Programming Advanced Java

WEEK 10 - Servlets

PXL DIGITAL

Goals

The **junior-colleague**

- can explain what a Servlet is
- can explain how Spring Boot uses Servlet technology
- can explain and use the lifecycle moments of a Servlet
- can explain the usage of @WebServlet, @WebFilter and @WebListener
- can explain the usage of HttpSession
- can create an HTTPServlet to handle requests
- can use cookies and HTTPSession in a web application
- can use WebClient to send a simple HTTP request
- can implement a @WebFilter
- can implement a @WebListener

Overview

1. What is a Servlet
2. Demo: DateTimeServlet
3. @ServletComponentScan
4. Servlet class hierarchy
5. Lifecycle of a Servlet
6. @WebServlet, @WebFilter and @WebListener
7. Exercises

What is a Servlet?

a Servlet is a class that handles requests, processes them and reply back with a response.

For example, we can use a Servlet to collect input from a user through an HTML form, query records from a database, and create web pages dynamically.

Servlets are under the control of another Java application called a ***Servlet Container.*** When an application running in a web server receives a request, the Server hands the request to the Servlet Container – which in turn passes it to the target Servlet.

Java servlets typically run on the HTTP protocol.

DateTimeServlet

Find the code here: https://github.com/custersnele/PAJ_servlets.git

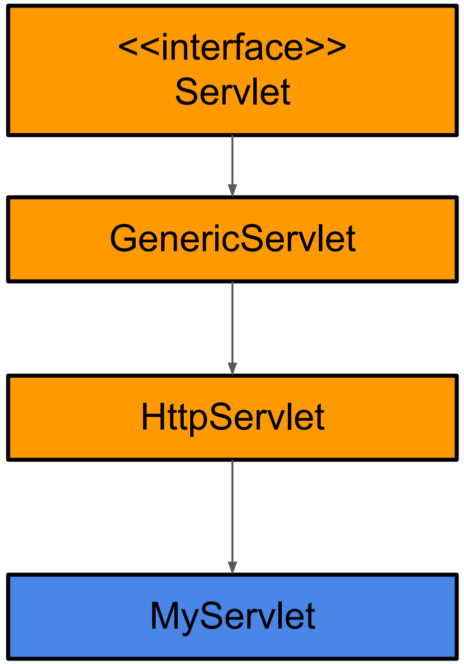
@ServletComponentScan

```
@SpringBootApplication
@ServletComponentScan
public class ServletDemoApplication { … }
```

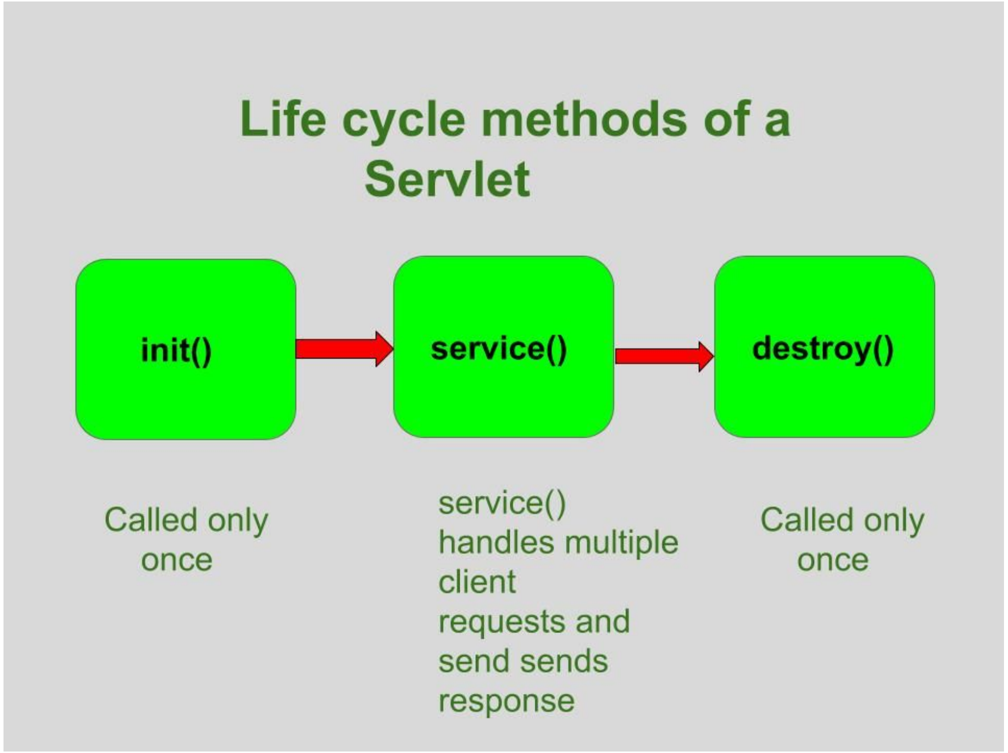
The *@ServletComponentScan annotation* automatically register the following Servlet components for **embedded web servers**. This annotation supports the following *Servlet 3.0 annotations*:

1. `@WebServlet` annotation.
2. The `@WebFilter`.
3. `@WebListener` annotation

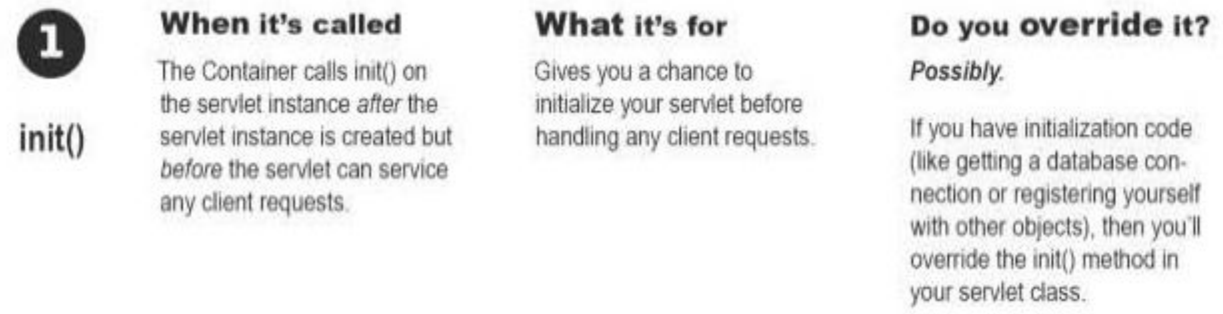
Servlet class hierarchy



Lifecycle moments of a Servlet

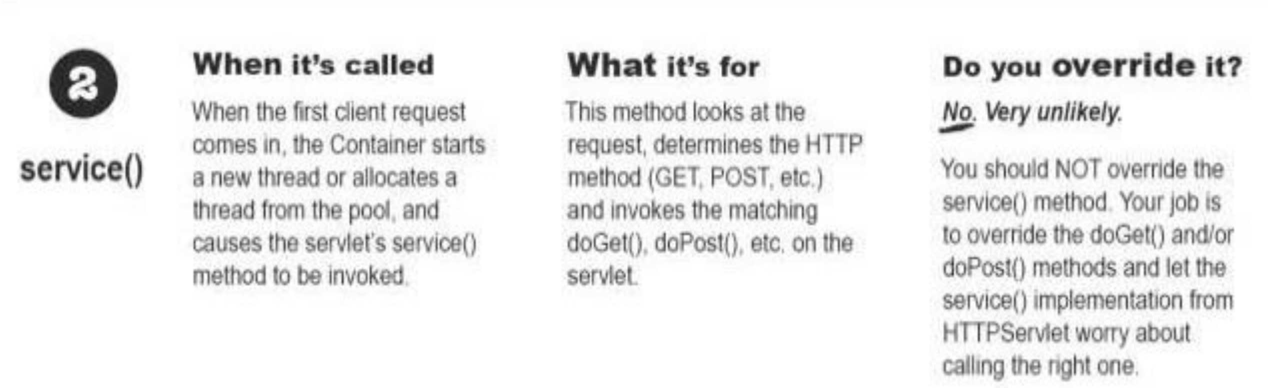

Life cycle methods of a Servlet
init()
service()
destroy()
Called only once
service() handles multiple client requests and send sends response
Called only once

Lifecycle moments of a servlet



	When it's called	What it's for	Do you override it?
1 init()	The Container calls init() on the servlet instance *after* the servlet instance is created but *before* the servlet can service any client requests.	Gives you a chance to initialize your servlet before handling any client requests.	***Possibly.*** If you have initialization code (like getting a database con-nection or registering yourself with other objects), then you'll override the init() method in your servlet class.

Lifecycle moments of a servlet



2

service()

When it's called

When the first client request comes in, the Container starts a new thread or allocates a thread from the pool, and causes the servlet's service() method to be invoked.

What it's for

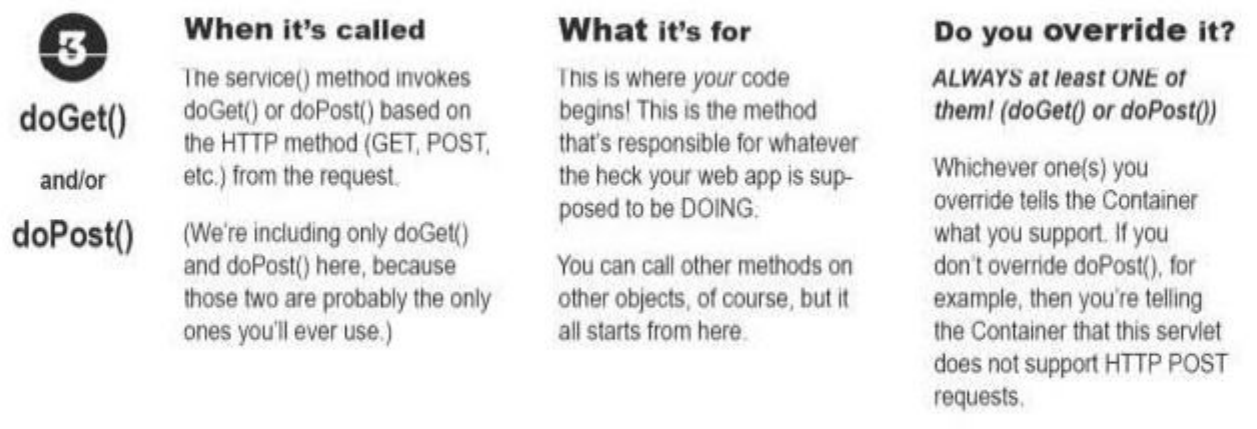
This method looks at the request, determines the HTTP method (GET, POST, etc.) and invokes the matching doGet(), doPost(), etc. on the servlet.

Do you override it?

No. Very unlikely.

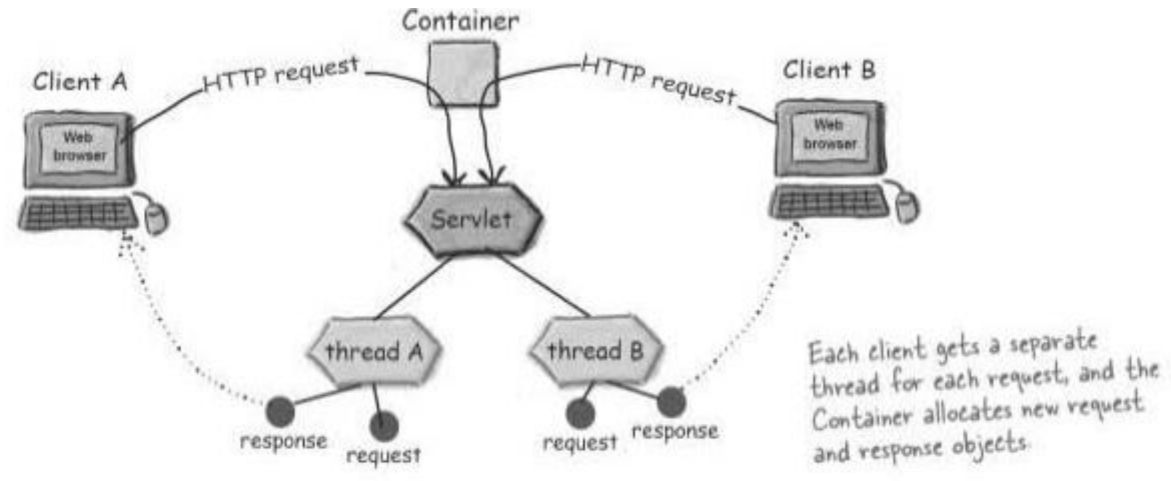
You should NOT override the service() method. Your job is to override the doGet() and/or doPost() methods and let the service() implementation from HTTPServlet worry about calling the right one.

Lifecycle moments of a servlet

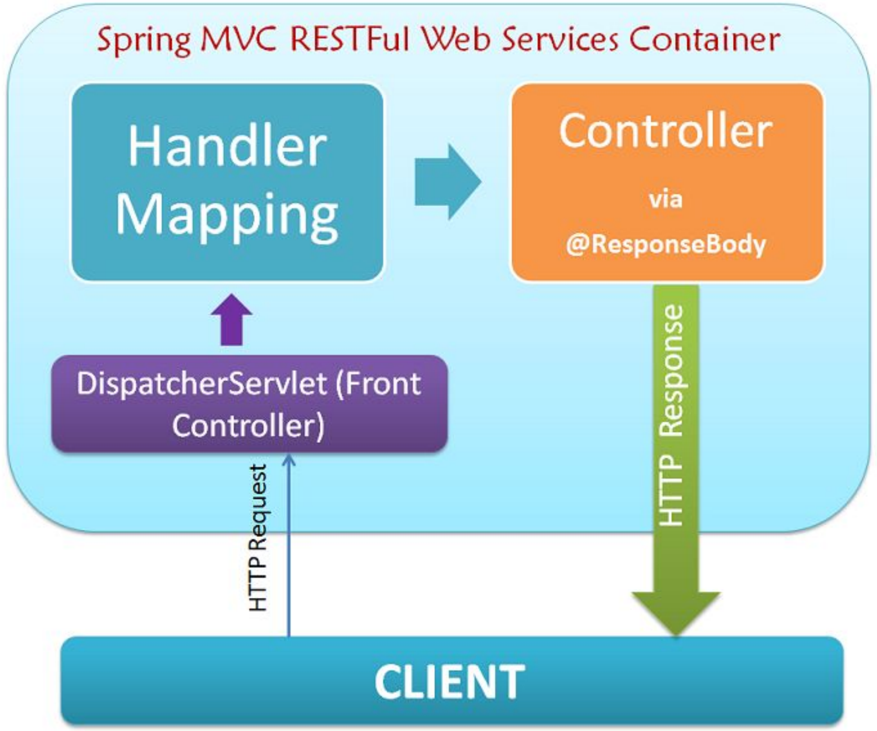


	When it's called	What it's for	Do you override it?
3 doGet() and/or doPost()	The service() method invokes doGet() or doPost() based on the HTTP method (GET, POST, etc.) from the request. (We're including only doGet() and doPost() here, because those two are probably the only ones you'll ever use.)	This is where *your* code begins! This is the method that's responsible for whatever the heck your web app is supposed to be DOING. You can call other methods on other objects, of course, but it all starts from here.	***ALWAYS at least ONE of them! (doGet() or doPost())*** Whichever one(s) you override tells the Container what you support. If you don't override doPost(), for example, then you're telling the Container that this servlet does not support HTTP POST requests.

Each request runs in a separate thread



Spring DispatcherServlet


Spring MVC RESTFul Web Services Container
Handler Mapping
Controller
via
@ResponseBody
DispatcherServlet (Front Controller)
HTTP Request
HTTP Response
CLIENT

Second demo

SelectBeerServlet (start with opening http://localhost:8085/beer_selection.html in a browser)

Using Filter

```
@WebFilter(urlPatterns = { "/*" })
public class HeaderLogFilter implements Filter {

  private static final Logger LOGGER = LoggerFactory.getLogger(HeaderLogFilter.class);

  @Override
  public void doFilter(ServletRequest request, ServletResponse response,
                       FilterChain chain) throws IOException, ServletException {
   …
  }
}
```

Implementing sessions using HttpSession

```
@WebServlet(value="/TestSession")
public class TestSessionServlet extends HttpServlet {

  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
     HttpSession session = req.getSession();

     PrintWriter writer = resp.getWriter();
     writer.println("Session ID: " + session.getId());
     …
  }
}
```

Using Listener

```
@WebListener
public class ExampleContextListener implements ServletContextListener {

  @Override
  public void contextInitialized(ServletContextEvent servletContextEvent) {
     System.out.println("ServletDemo starting up!");
  }

  @Override
  public void contextDestroyed(ServletContextEvent servletContextEvent) {
     System.out.println("ServletDemo shutting down!");
  }
}
```

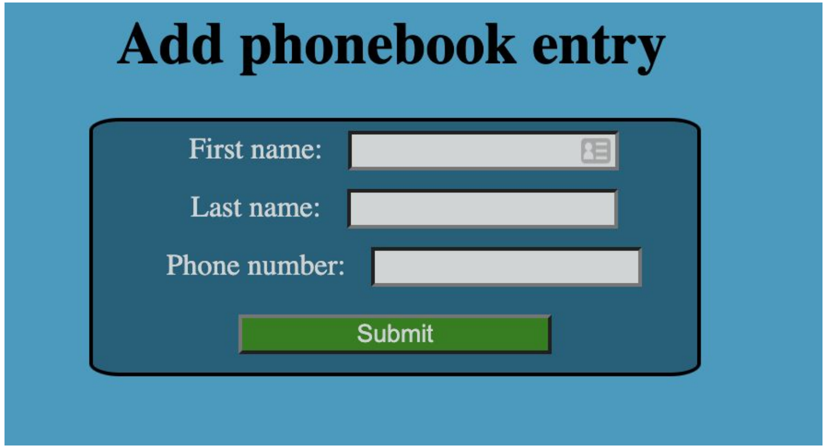
Using Listener

The **`@WebListener`** annotation is used to register a class as a listener of a web application. The annotated class must implement one or more of the following interfaces:

- `javax.servlet.ServletContextListener`
- `javax.servlet.ServletContextAttributeListener`
- `javax.servlet.ServletRequestListener`
- `javax.servlet.ServletRequestAttributeListener`
- `javax.servlet.http.HttpSessionListener`
- `javax.servlet.http.HttpSessionAttributeListener`

Exercise

The html-page phonebook_add.html is given. Create a Servlet to handle the form's POST request. The data is saved in a (in-memory) database. Make sure the phonenumber is unique. Create a Servlet to generate an overview of all phonebook entries (html).


Add phonebook entry
First name:
Last name:
Phone number:
Submit