MONASH University

MONASH
INFORMATION
TECHNOLOGY

FIT5192 Lecture 6: Developing Web Interfaces with Java Server Faces

GROUP OF EIGHT AUSTRALIA

MONASH University

Last Lecture

- Get a quick introduction into the Java Server Faces Framework (JSF) and how we can build web interfaces with it.
- Understand the purpose of Managed Beans and how we can work with data in different scopes.
- Understand how Expression Language can be used to retrieve values from managed beans.
- Review the processes involved in the JSF framework that make up the lifecycle of a deployed web application

GROUP OF EIGHT AUSTRALIA

MONASH University

This Lecture

- Examine how we can apply an MVC pattern approach to developing JSF web applications
- Review the approaches that we can take for validating data with the Java EE 7 platform
- Look at some examples including Ajax

GROUP OF EIGHT AUSTRALIA

MONASH
University

MVC

MVC in Java EE

- **Let's look at the overall architecture with Java EE:**
- ***Models* work with the business domain aspect of the application**
  – Managed Beans
  – Enterprise Beans
- ***Views* serves as the interface backend:**
  – Represented by JSF
- ***Controller* is handled by FacesServlet:**
  – Rules which dictate how HTTP requests are handled.
  – Additional rules and filters applied for configurations such as security

MVC in JSF

- **But we can break it down a bit more when working with JSF**
- **We can write Managed Beans which act as "controllers" for individual pages or major functionality concerned with one or more pages**
- **Examples:**
  - ApplicationBean
  - IndexController, ContactController
  - Navigation Controller
- We will be practicing this in the Labs

MONASH University

Data Validation

Why do we validate content?

- **Ensure that data stored within the application is of the correct format AND is accurate**
- **Vital that Enterprise data is always validated to prevent major issues from appearing**
  - Example: Banks cannot just validate that transaction values are just numbers, they need to verify that customers have a valid balance.
- **When data is accepted but considered invalid, inaccurate or incomplete, major issues can occur in applications**
  - Unhandled exceptions being raised.
  - Inconsistent application performance.

Client and Server Validation

- **Ideally, you should provide the client (web browser) a method in which it can validate content before it is sent to the server**
  - Helps the user correct quick mistakes or missing fields
  - Prevents many requests going to the server
  - Using AJAX allows server validation messages to be shown to the user dynamically
- **Most importantly, you should always enforce server validation as you cannot trust that clients will always validate inputs**
  - Providing validation for both gives us the best of both approaches

Validation Options in JSF

- We can take a few different approaches in validating data received in JSF
- Built-in Validation Components (via JSF Core Library)
  - Example: <f:validateDoubleRange> and <f:validateLength>
  - See tutorial tasks
- Managed Bean Validation Methods / Validator Interface
- Bean Validation (Java EE 6+)

JSF: Validator Classes

Validator Class	Tag	Function
BeanValidator	validateBean	Registers a bean validator for the component.
DoubleRangeValidator	validateDoubleRange	Checks whether the local value of a component is within a certain range. The value must be floating point or convertible to floating-point.
LengthValidator .	validateLength	Checks whether the length of a component's local value is within a certain range. The value must be a java.lang.String
LongRangeValidator	validateLongRange	Checks whether the local value of a component is within a certain range. The value must be any numeric type or String that can be converted to a long.
RegexValidator	validateRegEx	Checks whether the local value of a component is a match against a regular expression from the java.util.regex package.
RequiredValidator	validateRequired	Ensures that the local value is not empty on an EditableValueHolder component.

Bean Validation

- **New method of validation in the Java EE 6+ platform which uses annotations within a Bean class instead!**
- **Examples:**

```
@NotNull
private String lastname;
@Max(15)
int quantity;
```

- **We will explore this more with the REST services (in web services)**
- **Review**:
  https://docs.oracle.com/javaee/7/tutorial/jsf-page-core004.htm

Regular Expressions (Regex)

- **Series of characters that help form a search pattern**
  - Used for pattern matching with Strings
  - Very flexible for working with huge variations of patterns present within a provided String
- **Very common implementation supported by many modern programming languages**
  - Java uses it in many spots for validation! Methods such as String.matches(…)
  - Many search engines also provide support in their queries for using Regex
- **Unfortunately, the syntax is very complex and hard to read due to its robustness**

Regex Example

```
<h:inputSecret id="password"
value="#{user.password}"> <f:validateRegex
pattern="((?=.*\d)(?=.*[a-z])(?=.*[A-Z])
(?=.*[@#$%!]).{6,20})" />
</h:inputSecret>
```

- required 6 to 20 characters string with at least one digit, one upper case letter, one lower case letter and one special symbol (“@#$%!”)

RegEx Characters

Construct	Description
[abc]	a, b, or c (simple class)
[^abc]	Any character except a, b, or c (negation)
[a-zA-Z]	a through z, or A through Z, inclusive (range)
[a-d[m-p]]	a through d, or m through p: [a-dm-p] (union)
[a-z&&[def]]	d, e, or f (intersection)
[a-z&&[^bc]]	a through z, except for b and c: [ad-z] (subtraction)
[a-z&&[^m-p]]	a through z, and not m through p: [a-lq-z] (subtraction)

https://docs.oracle.com/javase/tutorial/essential/regex/char_classes.html

RegEx Pre-defined Characters

Construct	Description
.	Any character (may or may not match line terminators)
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\s	A whitespace character: [\t\n\x0B\f\r]
\S	A non-whitespace character: [^\s]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]

https://docs.oracle.com/javase/tutorial/essential/regex/pre_char_classes.html

RegEx Quantifiers

Greedy	Meaning
X?	X, once or not at all
X*	X, zero or more times
X+	X, one or more times
X{n}	X, exactly n times
X{n,}	X, at least n times
X{n,m}	X, at least n but not more than m times

https://docs.oracle.com/javase/tutorial/essential/regex/quant.html

Regular Expressions (Regex) cont.

- **Many examples available**
- **Supplementary material available on the Java Tutorial**
  https://docs.oracle.com/javase/tutorial/essential/regex/
- Syntax is quite extensive and it can often be best learned through examples
  http://www.mkyong.com/tutorials/java-regular-expression-tutorials/
- **Some helpful online resources:**
  – Reference and examples:
    http://www.regular-expressions.info
  – Online RegEx Tester: http://gskinner.com/RegExr/

Custom Validator examples

- More examples available online:

e.g.

http://www.tutorialspoint.com/jsf/jsf_customvalidator_tag.htm

Showing Error Messages

- **We have two tags we can work with for displaying errors: <h:message> and <h:messages>**
  - Message is designated for a single component using the *for* attribute using an id as its value
  - Messages can be used to store all messages not handled
  - The *globalOnly* attribute may need to be set to true if you want to override settings
  - Otherwise messages shows all messages, including those already shown in message

Working with Ajax

What is AJAX?

- **Asynchronous JavaScript and XML**
- – Although we aren't limited to just XML!
- **Enables web interfaces to be updated asynchronously**
  - – Allows for web pages to update content without requiring another page request
  - – Often the technology underpinning "Rich" web interfaces
- **Examples of websites using AJAX:**
  - – Google Gmail
  - – Google Maps
  - – Facebook
  - – Youtube

What is AJAX? cont..

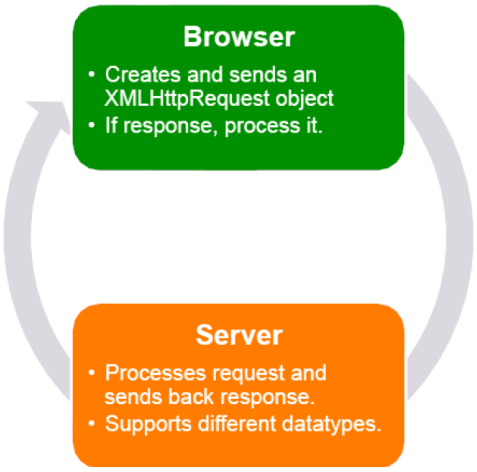
- "Ajax isn't a technology. It's really several technologies, each flourishing in its own right, coming together in powerful new ways.
- Ajax incorporates:
  – standards-based presentation using XHTML and CSS
  – dynamic display and interaction using the Document Object Model
  – data interchange and manipulation using XML and XSLT
  – asynchronous data retrieval using XMLHttpRequest
  – and JavaScript binding everything together"
    - **Jesse James Garrett**

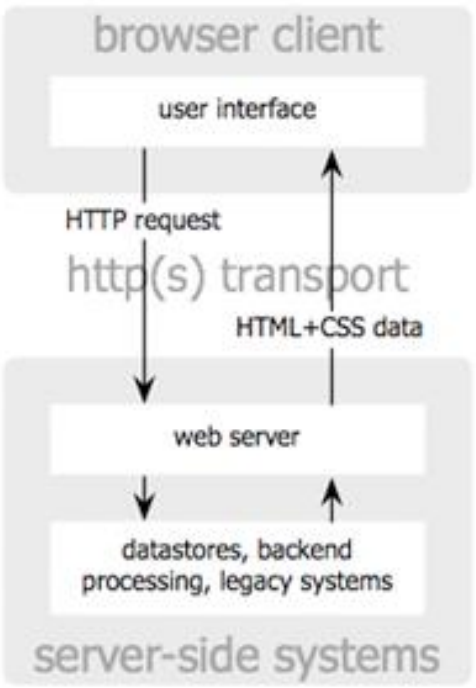
Source: http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

How does AJAX work?

- **AJAX works through an engine via the XmlHttpRequest JavaScript object**
  – First introduced in Internet Explorer 5 !
- **Steps involved:**
  1. XMLHttpRequest (XHR) object is created
  2. XHR attempts to get data from a specified address and awaits result
  3. Server receives the request and sends data back to the client
  4. Client processes the data and if required, notifies the user in some visual manner
     – Example: Displaying search results for a query

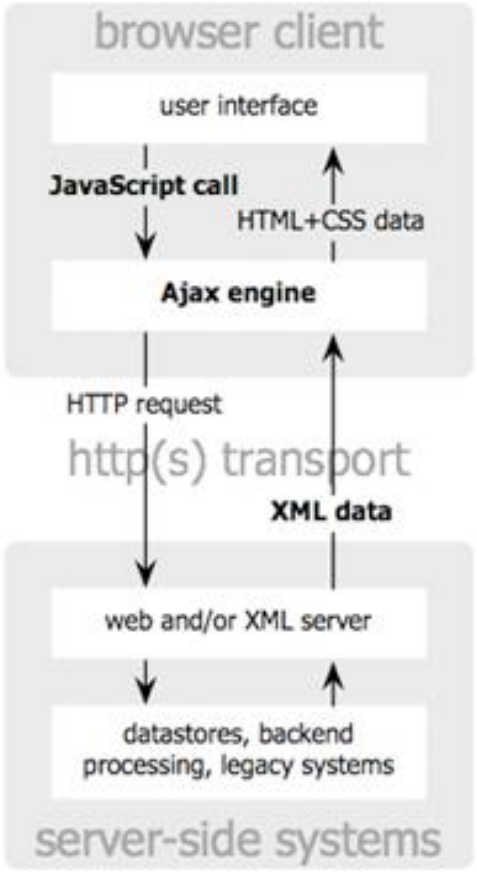
How does AJAX work? Cont..





classic
web application model

Jesse James Garrett / adaptivepath.com



Ajax
web application model

Source: http://adaptivepath.org/ideas/ajax-new-approach-web-applications/

AJAX Use Cases

- **Forms**
  – Ability to edit fields without reloading page
  – Quick validation feedback: “That username already exists”
- **Loading additional resources**
  – Example: Hit the bottom of the page, load additional content
- **Search suggestions**
  – Highlight common or recommended options for a query
- **Shopping carts**
  – Add items without reloading the entire page
  – Quick visual feedback highlighting item was added

Cross-Origin Resource Sharing (CORS)

- Browser implementation that dictates how AJAX requests are handled to external addresses
- Typically requests to domains outside of the server environment are forbidden by web browsers
- **Same-origin Policy**
  – example.com --> example.com/ajax.html – GOOD
  – example.com --> example.net/ajax.html – BAD
- **We can allow external requests by having the server say (via HTTP Response headers) that these requests are OK**
  – Access-Control-Allow-Origin header
  – Eg: Access-Control-Allow-Origin: ”http://example.com"

AJAX in JavaServer Faces

- AJAX support was introduced with JSF 2.0
  We can add AJAX operations using XML elements like other facelet tags via <f:ajax/>
  – xmlns:f="http://java.sun.com/jsf/core"

- **The framework worries about the JavaScript required to process the requests**
- We don't need to touch JavaScript at all if we are doing simple requests!
- **More flexibility in working with application values than with jQuery library**

JSF AJAX: Sending Requests (1)

Attributes defined in the <f:ajax> element dictate how the request will be sent

- **Event**
  - Request is sent once an event condition occurs
  - Example: click, keyup, mouseover…
- **Execute**
  - Specific component(s) which should be executed with the AJAX request
  - Example: Clicking button sends *inputAge* inputText.
- **Immediate**
  - Request is sent early in the lifecycle

JSF AJAX: Sending Requests (2)

- **Listener**
  - Method / expression which is executed in response to an AJAX request made by the client
- Render
  - One or more components that need to be updated after Ajax processing

JSF AJAX: Sending Requests Example

```
<h:form>
    <h:inputText id="name"
value="#{helloBean.name}”/>
    <h:commandButton value=”Say Hello!">
        <f:ajax execute="name" render="output" />
    </h:commandButton>
    <h2>
        <h:outputText id="output”
                      value="#{helloBean.sayWelcome}"/>
    </h2>
</h:form>
```

AJAX in jQuery

- Few different ways that we can make AJAX requests
- **jQuery library:**
  $.ajax(url[,settings])
  $.load(url,[,data][,complete(responseText, textStatus, XMLHttpRequest)])
  – Few other methods exist but the above are the most commonly used.
- Helpful compared to writing AJAX calls manually using XMLHttpRequest as provides callbacks for success / fail cases
- **API provides good examples on how to use AJAX with the methods above**

Example: jQuery AJAX Example

- Load a local page into the element with id ajaxExample

```
$('#ajaxExample').load('example/test.html');
```

- Load page element #articleContent into the page:

```
$('#ajaxExample').load('example/test.html
#articleContent');
```

- Run a function after AJAX content loaded:

```
$('#ajaxExample').load('example/test.html', function()
{ alert('AJAX request has been processed.'); });
```

- Load JSON into a JavaScript object:

```
Var jsonObj = $.load('data/books.json');
```

Other ways of using Ajax in JSF

- RichFaces
- Omnifaces
- iceFaces
- PrimeFaces
- All frameworks for “easing” development

Summary

- Examine how we can apply an MVC pattern approach to developing JSF web applications
- Review the approaches that we can take for validating data with the Java EE 7 platform
- Look at some examples including Ajax

Next Lecture

- **Lecture 7** Advanced Application of Java Persistence
- **After** - JEE enterprise beans

MONASH University

See you in the Studio !

Readings

MONASH University

- **Part III The Web Tier in The Java EE 7 Tutorial**

  Chapter 6 to 13, covers most of the topics we have looked at