Component Based Web Development with Apache Wicket

Web Application Development with just JAVA and HTML

- Karthik Gurumurthy,

Architect, Symcon Global Technologies

Author – Pro Wicket, Apress

Agenda

- What is Wicket
- Core concepts of Wicket
- Developing a custom component
- Q&A

The Java Ecosystem

‘ **When depressed, Women consume chocolates & go shopping while men invade other countries**’

- An “old” chinese proverb

‘ **When depressed, Women (still) consume chocolates & go shopping while men create “yet another open source java web framework”**’

- Karthik Gurumurthy ;-)

Luckily, Apache Wicket, by no means, gives an impression of having been created when under depression unlike a few others ;-)

Quotes

‘ Wicket is not a framework, it’s a candy bar and everybody loves candy bars..’

- Romain Guy, Lead Software Engineer, Java Swing Team, Author, ‘Filthy Rich Swing Clients’

'Dude, where are my objects?!’ and then I saw Wicket!

Apache Wicket in a Nutshell

- Open Source (ASF)
- Component Based
- Clean separation of Markup and Code
- Plain Old Java and Plain old HTML
  - No new templating / expression language to learn
- Minimum Configuration needs
  - Zero XML Framework (No Annotations either ☺)

- If you know Java and HTML, you already know quite a bit about Wicket!

Features

- Page Composition
  - Panels, Borders and Markup Inheritance
- Excellent Localization & Style Support
  - template and resource loading (_be.html, .xml)
  - localized models (e.g. for labels)
  - sophisticated resource bundle lookup (composition & inheritance hierarchy)

Features

- Integration
  - Spring
  - Guice
  - Hibernate
  - Jasper Reports
- Fancy Components
  - sortable, filterable, pageable, data aware tables
  - date picker, rich text editor, Google Maps
  - tabbed panel, navigation, tree, wizard

Features

- State management
  - type safe sessions
- clustering support
- back button support
- Double submit strategies
  - render redirect/ redirect to buffered response/ none
- Testing support
  - JUnit testing
- extensive logging and error reporting

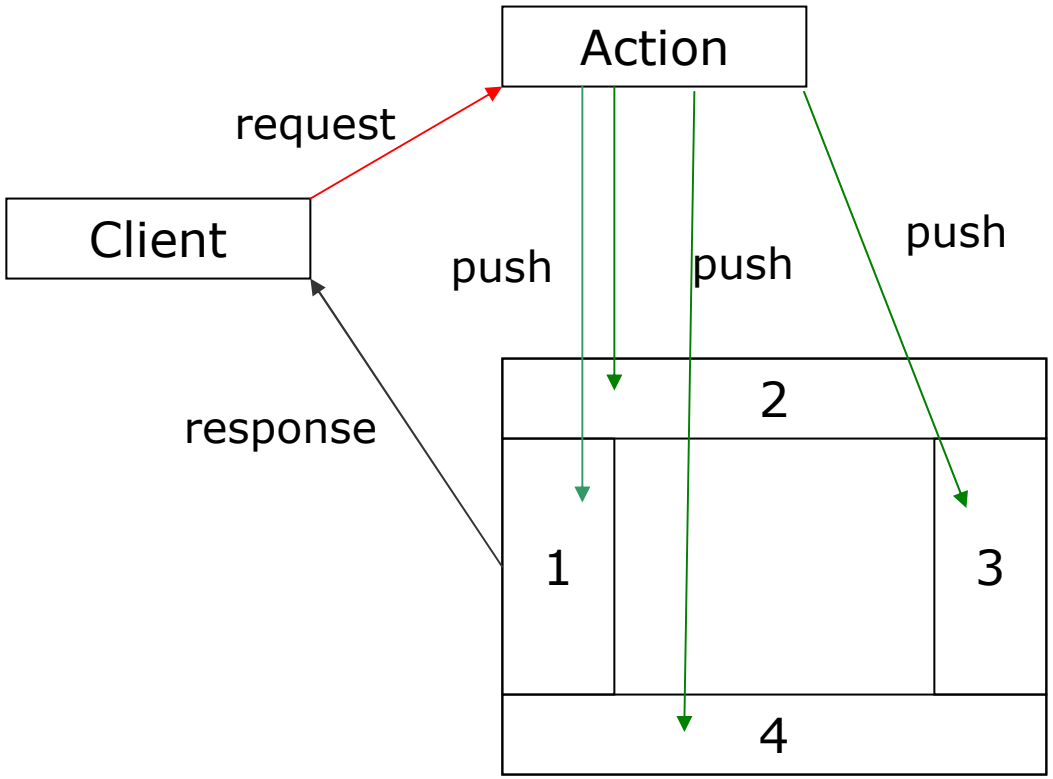
Features

- Ajax support and components
  - Ajax without writing JavaScript, Dojo, Scriptaculous, ...
- Header contribution
  - Javascript & CSS
- URL mounting
  - pretty URLs
- Component level security

Apache Wicket in a Nutshell

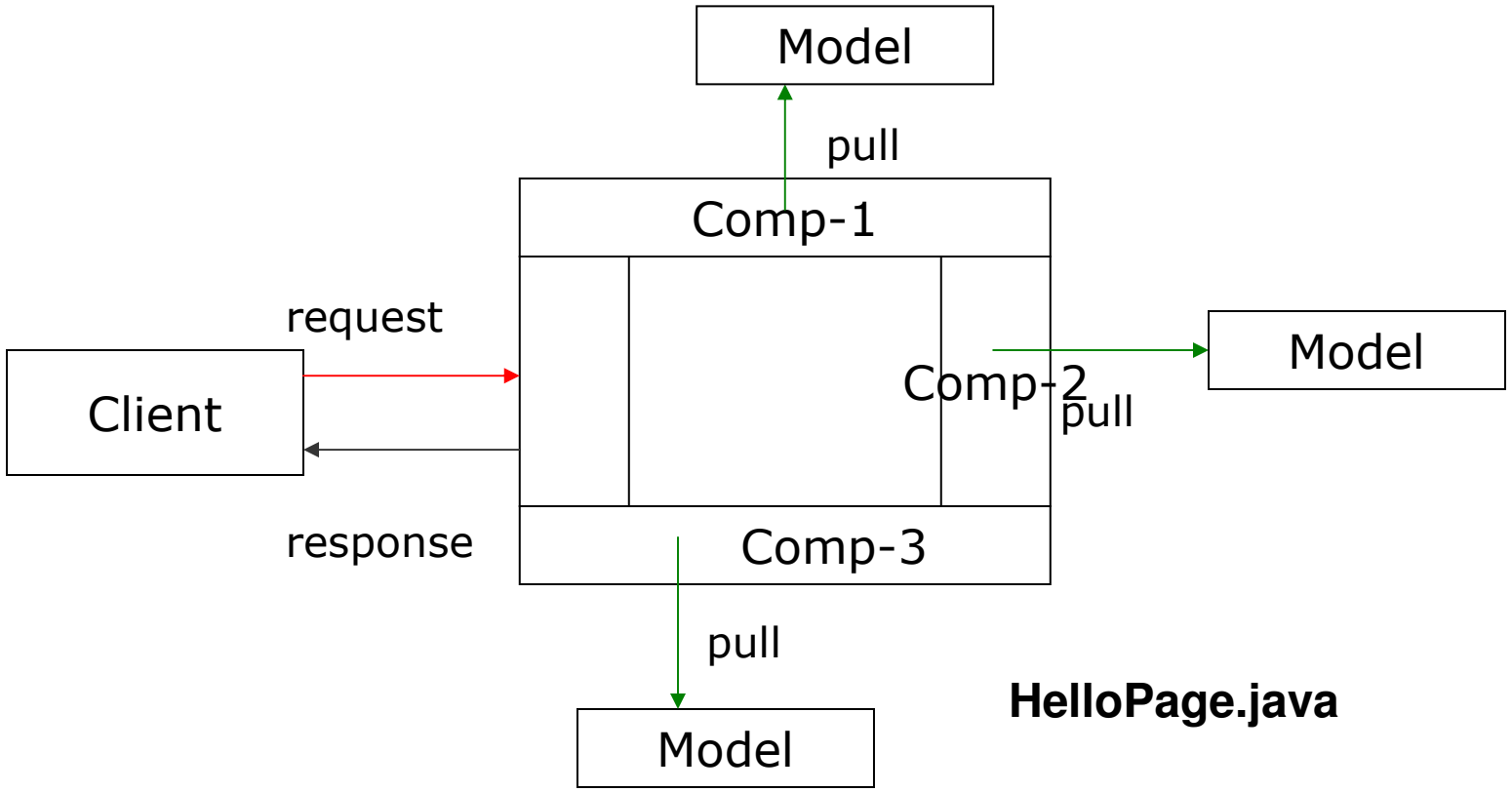
- 'Event' based as opposed to 'Action' based
- Mission Statement
  - A Java Software Framework to enable component-oriented, programmatic manipulation of markup

Model2 (Struts)



Hello.jsp

Component Oriented (Wicket)



HelloPage.java

Agenda

- What is Wicket
- Core concepts of Wicket
- Developing a custom component
- Q&A

Wicket Concepts

- Application
- Session
- RequestCycle
- Components
- Behaviors
- Models

Application

- Main entry point for your web application
- Configuration
  - Output Wicket specific tags?
  - Watch for markup changes every …?
  - Defines homepage
- Factories for
  - Session
  - RequestCycle
  - Security
  - …

Application

Configured in web.xml

```
<filter>
    <filter-name>wicket</servlet-name>
    <filter-class>
        org.apache.wicket.protocol.http.WicketFilter
    </filter-class>
    <init-param>
        <param-name>applicationClassName</param-name>
        <param-value>
            com.mycompany.MyApplication
        </param-value>
    </init-param>
    <load-on-startup>1</load-on-startup>
</filter>
```

Wicket Concepts

- Application
- Session
- RequestCycle
- Components
- Behaviors
- Models

Session

- Abstraction of a user session
- Storage typically in HttpSession
- Stores session specific data
  - Locale, Client info (browser vendor and version)
  - Your own data?
    - Logged in user
    - Shopping cart contents
  - Limited page history for back button support

Session

```
class ApplicationSession extends WebSession {
    private User loggedInUser;
    public User getLoggedInUser() { … }
    public void setLoggedInUser (User cart) { … }
}
appSession.setLoggedInUser(new User());
…
User loggedInUser = mysession. getLoggedInUser();
Permission[] permissions = loggedInUser.getPermissions();
```

Wicket Concepts

- Application
- Session
- RequestCycle
- Components
- Behaviors
- Models

RequestCycle

- Steps in a request cycle:
  - Create request cycle object
  - Decode the request
  - Identify request target (page, component, …)
  - Process event (onClick, onSubmit, …)
  - Respond (page, component, image, pdf, …)
  - Clean up

RequestCycle

- Two types of requests:
  - Stateful
    - Tied to specific user session
    - Not bookmarkable
  - Stateless
    - Not necessarily tied to specific user session
    - Bookmarkable

Wicket Concepts

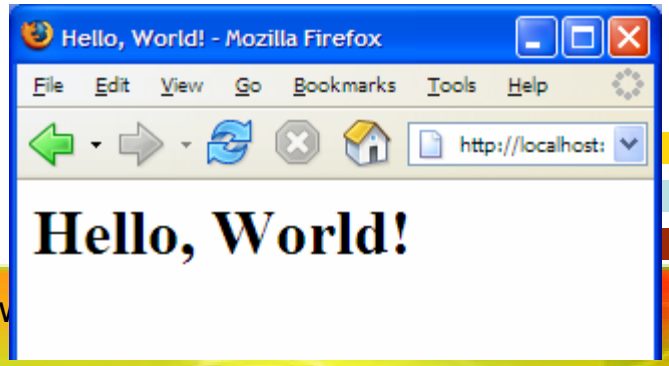
- Application
- Session
- RequestCycle
- Components
- Behaviors
- Models

Component

- Ultimate super class org.apache.wicket.Component
  - Label
  - MultiLineLabel
  - TextField
  - PasswordTextField
  - Image
  - Link
  - Tree
  - BookmarkablePageLink
  - Panel
  - ListView
  - Loop
  - PagingNavigator
  - ImageMap
  - Button
  - Ajax…
  - Sorting, paging repeaters
  - Wizard
  - DatePicker

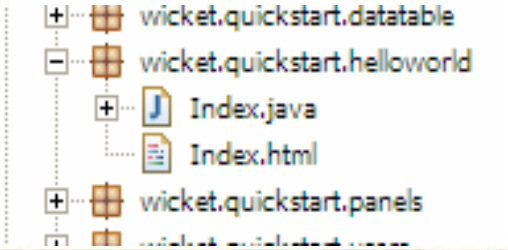
Components and Markup

- A component is identified in markup with wicket:id
  - Html:
    <h1 wicket:id="msg"**>Gets replaced</h1>**
  - Java:
    new Label("msg"**, "Hello, World!");**
  - Final (wicket tags can be stripped):
  - **<h1>Hello, World!</h1>**



Convention over configuration

- Component can have its own markup file
  - Page
  - Panel
  - Border
- Put java, markup and supporting files in same package on class path
- Can also place localization properties file in the same package



A Page: Hello, World!

```html
<html>
<head>
<title>Hello, World!</title>
</head>
<body>
<h1 wicket:id="msg">Message goes here</h1>
</body>
</html>
```

```java
package wicket.quickstart.helloworld;

import wicket.markup.html.WebPage;

public class Index extends WebPage {
    public Index() {
        add(new Label("msg", "Hello, World!"));
    }
}
```

Wicket Concepts

- Application
- Session
- RequestCycle
- Components
- Behaviors
- Models

Behaviors

- Behaviors are plug-ins for Components
- They can modify the components markup

```
item.add(new AbstractBehavior() {
    public void onComponentTag(
                Component component, ComponentTag tag) {
      String css = (((Item)component).getIndex() % 2 == 0)
                            ? "even" : "odd";
      tag.put("class", css);
    }
});
```

Output:

```
<tr class="odd">…</tr>
<tr class="even">…</tr>
```

Behaviors

- Change attributes of your component's markup
- Add javascript events
- Add Ajax behavior

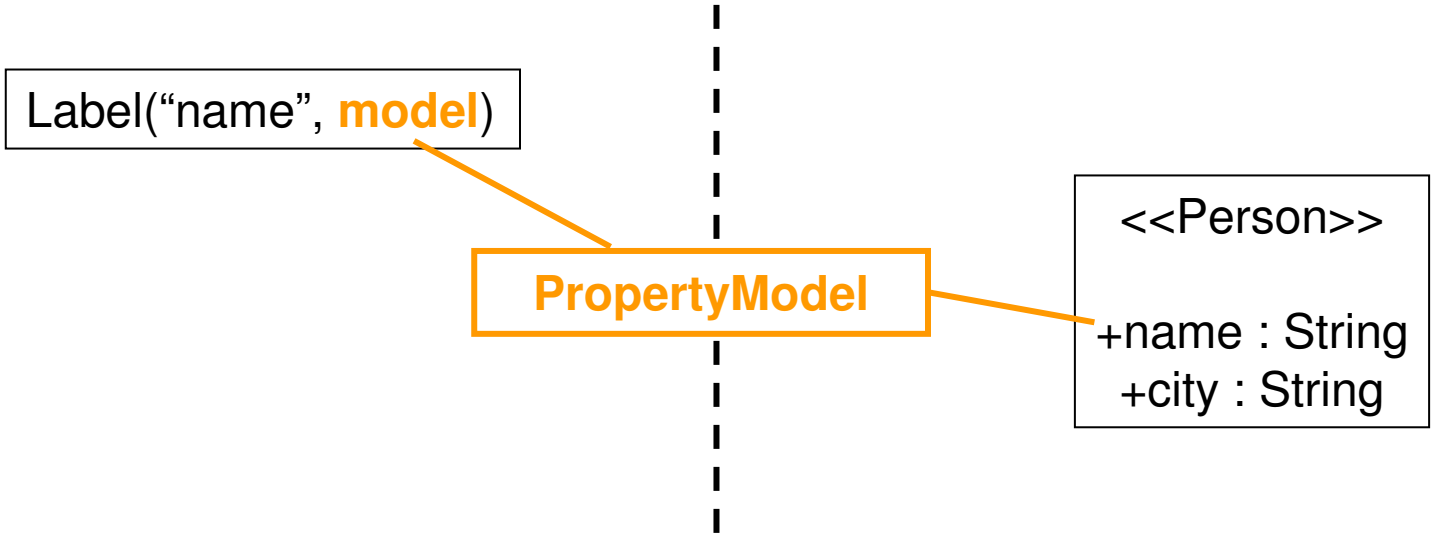
- **component.add(**
  new AjaxSelfUpdatingBehavior(
  Duration.seconds(1)));

Wicket Concepts

- Application
- Session
- RequestCycle
- Components
- Behaviors
- Models

Models

- Models bind your POJO's to Wicket components



Models

- Lazy binding in Java sucks
  - Doesn't update:
    - new TextField("txt", person.getName())
  - Gives null pointers:
    - new Label("street", person.getAddress().getStreet())
- Solution: OGNL/EL like expressions

Models

- PropertyModel:
  - new PropertyModel(person, “name”)
  - new PropertyModel(person, “address.street”)

Advanced Models

- StringResource
  - Internationalization & localization
- LoadableDetachableModel
  - Keeps session state minimal
  - Store only entity ID in session
- CompoundPropertyModel
  - Shared between components
  - Uses component ID for OGNL traversals

Compound Property Model

```
form.add(new Label("name", new PropertyModel(person, "name"));

form.add(new Label("birthdate", new PropertyModel(person, "birthdate"));

form.add(new TextField("street",
    new PropertyModel(person, "address.street")));
```

```
form.setModel(new CompoundPropertyModel(person));

form.add(new Label("name"));
form.add(new Label("birthdate"));
form.add(new TextField("address.street"));
```

Ajax

- Out-of-the-box
  - Dynamic updating of components in page
  - Update multiple components in one request
  - No JavaScript skills necessary
  - Several Ajax components and behaviors available
- Other Projects
  - Dojo integration
  - Scriptaculous integration
  - Yahoo! UI library integration

Wicket Ajax Components

- Wicket core
  - AjaxLink
  - AjaxFallbackLink
  - AjaxPagingNavigator
  - AjaxSubmitButton
  - AjaxSubmitLink
  - AjaxCheckBox
- Wicket extensions
  - IndicatingAjaxLink
  - AjaxEditableLabel
  - AutoCompleteTextField
  - UploadProgressBar
  - AjaxTabbedPanel

Agenda

- What is Wicket
- Core concepts of Wicket
- Developing a custom component
- Q&A

Wicket Lead on Custom Components

Eelco Hillenius:

« *Imagine being told that you can use Java as your programming language, but at the same time being told not to create your own classes. [...]*

I fail to understand why that has to be different for UI development, and Wicket proves it doesn't have to be so. »

Wicket Lead on Custom Components

- Creating custom component is as easy as typing in ‘extends’
- Extends wicket.Component down the line
- Available on application classpath

Custom Components

- Panels provide grouping
  - Have own markup file
  - Can be exchanged in pages for other components
  - Can contribute to header
  - Can contain any number of components, including panels

TogglePane HTML Markup (Designer)

```
<div id="I_D" class="pane_border">
   <div class="pane">
      <div class="pane_header">
         <span>Countries</span>
      </div>
      <div class="pane_content">
         <span>India, Pakistan, Australia</span>
      </div>
   </div>
</div>
```

TogglePane – JavaScript (toggle.js)

```
function init_toggle(){
   var panes = $$('div#I_D .pane');
   panes = $A(panes);
   panes.each(
        function(pane){
                var header = $A(pane.childNodes).find(
                                function(child){
                                    return child.className == 'pane_header';
                                }
                            );
                var content = $A(pane.childNodes).find(
                                function(child){
                                    return child.className == 'pane_content';
                                }
                            );
                header.onclick = function(){
                                    Effect.toggle(content,'blind');
                                };
        }
   );
}
```

Fetch all elements under a div with id 'I_D' having class 'pane'

For each pane , look for 'header' and 'content' elements.

Attach a 'onclick' listener to the 'header' to toggle the 'content' when clicked

Scriptaculous for "toggle" effect

Add the required scripts to the <head>

```
<head>
    <script src="toggle.js" type="text/javascript"></script>
    <script src="prototype.js" type="text/javascript"></script>
    <script src="scriptaculous.js" type="text/javascript"></script>
    <script src="effects.js" type="text/javascript"></script>
</head>
```

TogglePane Custom Component

Designer ensures that the HTML+ JS prototype works

Enter Wicket : Identifying Components

```
<div id="I_D" class="pane_border">
  <div class="pane">
    <div class="pane_header">
      <span>Countries</span>
    </div>
    <div class="pane_content">
      <span>India, Pakistan, Australia</span>
    </div>
  </div>
</div>
```

Repeating Block

Header

Content

Identifying Components (attach 'wicket:id')

```
<wicket:panel>
<div id="I_D" class="pane_border">
   <div class="pane“ wicket:id=“panes”>
      <div class="pane_header">
         <span wicket:id=“header”>Countries</span>
      </div>
      <div class="pane_content">
         <span wicket:id=“content”>India, Pakistan, Australia</span>
      </div>
   </div>
</div>
</wicket:panel>
```

This is a group of components – **A Panel !**

Repeating Block

Header

Content

Time to Code – TogglePane Component

Add “Dynamic Behavior” to the markup in Wicket

Identifying Components for Ajax Behavior

```
<div id="I_D" class="pane_border">
   <div class="pane“ wicket:id=“pane”>
     <div class="pane_header">
         <span wicket:id=“header”>Countries</span>
      </div>
      <div class="pane_content">
         <span wicket:id=“content”>India,</span>
      </div>
   </div>
</div>
```

“onclick” of header

‘Repaint’ content

Add wicket:id in order to attach Ajax Behavior

```
<div id="I_D" class="pane_border">
   <div class="pane“ wicket:id=“panes”>
      <div class="pane_header“ wicket:id=“h”>
         <span wicket:id=“header”>Countries</span>
      </div>
      <div class="pane_content">
         <span wicket:id=“content”>India, Pakistan, Australia</span>
      </div>
   </div>
</div>
```

On click of header

Repaint Content

Avoid FUD and illogical reasoning..

Q) “Why is JSF better than Wicket ? ”

A) “Well, uhhm because JSF is a standard”

http://www.theserverside.com/news/thread.tss?thread_id=45345

“I can stand brute force, but brute reason is quite unbearable”

- Oscar Wilde

Resources

http://wicket.apache.org

- **##wicket**
  - **irc.freenode.net**

Books:

Pro Wicket, Karthik Gurumurthy
Wicket in Action , Eelco Hillenius & Martijn Dashorst
Enjoying Web Development with Wicket, Kent Tong