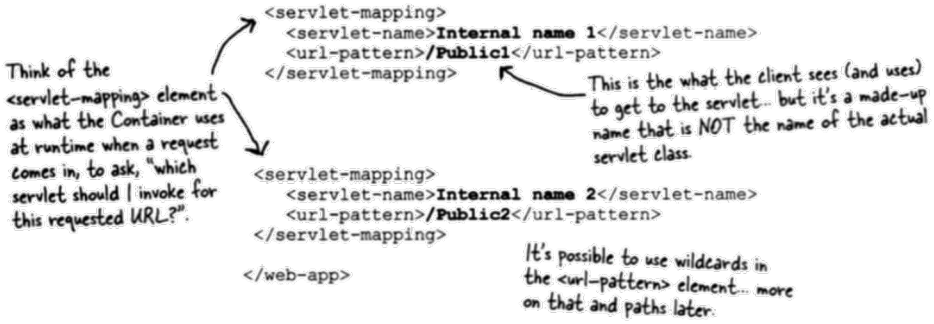
The <servlet-name> element is used to tie a <servlet> element to a specific <servlet-mapping> element. The end-user NEVER sees this name; it's used only in other parts of the DD.

The <servlet> element tells the Container which class files belong to a particular web application.

```
<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>
```

You put in the fully-qualified name of the class (but you don't add the ".class" extension).

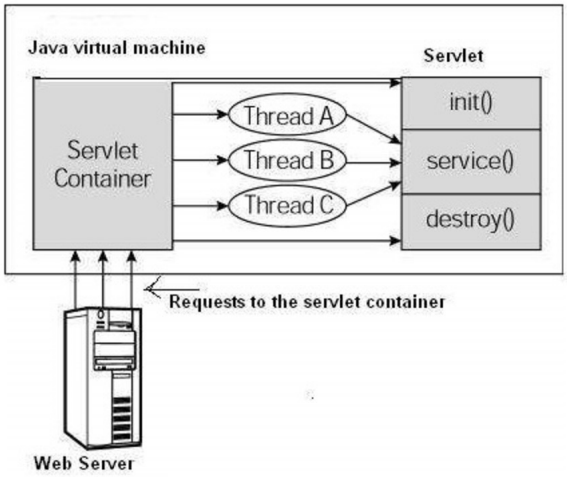
2.2 A servlet can have THREE names



2.3 Parameters

- String str = request.getParameter(“nameInput”)
- String []s = request.getParameterValues(“NameInput”)
- Map<String, String[]> m = request.getParameterMap()
- Enumeration<String> e = request.getParameterNames()

2.4 Life Cycle



2.4 Life Cycle

- The servlet is initialized by calling the **init()** method.
- The servlet calls **service()** method to process a client's request.
- The servlet is terminated by calling the **destroy()** method.
- Finally, servlet is garbage collected by the garbage collector of the JVM.

2.4 The init() method

- The init method is called only once.
- It is called only when the servlet is created, and not called for any user requests afterwards.

```
public void init() throws ServletException {
    //Initialization code...
}
```


2.4 The service() method

- The service() method is the main method to perform the actual task.
- The servlet container (i.e. web server) calls the service() method to
  - handle requests coming from the client(browsers)
  - write the formatted response back to the client.
- Each time the server receives a request for a servlet, the server spawns a new thread and calls service. The service() method checks the HTTP request type (GET, POST, PUT, DELETE, etc.) and calls doGet, doPost, doPut, doDelete, etc. methods as appropriate.

2.4 The destroy() method

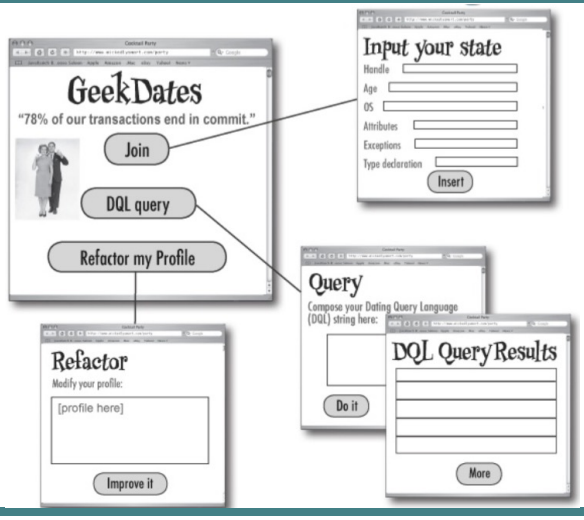
- The destroy() method is called only once at the end of the life cycle of a servlet.
- This method gives your servlet a chance to close database connections, halt background threads, write cookie lists or hit counts to disk, and perform other such cleanup activities.

Agenda

1. Container
2. Servlet
3. Story: build a site
4. Demo
5. Notes

3. Bob builds a Matchmaking Site

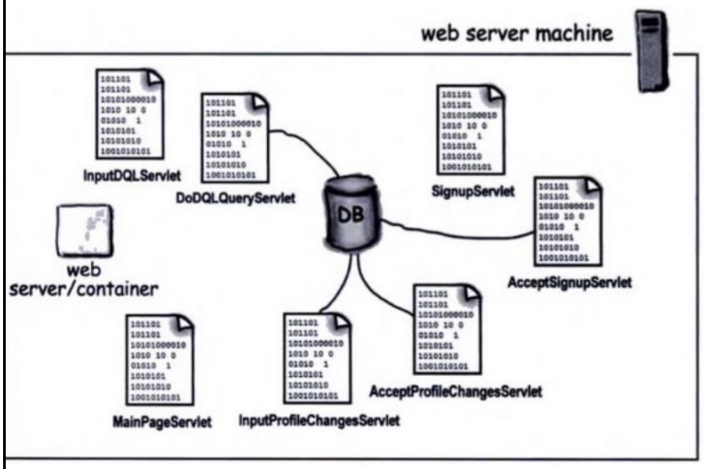
3. Bob builds a Matchmaking Site



3. … one for each page

```
// import statements

public class DatingServlet extends HttpServlet {

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

    // business logic goes here, depending
    // on what this servlet is supposed to do
    // (write to the database, do the query, etc.)

    PrintWriter out = response.getWriter();

    // compose the dynamic HTML page
    out.println( "something really ugly goes here");
  }
}
```

3. …



The servlet does whatever it needs to do to process the request (like insert or search the database) and returns the HTML page in the HTTP response.

All of the business logic AND the client HTML page response is inside the servlet code.

Agenda

1. Container
2. Servlet
3. Story: build a site
4. Demo
5. Notes

4. Demo

Agenda

1. Container
2. Servlet
3. Story: build a site
4. Demo
5. Notes

5.1 The deployment descriptor (DD)

The deployment descriptor (DD), provides a "declarative" mechanism for customizing your web applications without touching source code!

DD Benefits

- Minimizes touching source code that has already been tested.
- Lets you fine tune your app's capabilities, even if you don't *have* the source code.
- Lets you adapt your application to different resources (like databases), without having to recompile and test any code.
- Makes it easier for you to maintain dynamic security info like access control lists and security roles.
- Lets non-programmers modify and deploy your web applications while *you* can focus on the more interesting things. Like how appropriate your wardrobe isn't for a trip to Hawaii.

5.2 Servlet

- Form Data
- Client request
- Server response
- Filters
- Exceptions
- Cookies
- Session
- Database
- …

Hỏi đáp

Tài liệu tham khảo

- Tài liệu giảng dạy – ThS. Nguyễn Hoàng Anh, ĐH. Khoa học tự nhiên
- http://www.tutorialspoint.com/servlets/