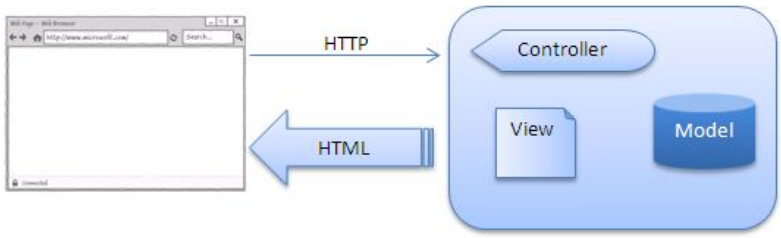
Web Applications 2017

Week 2, Slides 1

Content

- Server side MVC
- Templating
- Java Server Pages
- Mater Detail interface
- Mustache
- Express Framework
- Servlets, Filter and Listeners
- JEE In house MVC
- PRG pattern
- Sessions

Server Side MVC

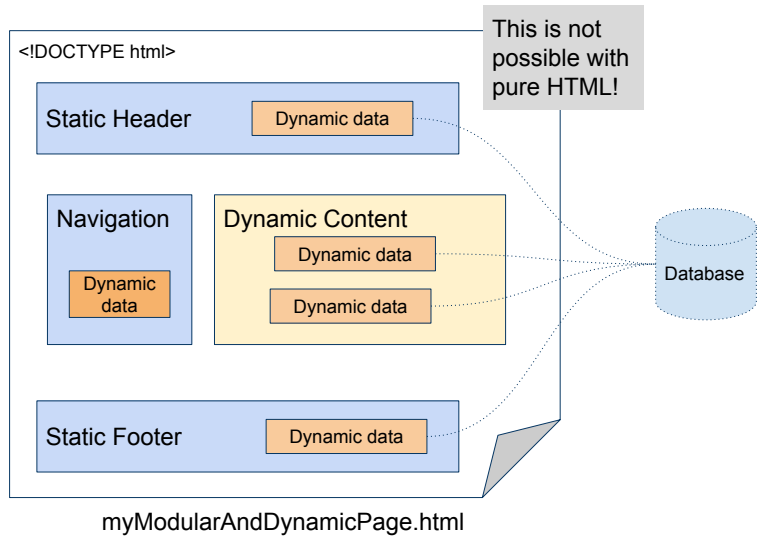


Server

Normally use a framework
on top of platform

The first “architecture” we’ll try.

Templating



Must have a way of composing pages, some parts common: header, footer .
Must have a way of inserting dynamic data.

Templating Solutions

Mustache

```
<!DOCTYPE html>
<html>
    <body>
    <h1>Mustache</h1>
    Details for {{person.id}}
    </body>
</html>
```

ASP.NET

```
<!DOCTYPE html>
<html>
    <body>
        <h1>ASP.NET</h1>
        <p>
            The time is @DateTime.Now
        </p>
    </body>
</html>
```

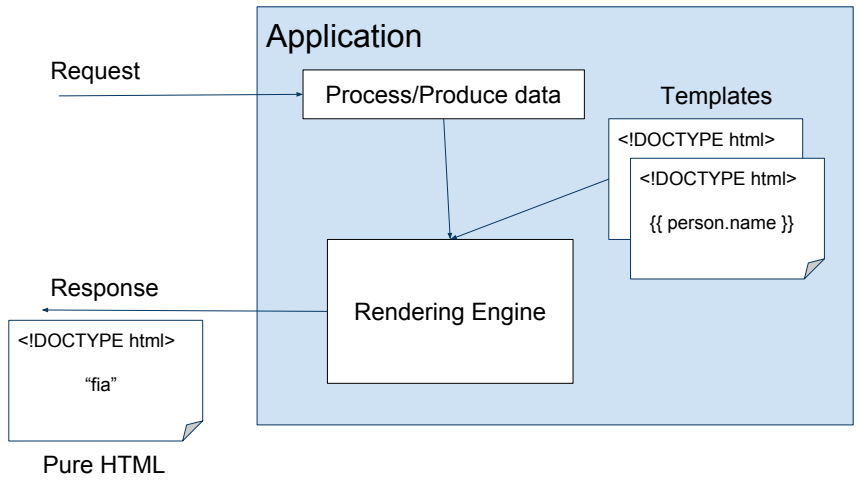
Java Server Pages

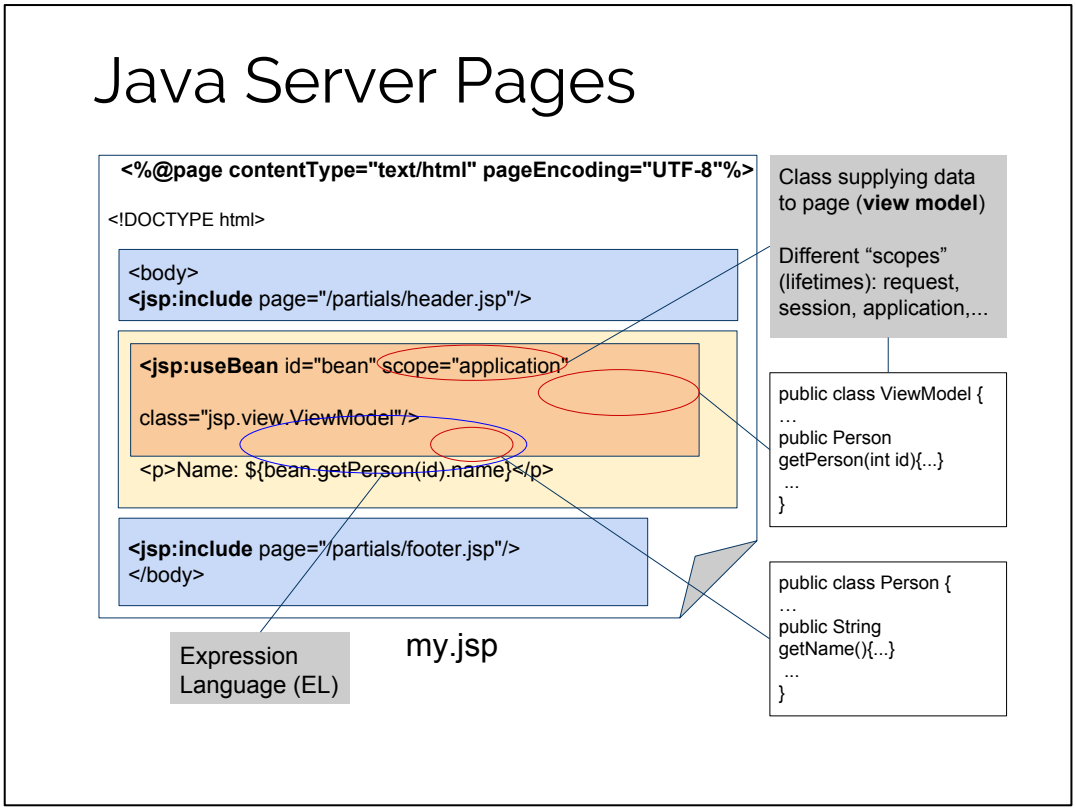
```
<!DOCTYPE html>
<html>
    <body>
        <h1>JEE</h1>
        <p>
            ${contact.name}
        </p>
    </body>
</html>
```

Similar approach for many platforms

- Let page composer write HTML with some added tags and/or expressions
- Server intercept page and replaces expression with values (possibly add static parts)
- Server sends processed page to client (i.e. this is server side rendering)

Server side Templating





JSP is a templating technique for JEE.

Readings

- JSP at Tutorialspoint (NOTE: Will not use "scriptlets" i.e. Java code in pages).

Java Standard Template Library

```
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>

<body>
<jsp:useBean id="bean" scope="application"
                                        class="jsp.view.ViewModel"/>
<table>
   <c:forEach var= "person" items="${bean.persons}" >
       <tr>
         <td>
           <a href="details.jsp?id=${person.id-1}">${person.id}</a>
         </td>
         <td>
           ${person.name}
         </td>
       </tr>
    </c:forEach>
</table>

</body>
```

Person[] getPersons(){...}

Dynamic table

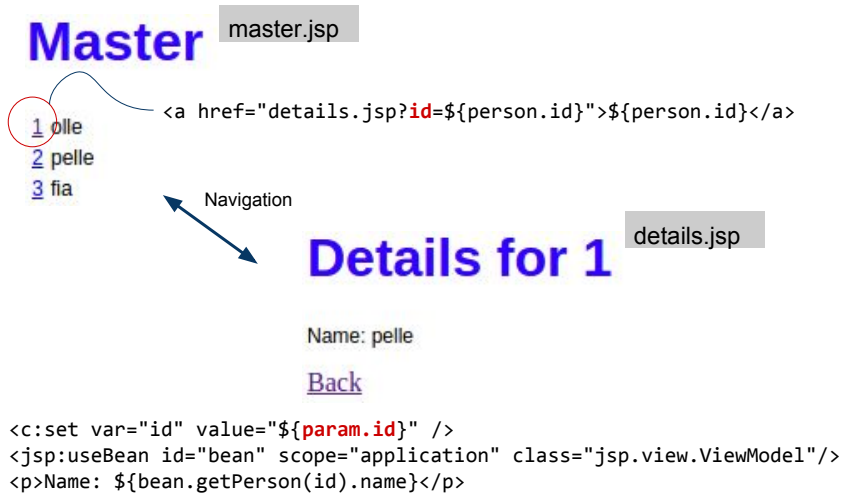
Java Standard Template Library (JSTL)

- Collection of tags for assignment, selection, iteration
- Must add as dependency

Readings

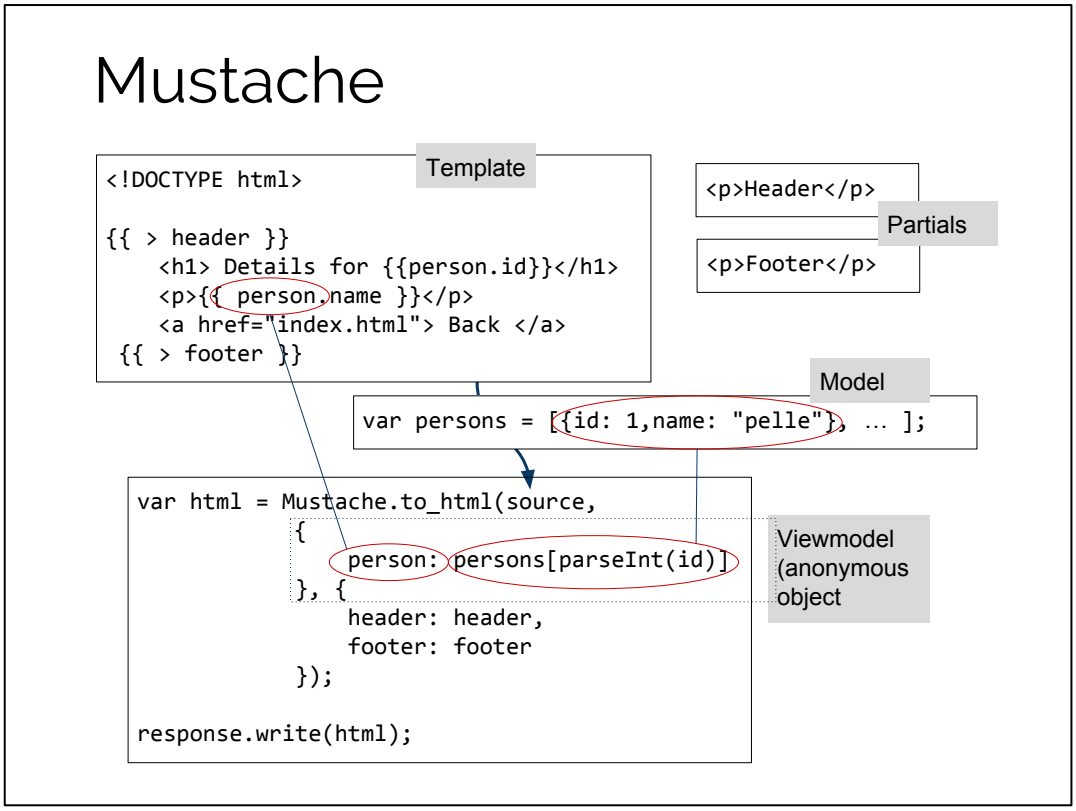
- JSTL at Tutorialspoint

JSP Master Detail Interface



Very common GUI design

- Master consisting of a list.
- When editing we would like to display the details of a single row



Mustache is a template system (works with many languages)

Readings

- [Mustache for JS]

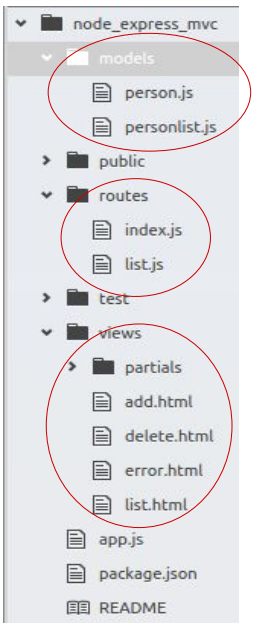
Mustache View Logic

```
<body>
  {{ > header }}
    <h1> Details for {{person.id}}</h1>
    <p>{{ person.name }}</p>
    <a href="index.html"> Back </a>
  {{ > footer }}
</body>
```

```
<body>
    {{ > header}}
    <h1>Master</h1>
    <ul>
        <!-- Be ware of unclosed element (spelling)!! -->
        {{#persons}}
        <li><a href="details.html?id={{id}}">{{id}}</a></li>
        {{/persons}}
    </ul>
    {{ >footer }}
</body>
</html>
```

Mustache is “logicless”, everything is tags (for iteration etc.)

Express Framework



M

C

V (using Mustache)

Express is a NodeJS server side MVC framework

- Rather fixed design, but many possible templating solutions.

Readings

- Express
- Mocha (testing)
- Node-inspector (debug) … hmmm, how to … ?!

JEE MVC

No fixed design. Have to design ourselves!

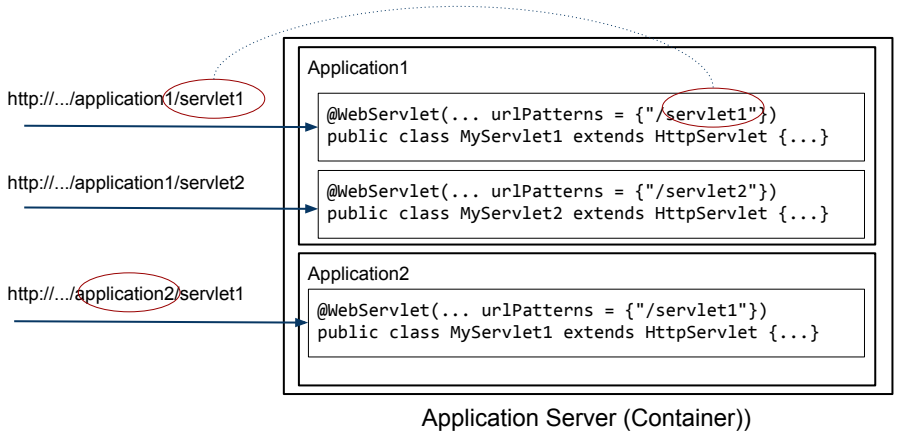
So far

- View: JSPs
- Model: Java classes

Need Control!

- Servlets
- Filters (optional)
- Listeners (optional)

JEE Servlets



Servlet = Java class able to handling incoming HTTP-request. May send responses (avoid, use JSPs)
Connection between web and Java universes

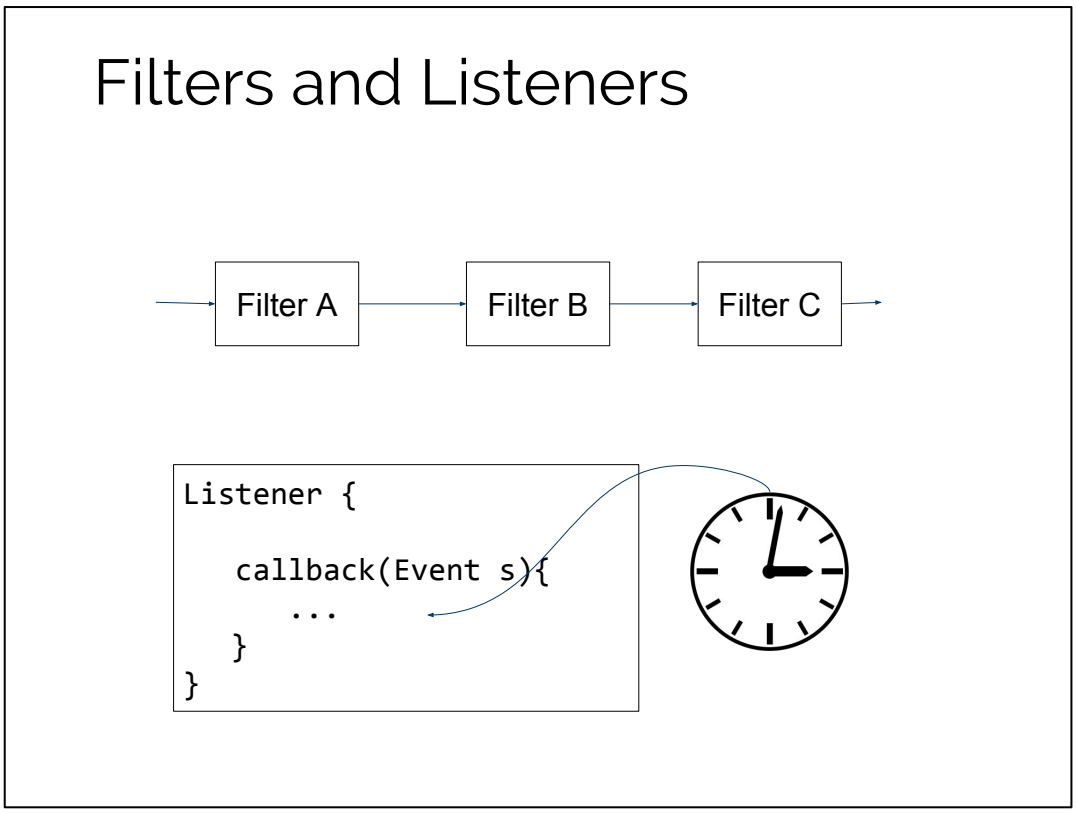
- Accessed using URL

Readings

- Java Servlets tutorial

Servlets API

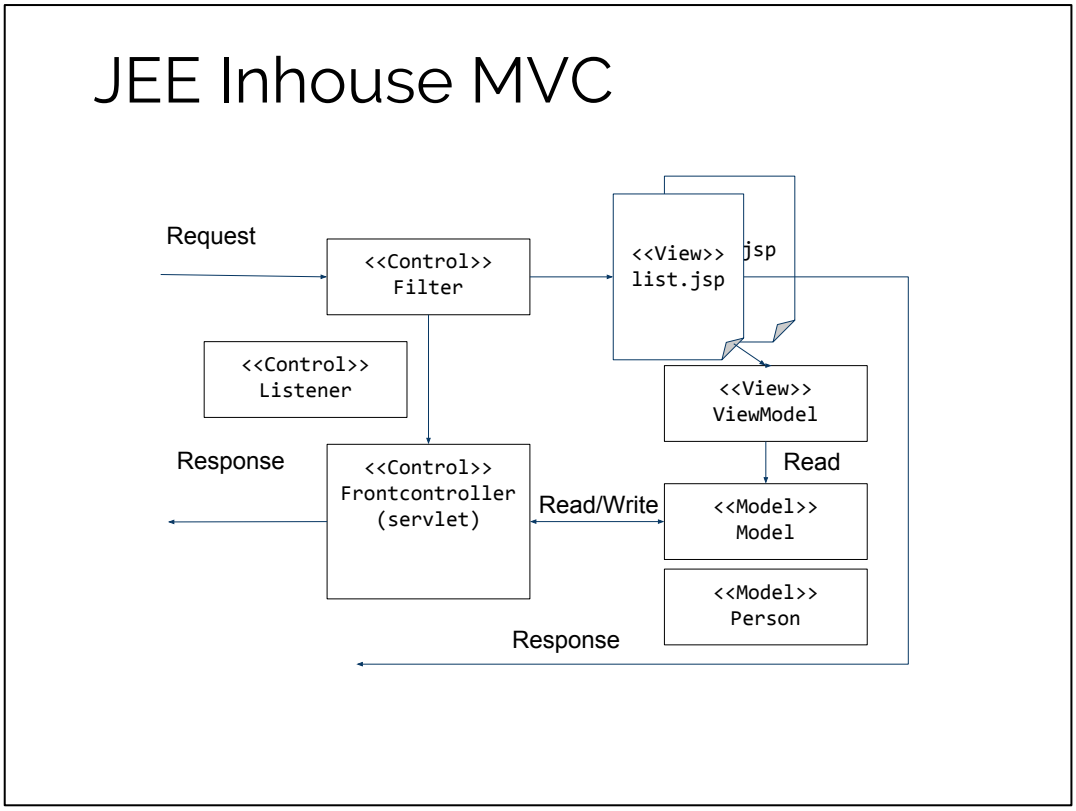
```
// Methods matching the verbs

protected void doGet(HttpServletRequest req, HttpServletResponse resp)
protected void doPost(HttpServletRequest req, HttpServletResponse resp)
protected void doPut(HttpServletRequest req, HttpServletResponse resp)
protected void doDelete(HttpServletRequest req, HttpServletResponse resp)
protected void doOptions(HttpServletRequest req, HttpServletResponse
resp)
protected void doHead(HttpServletRequest req, HttpServletResponse resp)

// Any request
protected void service(HttpServletRequest req, HttpServletResponse resp)
...
```



Filters: Well Known design patterns, processing chain
Listener: Class with callbacks invoked at certain times (life cycle callbacks)

Readings

- Something similar to filter DP

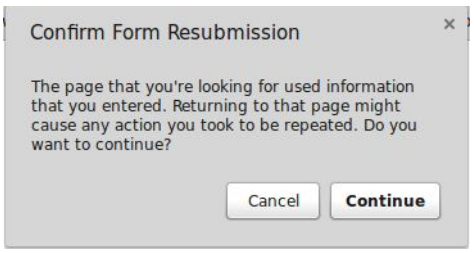


NOTE: There are other designs

Readings

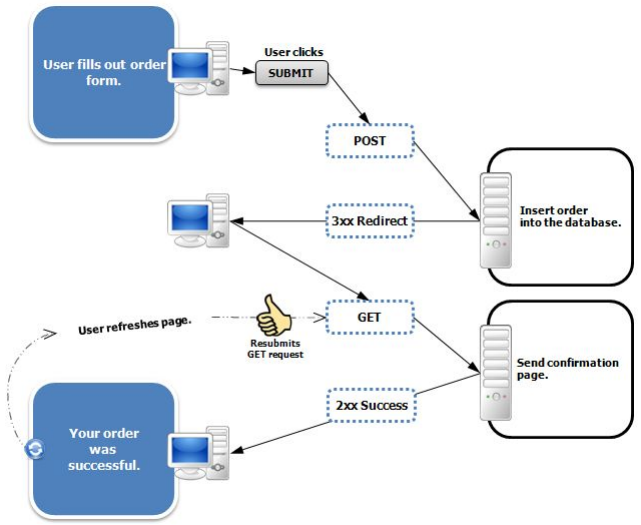
- Frontcontroller pattern

The Double Submit Problem



Shouldn't be possible to submit same POST data twice i.e. buy two pair of shoes or similar (POST not idempotent)

PRG Pattern



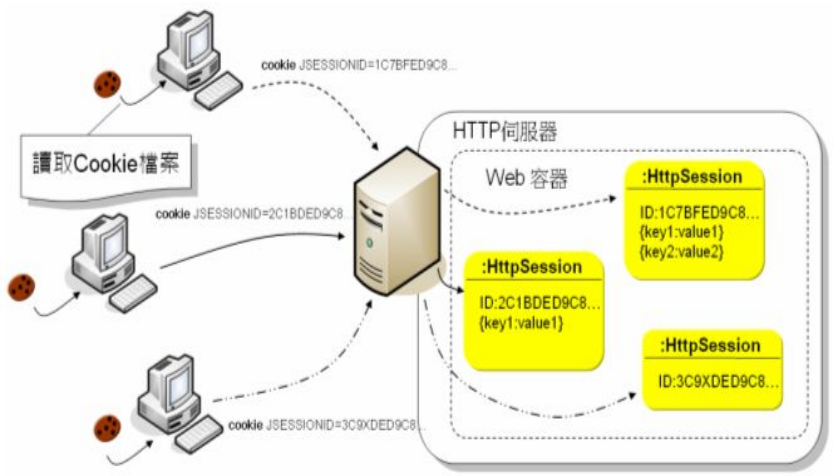
To avoid the double submit problem send a redirect to client (a GET after the POST)

- If reload, another GET…
- … problem solved!

Readings

- POST/Redirect/GET-pattern

Sessions



HTTP stateless, must “invent” some stateful mechanism i.e. sessions

- Normally using cookies

Readings

- Session
- A typical HTTP session
- JEE Sessions
- Express session