Advanced Java

Agenda - Servlets

- Tomcat Architecture
- Class hierarchy
- Life cycle
- Servlet config
- web.xml
- Init-params
- Load-on-startup
- Servlet communication/navigation

Servlet Hierarchy

- javax.servlet.Servlet interface
  - void init(ServletConfig config) throws ServletException;
  - void service(ServletRequest req, ServletResponse resp) throws IOException, ServletException;
  - void destroy();
- GenericServlet is abstract class that represents protocol-independent servlet.
- HttpServlet represent http based servlet class and user defined servlet classes are inherited from it.
  - Overrides service() method.
  - Provide doGet(), doPost(), doPut(), doDelete(), doHead(), doTrace(), doOptions()
  - Docs: https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServlet.html

HttpServletRequest interface

- https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServletRequest.html
- HttpServletRequest interface inherited from ServletRequest interface.
- It is created by webserver for each request and represent http req body & header.
- Request Parameters

  - Data from submitted HTML form (in previous page) in request body (POST) or URL (GET).
  - String paramValue = req.getParameter("param-name");
    - Used with textbox, radiobutton, drop-down, ...
  - String[] paramValues = req.getParameterValues("param-name");
    - Used with checkboxes, listbox, ...
- Request Headers
  - String headerValue = req.getHeader("header-name");
    - e.g. String value = req.getHeader("Content-Type");
  - String[] headerValues = req.getHeaderValues("header-name")
- Request upload
  - InputStream in = req.openInputStream();

HttpServletResponse

- https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpServletResponse.html
- HttpServletResponse interface inherited from ServletResponse interface.
- It is created by webserver for each request and represent http response body & header.
- Response content type
  - resp.setContentType("text/html"); --> Sets response header Content-Type
- Response send error -- return HTTP status code with message
  - resp.sendError(403);
  - resp.sendError(HttpServletResponse.SC_FORBIDDEN, "Fobidden resource");
- Response download/image
  - OutputStream out = resp.openOutputStream();
  - Need to setup content-type for download (application/octet-stream) or image (image/png) or audio.

Servlet config

- Each servlet is associated with a config object -- ServletConfig.
- It stores information about servlet like name, init parameters, url-patterns, load-on-startup, etc.
- This can be accessed in the servlet class in init() method (as argument) or other methods using `ServletConfig cfg = this.getServletConfig();`.
- Note that all servlet classes are indirectly inherited from ServletConfig, so ServletConfig methods are directly available on servlet object (this).

Init parameters

- ServletConfig may have some configurable values like JDBC url, username, password, etc.
- They can be attached to config using init-params.

```java
@WebServlet(value="/hi",
    initParams = {
        @WebInitParam(name="color", value="green"),
        @WebInitParam(name="greeting", value="Hi")
    },
    name = "DAC")
public class DacServlet extends HttpServlet {
    // ...
}
```

- These init params can be accessed in servlet class using getInitParameter() method.

```java
ServletConfig cfg = this.getServletConfig();
String color = cfg.getInitParameter("color"); // returns "green"
```

```java
String message = this.getInitParameter("greeting"); // returns "hi"
```

Load On Startup

- By default servlet is loaded and initialized on first request. If init() includes heavy processing, the first request will execute slower.
- Alternatively servlets can be loaded while starting the web server. This can be done by marking servlet as load-on-startup.

```java
@WebServlet(value="/hi",
    loadOnStartup = 1,
    name = "DMC")
public class DmcServlet extends HttpServlet {
    // ...
}
```

- The number after "loadOnStartup" indicate the sequence of loading the servlets if multiple servlets are marked as load-on-startup. If multiple servlets load-on-startup number is same, web container arbitrarily choose the sequence.
- Servlet config in web.xml

```xml
<servlet>
    <servlet-name>DAC</servlet-name>
    <servlet-class>com.sunbeam.DacServlet</servlet-class>
    <init-param>
        <param-name>color</param-name>
        <param-value>pink</param-value>
    </init-param>
    <init-param>
        <param-name>greeting</param-name>
        <param-value>Good Afternoon</param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>DAC</servlet-name>
    <url-pattern>/hi</url-pattern>
</servlet-mapping>
```

Servlet communication/navigation

- HTTP redirection
  - resp.sendRedirect("url");
  - Can navigate from one web component to another web component (within or outside the current application).
  - resp.sendRedirect() sends a minimal response to the client which contain status code 302 and location (url) of next web component.
  - The client (browser) receives this response and send new request to the next web component.
  - In browser, URL is modified (i.e. client is aware of navigation).
- RequestDispatcher
  - https://docs.oracle.com/javaee/7/api/javax/servlet/RequestDispatcher.html
  - RequestDispatcher rd = req.getRequestDispatcher("url");
    - url is w.r.t. current request.
  - RequestDispatcher rd = ctx.getRequestDispatcher("/url");
    - url is w.r.t. application (context) root.
- RequestDispatcher – forward()
  - rd.forward(req, resp);
  - Forwards the current request to the given web component (within application only).
  - The next web component produces final response (to be sent to the client).
  - Note that new request & response objects are not created.
  - In browser, URL is not modified (i.e. client is not aware of navigation).
  - Faster than HTTP redirection.
  - Used in Spring MVC by the controller.
- RequestDispatcher – include()
  - rd.include(req, resp);
  - Calling given web component (within application only) to produce partial response.
  - The final response is generated by the current (first) web component itself.
  - Note that new request & response objects are not created.
  - In browser, URL is not modified (i.e. client is not aware of navigation).
  - Slower than RequestDispatcher – forward().
  - Mostly used for rendering header/footer in dynamic web pages.