Advanced Java

Agenda

- Servlet navigation
- PRG design pattern
- RequestDispatcher vs Redirection
- State management
- Cookies
- Session

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.

PRG design pattern

- The Post/Redirect/Get (PRG) design pattern is a common web development pattern used to prevent the duplicate form submissions problem in web applications.
- **Problem Addressed by PRG**: Without PRG, when a user submits a form using a POST request, the server processes the form data and sends a response (typically an HTML page). If the user refreshes the page or navigates backward, the browser may attempt to resubmit the form, causing the form action (e.g., creating a record, making a payment, etc.) to be repeated. This is known as the duplicate form submission problem.
- **The PRG pattern breaks the process into three steps**:
    - POST: The form is submitted using a POST request to the server.
    - Redirect: After processing the POST request, the server redirects the client to another URL (usually using an HTTP 302 redirect).

```
resp.sendRedirect("nextpage");
```

    - GET: The browser follows the redirect and performs a GET request to the new URL, which usually displays a success page or other results.
- This way, if the user refreshes the page after the redirect, the form is not resubmitted, since the browser is now performing a simple GET request instead of redoing the POST operation.

State Management

- HTTP is stateless protocol.
- State management is maintaining information of the client.
- Client side state management
  - Cookie
  - QueryString
  - Hidden form fields
  - SessionStorage and LocalStorage -- Java Script.
- Server side state management
  - Session
  - ServletContext
  - Request

Cookie

- Cookie is a text information in form of key-value pair maintained at the client (browser).
- Server creates a cookie and send to the client in a response.

```
Cookie c = new Cookie("key", "value");
resp.addCookie(c);
```

- Thereafter with each request client send that cookie back to the server.

```
Cookie[] arr = req.getCookies();
for(Cookie c:arr) {
    if(c.getName().equals("key")) {
        String value = c.getValue();
        // ...
    }
}
```

- Temporary cookies
  - Cookies are stored in browser memory. By default, cookies are destroyed when browser is closed.
- Persistent cookies
  - Server can set expiry date for the cookie. Such cookies are stored on client machine (disk) until expiry time.

```
Cookie c = new Cookie("key", "value");
c.setMaxAge(seconds);
resp.addCookie(c);
```

  - Such cookie is accessible even after browser is restarted.
  - Such cookie can be destroyed forcibly by setting max age = -1.

```
Cookie c = new Cookie("key", "");
c.setMaxAge(-1);
resp.addCookie(c);
```

- Limitations/Drawbacks
  - Cookies are stored on client machine. So they are visible to client. Never store sensitive information into cookies.
  - Clients may modify/tamper the cookies (using browser plugins).
  - Cookie max size is 4 KB. Also sending cookie in each request consumes bandwidth.

Session

- https://docs.oracle.com/javaee/7/api/javax/servlet/http/HttpSession.html
- Http Session is used to save data/state of user on server side in form of key-value pair.
- One session is created for each user/client for first call to req.getSession(). The sub-sequent calls returns the same session object (for that user).

```
HttpSession session = req.getSession();
```

- The data can stored in session as attributes -- (String)key-value(Object) pairs.

```
session.setAttribute("key", value);
```

- This data can be retrieved back (from same user session).

```
value = session.getAttribute("key");
```

- The session data can be destroyed while logout.

```
session.invalidate();
```

Assignments

1. Complete the classwork -- bookshop application. Add BookListServlet to display list of all books for "admin" user.