A Request Based Approach with JEE

BWA Slides #7

Finally Arriving at Request Based Application

Have platform and all techniques

Need some design
- MVC
- PRG-pattern

Characterization Review

Request based approach interact directly with low level HTTP concepts, explicitly handling request and responses

Pro

- Full control
- Many programmer like

Cons

- Low abstraction level

Approach present on many other platforms (PHP,..)

MVC for Request Based JEE

Often hidden in some framework but we decided not to use any server side framework

- We do an in-house design using the **FrontController (JEE) design pattern**

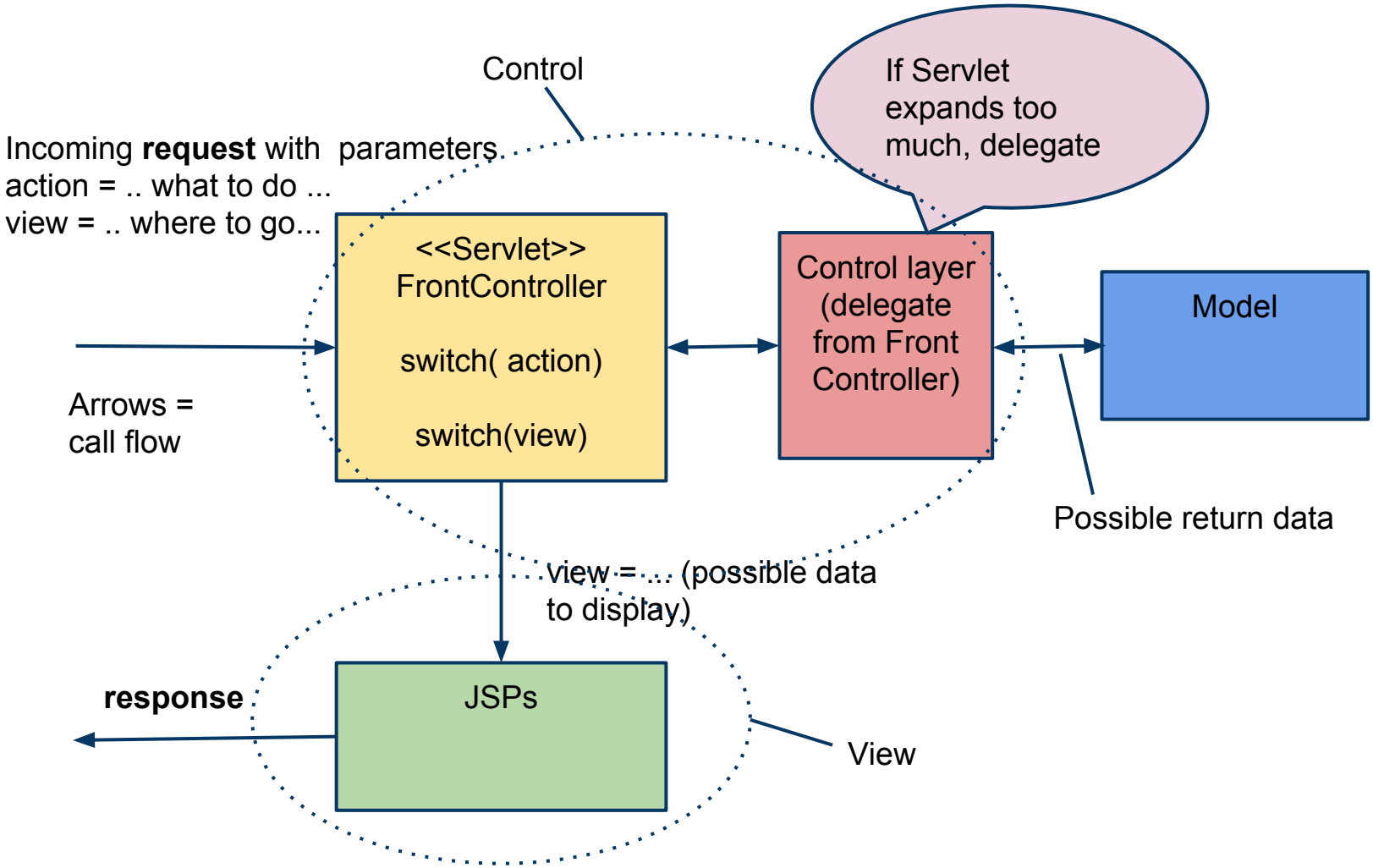
MVC Parts

Model: Model objects (Product, ShoppingCart,...)

View: JSPs

Control: One or more "FrontController" Servlet(s)
- "All" request handled by the Servlet(s)
- Servlet do all processing (or delegate to control layer) using model objects and lastly decides the outcome view
- Asynchronous processing possible

MVC/Front Controller



The "Double Submit" Problem

Shouldn't be possible to submit same post data twice i.e. buy two Gibson or other (post not idempotent)

If page rendered as result of a post (remember!) we possibly get another post if

- reloading result page using Refresh/Reload browser button (explicit page reload, implicit resubmit of request)
- clicking Back and then Forward browser buttons (implicit page reload and implicit resubmit of request)
- returning back to HTML form after submission, and clicking Submit button on the form again (explicit resubmit of request)

Browser shows warning (but possible confuses user)

The PRG pattern

Post-Redirect-Get pattern is a solution to the double submit problem

- Never show pages in response to POST
- Always load pages using GET
- Navigate from POST to GET using redirect
- Page reloaded with GET no problem

More issues and details

JEE request based implementation

- Let FrontController servlet redirect if handling a post

The "Master/Detail" Problem

We have a (master) table with items. Would like to display details about single item

- How to transfer search criteria from master to detail (which item to select)?

Request Based Solution

- Add link to detail in master, use EL to add search criteria

```
<!-- In master (a table) add links to detail -->
<tr>
    <td>${i.id}</td>
    <td>${i.name}</td>
    <td>${i.price}</td>
    <td><a href="products?view=edit&amp;id=${i.id}">Edit</a></td>  <!-- Details link --
    >
    <td><a href="products?view=del&amp;id=${i.id}">Del</a></td>
</tr>
```

User Navigation

Navigation in long lists (spread over many pages)

- Use a "current page" as a session attribute
- Use EL in page to navigate (accessing current page)
- Possible need extra Servlet to do navigation

```
<!-- The previous link -->
<a href="shop?view=products&amp;page=${sessionScope.CURRENT_PAGE-1}"
                                class="btn">Prev</a>
```

Fancy URI's (URLs)

"...Rewritten URIs (sometimes known as short or fancy URIs) are used to provide shorter and more relevant-looking links to web pages. The technique adds a degree of separation between the files used to generate a web page and the URI that is presented to the world [hide implementation details]."

//Wikipedia

Examples

- Non fancy: http://www.example.com/Blogs /Posts.php?Year=2006&Month=12&Day=19
- Fancy: http://www.example.com/Blogs /2006/12/19/

Designing URI's

Hard to say where it's going to end. i.e what URI will we finally show the user

Possible adhere to using "technical" URI inside application
- Easy for developer to understand

Map technical URIs to user friendly (fancy) URIs
- No direct support for mapping in JEE
- PrettyFaces
- Possible use server (application external). Not covered

Authentication and Authorization

Simple in-house solution for now

- When user logs in, if login ok, put User-object in session
    - Hard code some users, later we use a database
- Use a Filter for protected resources
- Filter check if there is a user in session, if not redirect to login
- Logout or timeout removes destroys session (and user)

We'll later look at the official (and other) way to do it