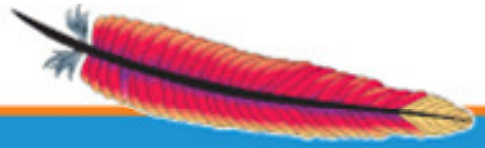
ApacheCon

Ajax in Apache MyFaces

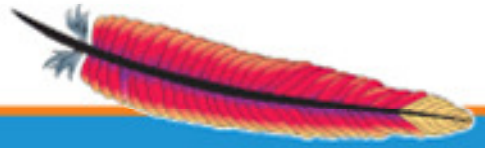
A new Approach to Web Applications



EU 2007

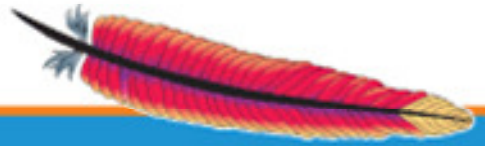
Agenda

- Introduction AJAX and Web 2.0
- Integrating AJAX in JavaServer Faces
- AJAX components in MyFaces
- Discussion (or Question & Answer)



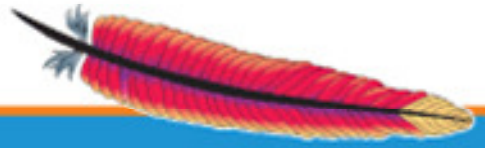
Agenda

- **Introduction AJAX and Web 2.0**
- Integrating AJAX in JavaServer Faces
- AJAX components in MyFaces
- Discussion (or Question & Answer)



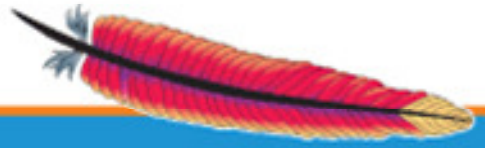
The New Web - Web 2.0

- Highly dynamic and interactive Websites
- Openness (Interaction with other Systems)
- Break the limitations of traditional Websites



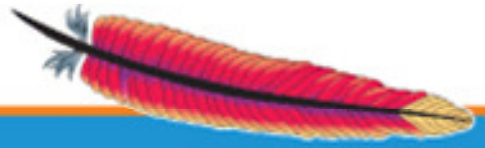
Web 2.0 - History

Web 1.0	Static HTML Pages
Web 1.5	Dynamically generated HTML
Web 2.0	WebSites which dynamically interact with the Server



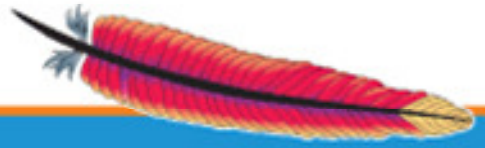
What is AJAX?

- a terminology affected by “Jesse James Garrett” from Adaptive Path in february 2005
- short name for “**A**synchronous **J**avaScript **A**nd **X**ML”
- became a hype in 2005
- popularity raised with the help of Google
  - Gmail, Google Maps, Google Calendar



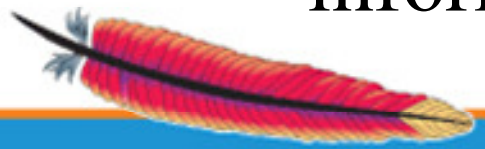
AJAX – a combination of technologies

- HTML (or XHTML) and CSS
- JavaScript / DOM
- XML / JSON (JavaScript objectoriented Notation)
- JavaScript, XMLHttpRequest Object

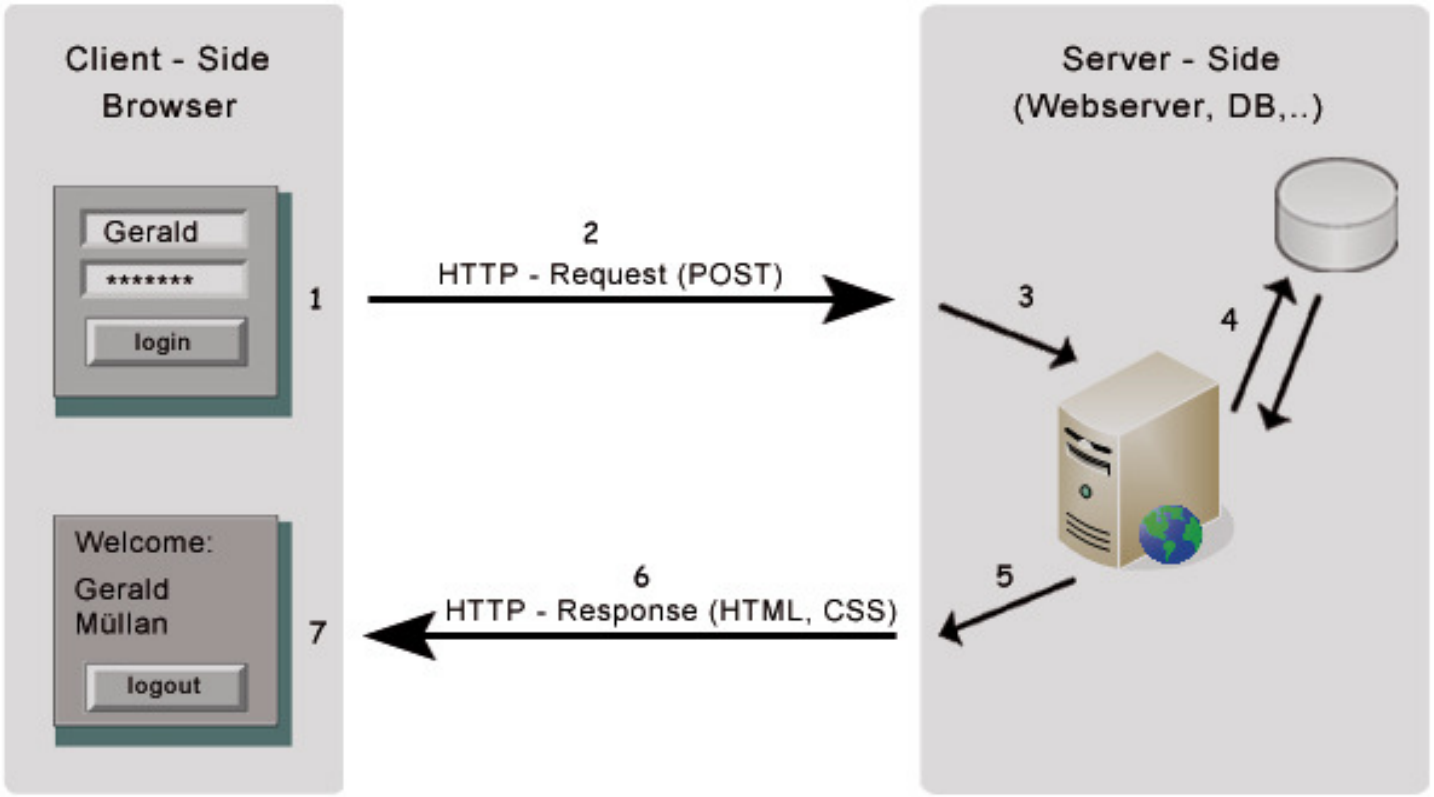


AJAX Interaction

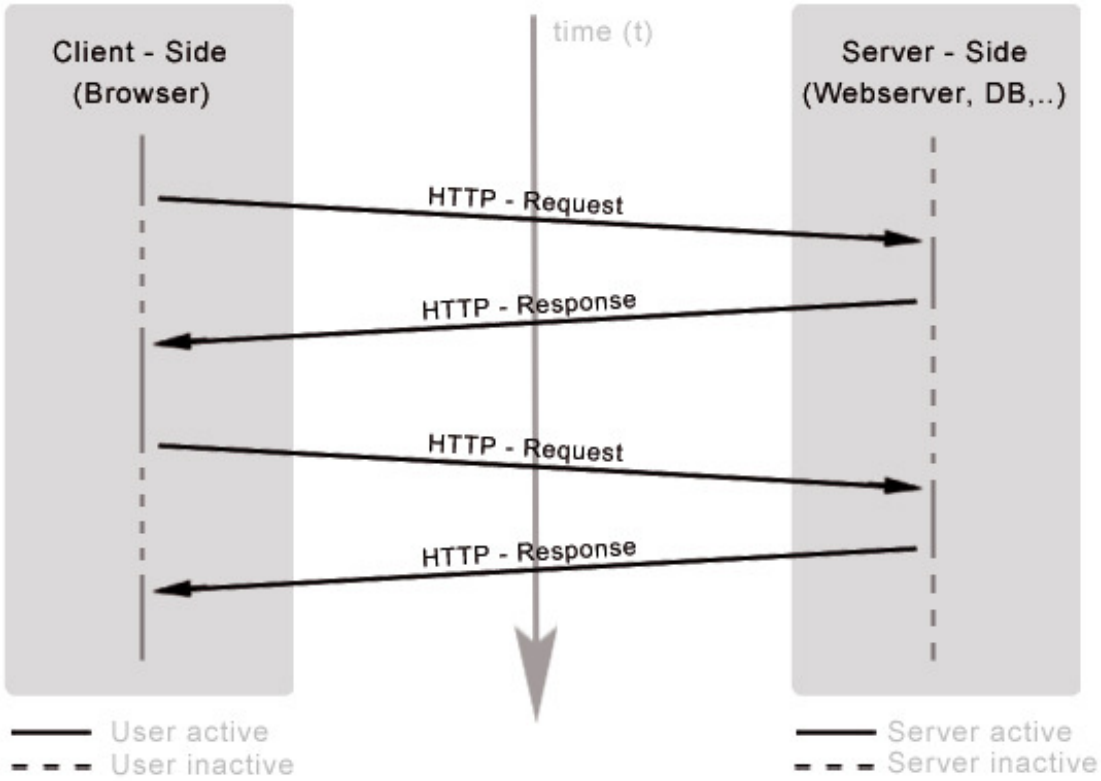
- an AJAX application looks as if it resided on the user's machine
- JavaScript call to AJAX engine
  - HTTP - Request back to server
- browser is updated with gathered information



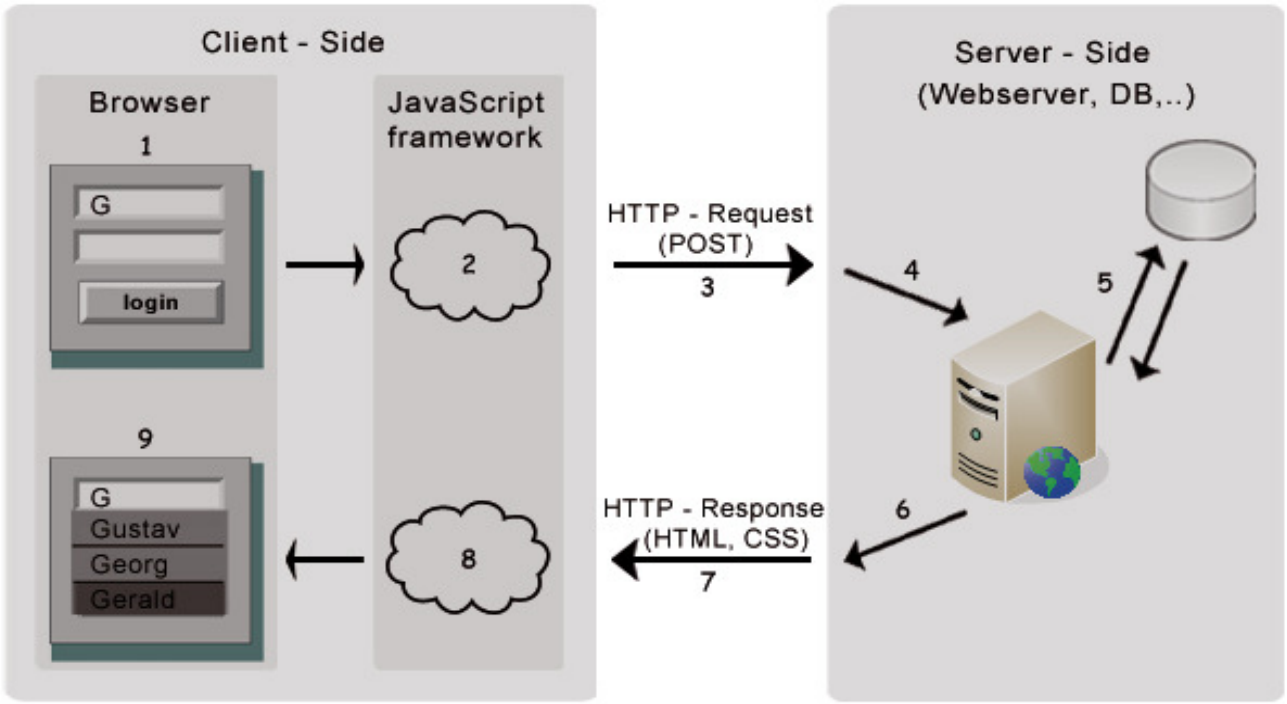
HTTP Request - Response



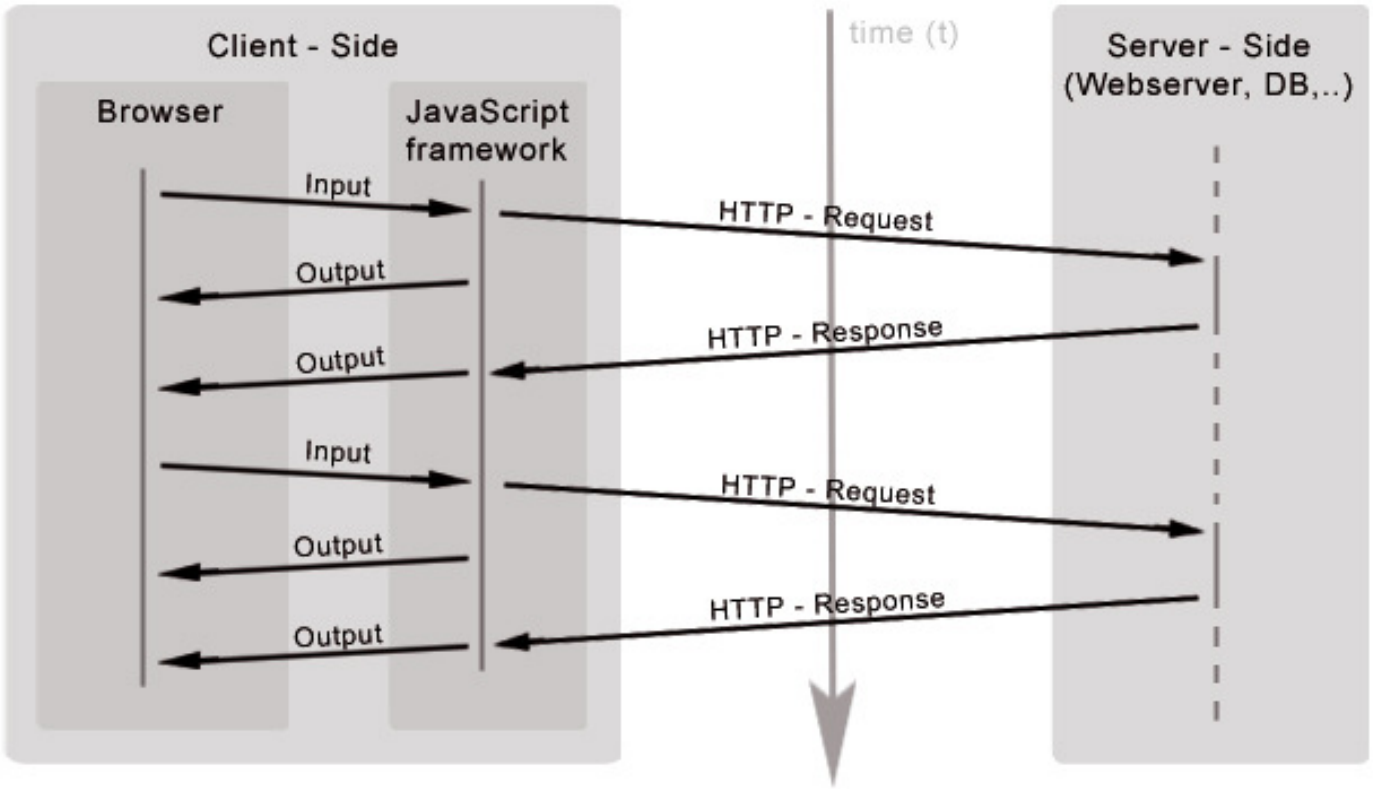
HTTP Process Flow



AJAX Request - Response



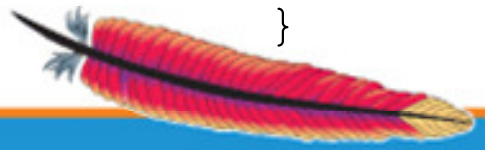
AJAX Process Flow



XMLHttp Object

- Http request to server using a JavaScript object
- Microsoft:
  - XMLHttp since IE 5.0 (ActiveX Object)
  - Instantiation:

```
if (window.ActiveXObject) //IE
{
  http_request = new ActiveXObject
    ("Microsoft.XMLHTTP");
}
```



XMLHttpRequest Object

- Mozilla, Safari..
  - XMLHttpRequest
  - supports same methods and properties

```
if (window.XMLHttpRequest) // Mozilla, Safari
{
  http_request = new XMLHttpRequest();
}
```

- function call after server response: (for both)

```
http_request.onreadystatechange = function(){
    // do useful stuff
};
```

XMLHttpRequest Object

- do the request:

```
http_request.open('GET', URL, true);
```

GET, POST, HEAD…

asynchronous/synchronous

AJAX!!

```
http_request.send();
```

- response ok, continue processing:

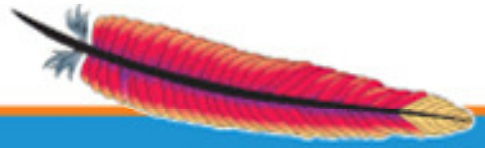
```
if (http_request.readyState == 4)
```

- check the status code:

```
if (http_request.status == 200)
```

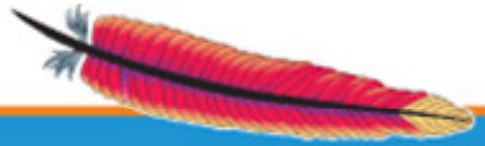
JavaScript Libraries

- abstraction layer
  - no handling of XMLHttpRequest object
- libraries provide…
  - ...plain AJAX support
  - ...visual effects
  - ...widgets (dojo toolkit)
  - ...ease of JavaScript handling



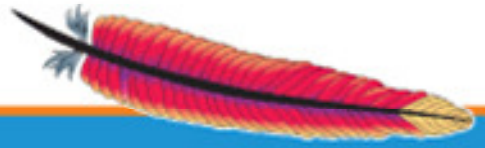
JavaScript Libraries

- prototype (Ruby on Rails)
  - open-source JavaScript framework
  - used in Tobago project
  - former used in MyFaces AJAX components
  - also used by script.aculo.us
- open rico
  - drag and drop management
- dojo toolkit framework
  - widgets
  - namespace structure



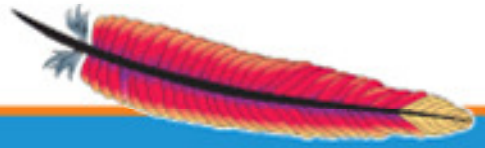
Why AJAX?

- richness and interactivity
  - web application -> desktop application
  - AJAX effects look great
- only parts of the page are changed
  - less redundant data
  - faster update of the page
- no stalling of user's interaction with the application
- No additional plug-in like flash is needed



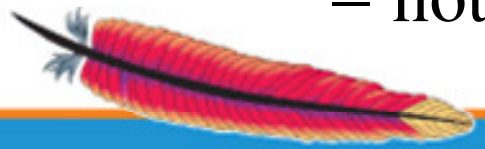
Why AJAX? (continued)

- platform independent
  - works across all browsers if JavaScript is enabled
  - works across all operating systems



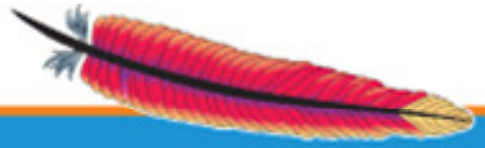
Why not?

- stressing the server
  - more requests
- state handling can be difficult
  - Browsers „back“ button
  - bookmarking
- need of JavaScript support
- differences of browsers -> more testing
  - not a big issue with JavaScript framework



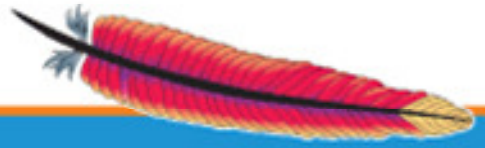
Agenda

- Introduction AJAX and Web 2.0
- **Integrating AJAX in JavaServer Faces**
  - **AJAX handling in MyFaces**
- AJAX components in MyFaces
- Discussion (or Question & Answer)



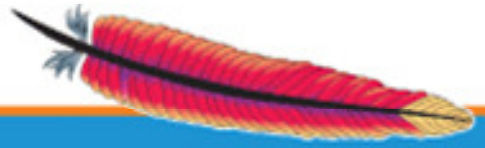
What is JSF?

- JSF is a ...
  - ... Java-Technology for Web-Apps
  - ... component-based framework
  - ... event-driven framework
  - ... RAD
  - ... industrial standard



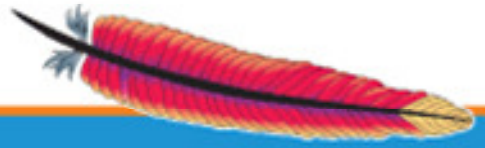
Ajax integrated into JSF

- Custom Servlets
- Filters
- PhaseListeners

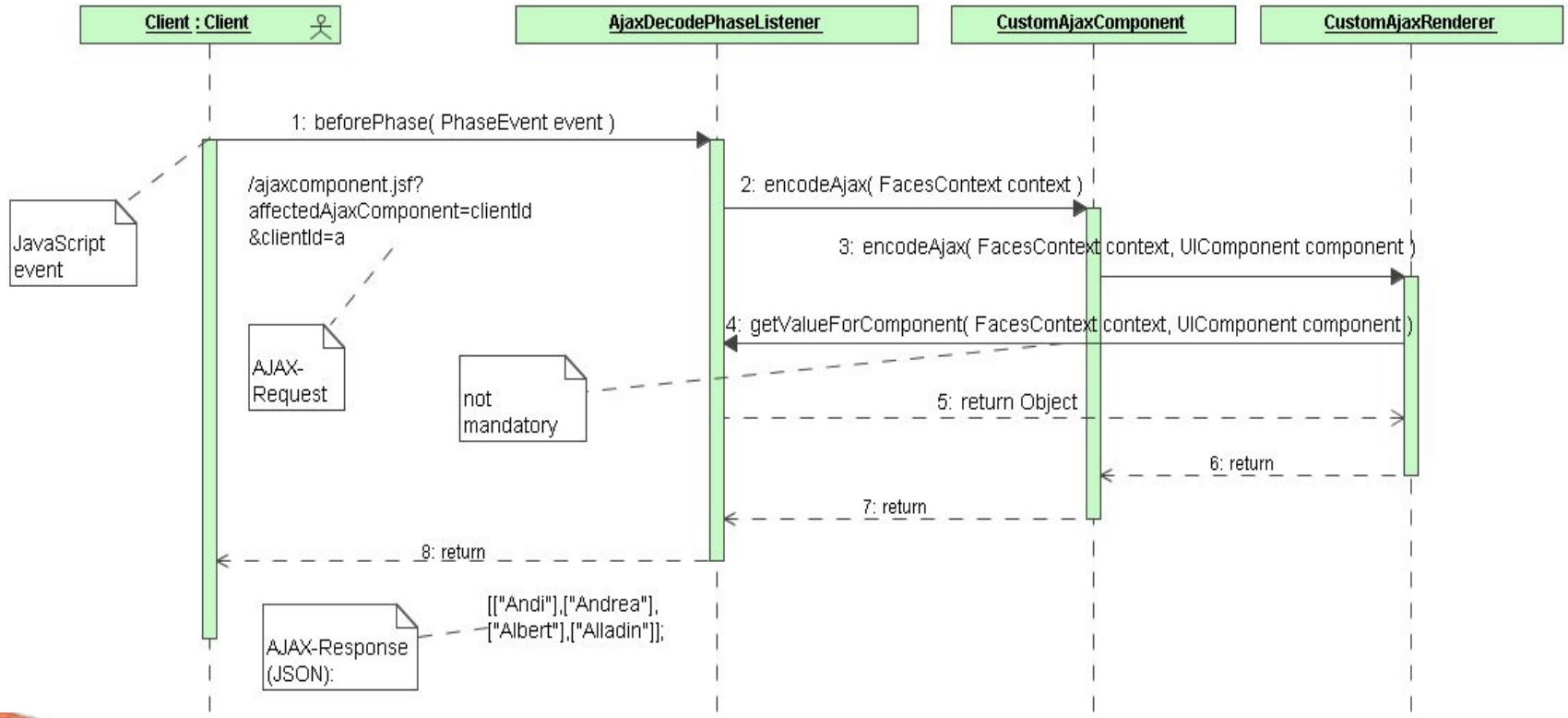


AJAX handling in MyFaces

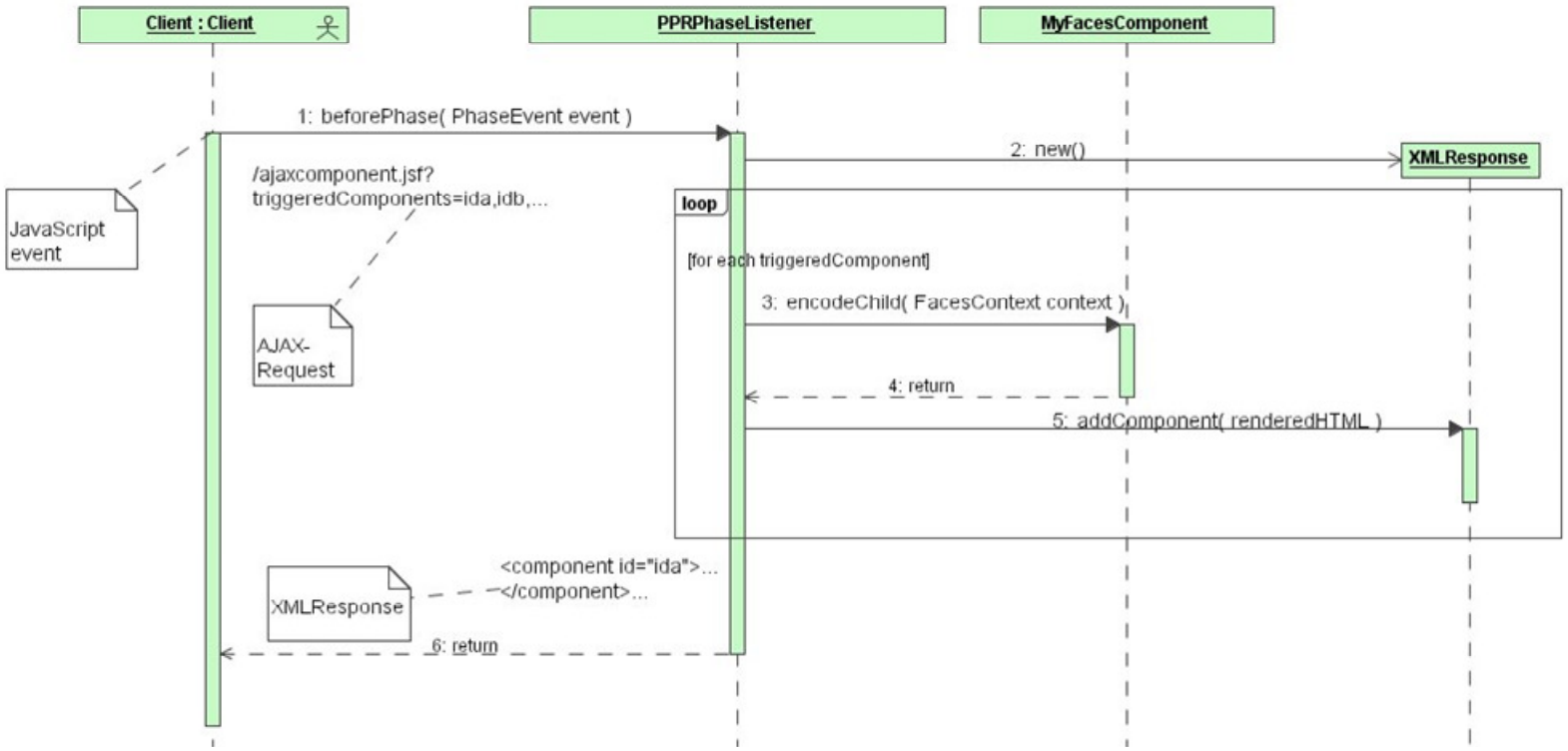
- AjaxPhaseListeners (AjaxDecodePhaseListener, PPRPhaseListener)
  - listening for an incoming AJAX request
    - tagged through Request Parameters which provide additional Information for the Request
  - seeks the affected components in the JSF tree
  - further processing delegated to component/renderer
  - quits JSF life cycle after Response has been sent



AjaxDecodePhaseListener (ApplyRequestValues)

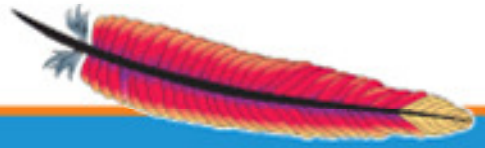


PPRPhaseListener (RenderResponse)



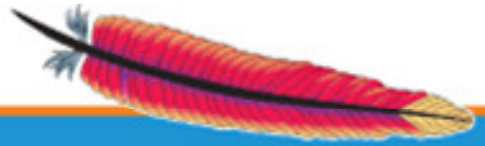
AJAX handling in MyFaces : problems

- No concurrent access on UIViewRoot allowed
- StateSaving
- Browser dependent memory leaks



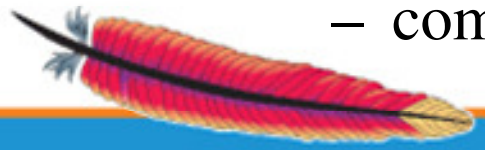
Agenda

- Introduction AJAX and Web 2.0
- Integrating AJAX in JavaServer Faces
- **AJAX components in MyFaces**
  - **Dojo`s toolkit**
  - **Tomahawk**
- Discussion (or Question & Answer)



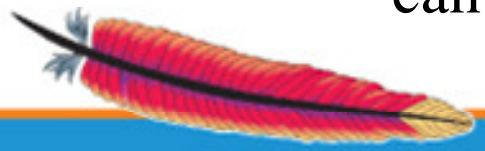
The Dojo Toolkit

- MyFaces has used prototype -> now Dojo!
- advantages
  - huge library
    - abstracts common js handling
    - AJAX api
    - animation
    - event handling
    - widgets!
  - clean separation through namespacing
    - avoids naming conflicts
    - supports lazy-loading of js-modules
  - compression of js code



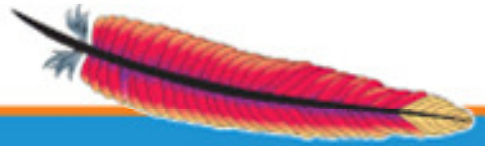
Dojo`s Widgets

- client side js components
- encapsulated js objects
- represent an abstraction layer
  - no need for html/dom handling
  - well defined interface: only one tag!
- hence easy to use
  - eg. `<input dojoType="comboBox" dataUrl="...">`
- not well-formed XHTML
  - can be also done programatically



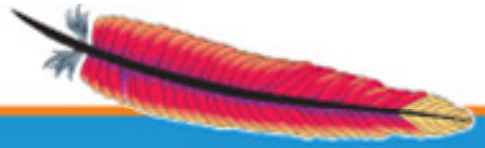
Dojo`s Widgets (continued)

- fast widget authoring
- client side widgets -> server side JSF comps
- ease of rendering
  - former: `<div/>, <input>, <table/>, js-code,…`
  - now: only one widget tag!
- under the hood
  - one widget = many html tags at runtime



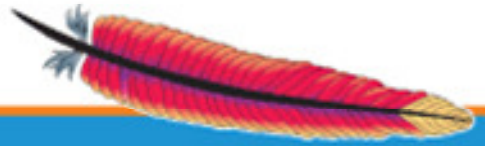
Dojo`s Widgets (continued)

- examples
  - ComboBox (InputSuggestAjax)
  - Menues (FishEyeList)
  - Accordion
  - Tree
  - Editor
  - TabbedPane
  - Wizard



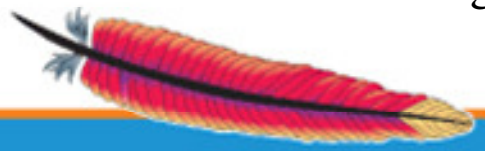
Tomahawk Examples: SuggestAjax

- sandbox components
  - autosuggest control with AJAX
- InputSuggestAjax
  - shows suggested list of items
  - completely based upon dojo`s comboBox widget
- TableSuggestAjax
  - suggests a html table
  - column value -> DOM node
  - based upon dojo`s js abstraction
  - dojo`s AJAX API (dojo.io.bind)



SuggestAjax

- suggestedItemsMethod
  - method of backing bean
  - delivers preview data
  - realized with JSF's MethodBinding
- call order
  - 1.) Ajax request
  - 2.) AjaxDecodePhaseListener calls SuggestAjax comp
  - 3.) delegates encoding to renderer
  - 4.) renderer calls suggestionMethod in bean (MethodBinding)
  - 5.) computing of result list
  - 6.) result sent back to client; dojo control shows suggestion drop down

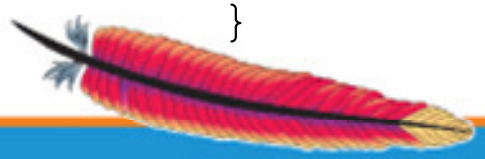


InputSuggestAjax: .jsp

```
<h:form>
  <h:panelGrid columns="3">
    <x:outputLabel
   value="#{label.title_product}"/>

    <s:inputSuggestAjax
      suggestedItemsMethod=
        "#{product.getSuggestedProducts}"
      value=
        "#{product.productNameToSearch}"/>

    <x:commandButton
      value="#{label.productButton}"
      action="#{product.searchForProducts}"/>
  </h:panelGrid>
</h:form>
```

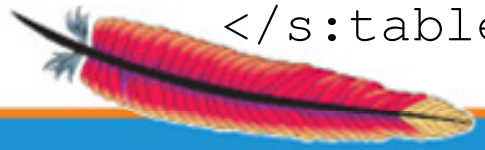
InputSuggestAjax:suggestedItemsMethod

```
public List getSuggestedProducts(String prefix)
{
   List<String> suggestedNames = new
   ArrayList<String>();

   Session session = getSession();
   Criteria crit =
   session.createCriteria(Product.class)
     .add(Restrictions.like("name", prefix+"%"));

   List<Product> tempProds = crit.list();

   for(Product prod : tempProds)
     suggestedNames.add(prod.getName());

    return suggestedNames;
}
```



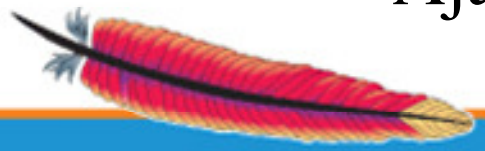
TableSuggestAjax: .jsp

```
<s:tableSuggestAjax var="address"
   startRequest="2“
   value="#{inputSuggestAjax.suggestValue}"
     suggestedItemsMethod=
     "#{inputSuggestAjax.getAddrList}">
   <t:column>
     <f:facet name="header">
       <s:outputText value="city"/>
     </f:facet>
     <s:outputText for="cityField"
       label="#{address.city}"/>
   </t:column>
   …
</s:tableSuggestAjax>
```



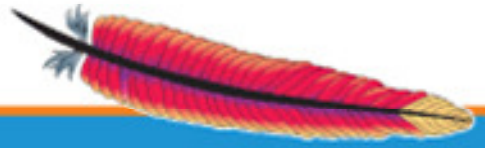
AutoUpdateDataTable

- sandbox component
- frequency controlled updated DataTable
- uses prototype.js
  - Ajax.PeriodicalUpdater(…)



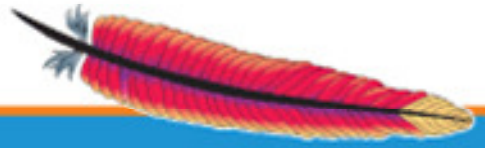
AutoUpdateDataTable: .jsp

```
<h:form>
   <s:autoUpdateDataTable
     var="bids"
     value="#{dataTableData.bids}"
     preserveDataModel="true"
     frequency="3">
     <h:column>
       <f:facet name="header">
         <h:outputText escape="false"
   value="Bid"/>
       </f:facet>
       <h:outputText value="#{bids.bidData}" />
     </h:column>
   </s:autoUpdateDataTable>
</h:form>
```



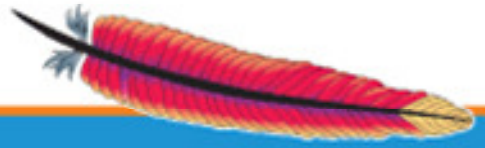
AJAX Form Components

- Server side validation through AJAX
- Model updating



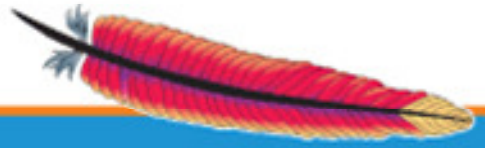
PPRPanelGroup

- Implementation of Partial Page Rendering in Apache MyFaces
- Each AJAX-Request Response fullfills a complete JSF-Request-Lifecycle (except RenderResponse which returns XML instead of HTML)
- Knows which Components trigger AJAX-Requests and which do not



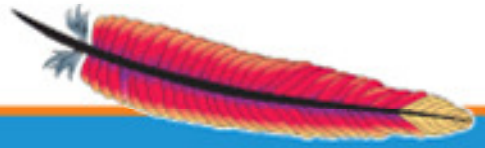
PPRPanelGroup - Attributes

- partialTriggers – List of Component Ids which trigger a partial update
- partialTriggerPattern – Regular Expression: all matching ClientIds trigger a partial update
- inlineLoadingMessage – While updating PPRPanelGroup is replaced with Message



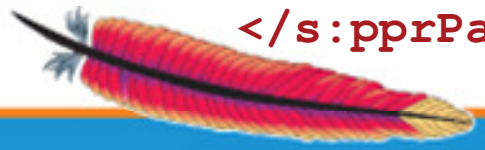
PPRPanelGroup: .jsp

```
<h:form id="mainform">
  <h:inputText value="#{exampleBean.suggestValue}"/>
  <h:commandButton value="update" id=„button" />

  <s:pprPanelGroup id="suggestValue"
    partialTriggers="button">
    <h:outputText
   value="#{exampleBean.suggestValue}"/>
  </s:pprPanelGroup>

</h:form>
```



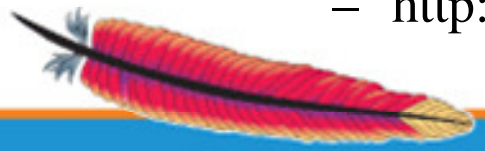
PPRPanelGroup – dataTable/dataScroller

```
<s:pprPanelGroup id="dataTableArea"
  partialTriggerPattern="mainform:.*">
  <t:dataTable id=„data“ var="address" ...>
    <t:column>
      <f:facet name="header">
        <h:outputText value="streetnumber"/>
      </f:facet>
      <h:outputText value="#{address.streetNumber}"/>
    </t:column>
    ...
    </t:dataTable>
    <t:dataScroller for=„data“>
     ...
   </t:dataScroller>
</s:pprPanelGroup>
```



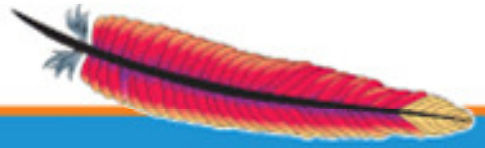
Links and books

- MyFaces AJAX examples
  - http://www.irian.at/myfaces.jsf
    (sandbox components)
  - http://tobago.atanion.net/tobago-example-demo
- AJAX web resources
  - http://www.script.aculo.us
  - http://www.adaptivepath.com
  - http://www.dojotoolkit.org
  - http://www.ajaxpatterns.org/Ajax_Frameworks
  - http://www.ajaxdeveloper.org



Questions?

Answers!!!



Finish

Thank you for your Attention!