MVC and EL

Moving away from scriplets

EHSDI
eBuzima

Scriptless JSPs

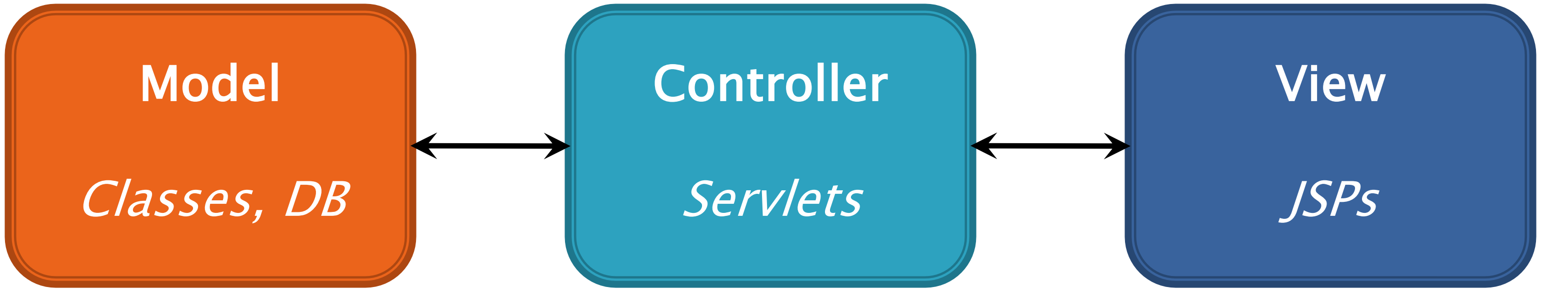
- Creating HTML web pages entirely within servlet classes is difficult
- JSPs make life easier because Java code can put inside HTML
- But this still mixes the business logic and the presentation!

Why scriptless JSPs?

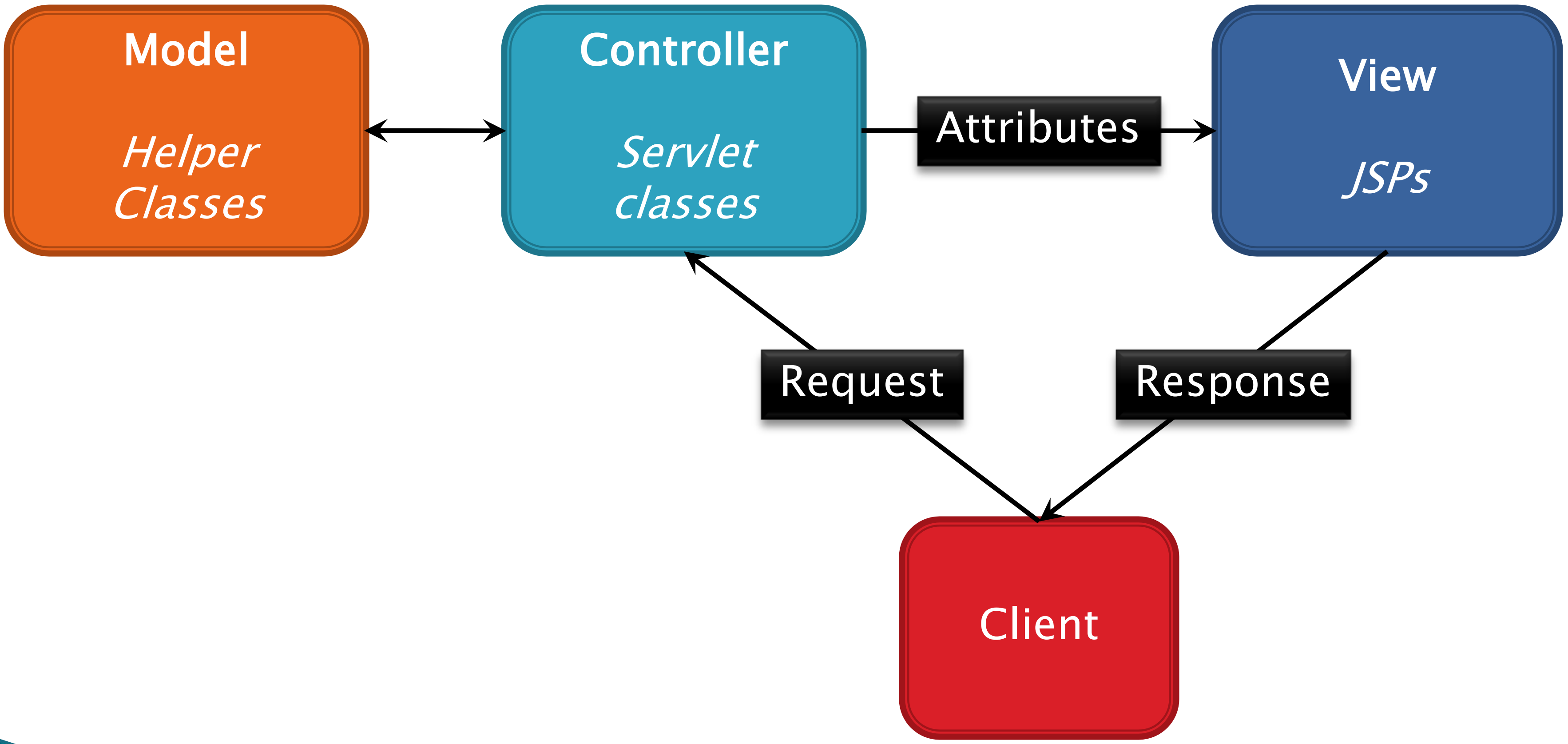
- Still requires the person designing the web page to understand Java
- Makes it very difficult to use a WYSIWYG editor
- Makes it harder to update the design of the site

Model-view-controller

- Is an architecture which separates
  - the data (model)
  - the business logic (controller)
  - the presentation (view)

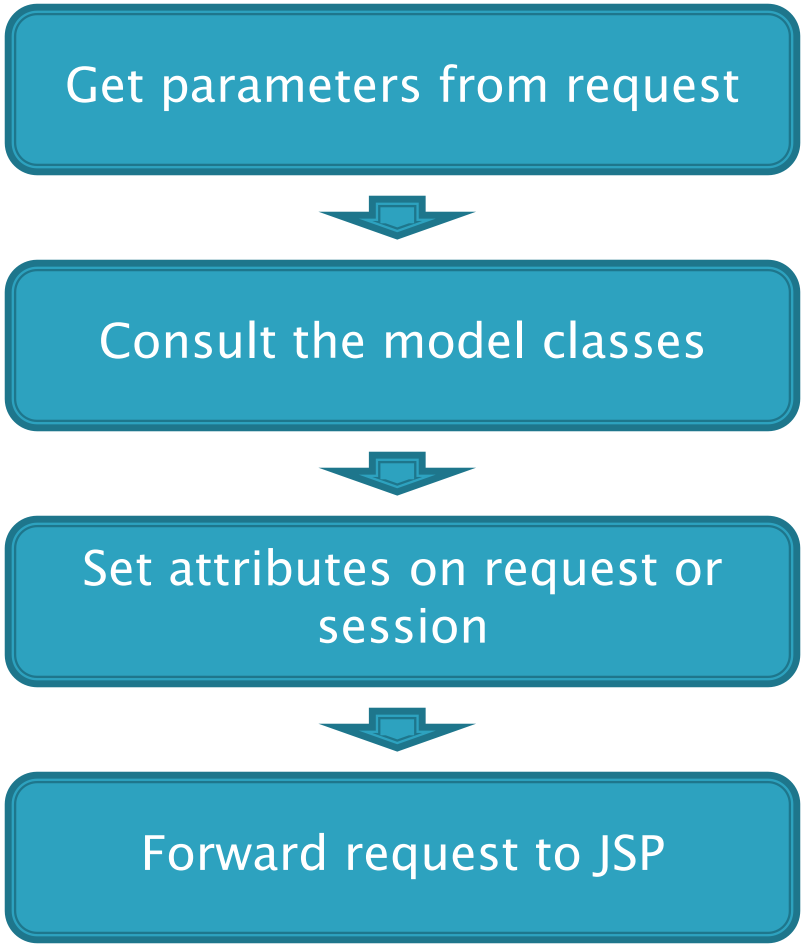


In more detail…



The Controller

- The controller receives the request from the client
- Parameters are read from the request
- Model classes used for any business logic
- Controller doesn't return any HTML
- JSP creates the response


Get parameters from request
Consult the model classes
Set attributes on request or session
Forward request to JSP

The Controller

```
public void doGet(HttpServletRequest request, HttpServletResponse) {
  String name = request.getParameter("name");
  String password = request.getParameter("password");        1

  User user = Security.login(name, password);        2

  request.setAttribute("curUser", user);        3

  RequestDispatcher view = request.getRequestDispatcher("/login.jsp");
  view.forward(request, response);        4
}
```

Gets parameters from the view 1

Consults the model for any business logic 2

Sets attributes to be used by view 3

Forwards the request to the view 4

EHSDI
eBuzima

The View

- This JSP can now return an HTML response using the attributes set by the controller

```
<p>Result is:
  <%= ((User)request.getAttribute("user")).getName() %>
</p>
```

- However, this still requires Java code, which might be quite complex if we need to generate something like a table of results
- One alternative to this is...

Actions

- These are server side tags which we can use instead of scriptlets, e.g.

```
<jsp:include page="header.jsp" />
```

```
<c:set var="name" value="Guest" />
```

- Tags in the `jsp` namespace are called *standard actions*
- Other tags are called *custom actions*

Displaying attribute properties

- This requires the `useBean` and `getProperty` actions, e.g.

```
<p>Result is:
  <%= ((User)request.getAttribute("curUser")).getName() %>
</p>
```

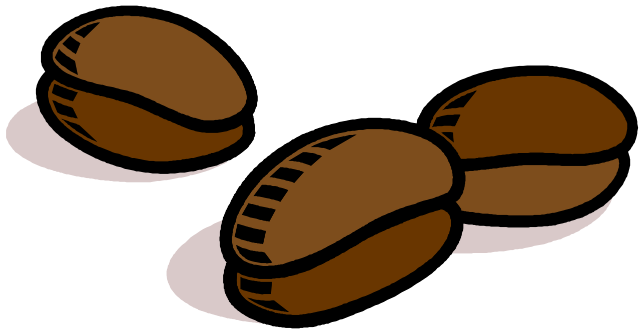
```
<p>Result is:
  <jsp:useBean id="curUser" class="User" scope="request" />
  <jsp:getProperty name="curUser" property="name" />
</p>
```

EHSDI
eBuzima

Java Beans

- The `jsp:useBean` action requires that the object be a "Java Bean", i.e.
  - It must have a *public default (no arg) constructor*
  - It's properties must be exposed using *getters and setters*

- Note: Enterprise Java Beans (EJB) are a little different and those must also be *serializable*



EHSDI
eBuzima

Bean properties

These must use the proper syntax !

creates a writable property called **name**

creates a readonly property called **age**

booleans use `is` instead of `get`

```
public class User {
   ...

   public String getName() {
      ...
   }
   public void setName(String name) {
      ...
   }

   public int getAge() {
      ...
   }

   public boolean isAdmin() {
      ...
   }
}
```

Introducing Expression Language

- Standard actions are better than scripting, but *Expression Language (EL)* can really simplify our JSPs

```
<p>Result is:
  <%= ((User)request.getAttribute("curUser")).getName() %>
</p>
```

↓

```
<p>Result is:
  ${curUser.name}
</p>
```

EHSDI
eBuzima

EL Syntax

- EL expressions are always given in curly braces, preceded by a dollar sign, i.e.

```
${curUser.name}
```

- The first thing in an expression must be one of two things:
    - An attribute (in page, request, session or application scope)
    - An implicit object (not the same implicit objects available to scriptlets!)

EHSDI
eBuzima

EL is not Java!

- Its always a single expression – not statements
- It has two methods for accessing object properties
  - Dot (.) operator
  - [ ] operator

`${curUser.name}` OR `${curUser["name"]}`

EHSDI
eBuzima

Dot operator

- Looks neater

```
${curUser.name}
```

$$\underbrace{\texttt{curUser}}_{\text{Must be a \textbf{Map} or a \textbf{Bean}}}.\underbrace{\texttt{name}}_{\text{\textbf{Map key} OR \textbf{Bean property}}}$$

Map key OR **Bean property**

Map key $\xrightarrow{\text{e.g.}}$ curUser.get("name")

Bean property $\xrightarrow{\text{e.g.}}$ curUser.getName()

[ ] operator

- More flexible

```
${curUser["name"]}
```

Can be a...
- Map
- Bean
- Array
- List

Can be a...
- Map key
- Bean property
- Array index
- List index

[ ] operator

- Will also evaluate attributes as keys, e.g. if we have something like...

```
request.setAttribute("curProp", "name");
```

in the servlet, then we can use it in EL...

```
${curUser[curProp]}
```

- And it allows nested expressions, e.g.

```
${curUser[userProps[0]]}
```

EHSDI
eBuzima

EL implicit objects

- Remember these are different to JSP's implicit objects!
- By default, EL searches in all scopes for an attribute name, but we can specify a scope with the following implicit objects

`pageScope`	`requestScope`
`sessionScope`	`applicationScope`

EHSDI
eBuzima

EL implicit objects: parameters

- To get a request parameter, use `param`, e.g.

```
<input type="text" name="username" />
```

```
${param.username}
```

- If request parameter can have multiple values, use `paramValues`, e.g.

```
<input type="checkbox" name="roles" value="admin" />
<input type="checkbox" name="roles" value="visitor" />
```

```
${paramValues.roles[0]}
${paramValues.roles[1]}
```

EHSDI
eBuzima

EL implicit objects: cookies

- EL makes working with cookies easy compared to scriplets, e.g.
  - To get a cookie by name in a scriptlet

```
<% Cookie[] cookies = request.getCookies();
for (Cookie c : cookies)
  if (c.getName().equals("username"))
    out.println(c.getValue());
%>
```

  - But in EL....

```
${cookie.username.value}
```

EHSDI
eBuzima

Summary of EL implicit objects

Name(s)	Desciption

`pageScope`
`requestScope`
`sessionScope`
`applicationScope`
	Maps of attributes in each scope

`param` `paramValues`	Maps of request parameters (i.e. GET and POST parameters)
`header` `headerValues`	Maps of request headers
`cookie`	Map of cookies
`pageContext`	The actual `pageContext` object – for accessing anything!

EL operators

- Remember: EL is not for your business logic – that should be in the controller or model
- But it does have some operators to provide basic functionality, like…

Arithmetic		
Addition	+	
Subtraction	–	
Multiplication	*	
Division	/	div
Remainder	%	mod

EL has alternatives for some operators

More EL operators…

Logical		
And	&&	and
Or	||	or
Negation	!	not

Comparative		
Equality	==	eq
Not equals	!=	ne
Less than	<	lt
Greater than	>	gt
Less than or equals	<=	le
Greater than or equals	>=	ge

EL is null friendly

- Web designers don't want to worry about things like `NullPointerExceptions`
- Even if an attribute doesn't exist
  - EL displays nothing instead of an exception
  - If its used in an arithmetic expression it treats it as zero
- If you divide by zero using EL you get infinity instead of an exception

References

- Books
  - Head First Servlets and JSP (O'Reilly)
- Websites
  - http://java.sun.com/javaee/reference/tutorials/