Israel JBoss User Group
Session 09

Wicket

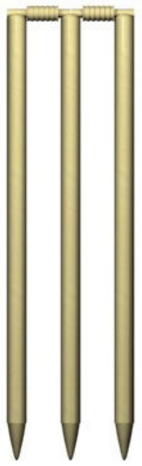
Ittay Dror
“Tikal Knowledge"
ittayd@tikalk.com

Agenda

- Introduction
- Architecture
- Getting Started
- Workflow
- Demo
- Q&A

What is Wicket

- Web framework
- Component oriented
- Stateful (optional)
- Design-logic separation



Concepts

- No XML
  - All configuration is in Java
  - GUI templates are HTML
- Java used for all GUI logic
  - Pages are represented as classes
  - Pages contain other objects
- Component Oriented
  - Components manipulate HTML

Advantages

- Templates are in HTML
- No split of logic – All logic in Java
- Component oriented
- Good support of AJAX
- Simple
- Strong community
- Many components & examples
- All-around: validators, borders...

Advantages – Advanced

- User navigation history
  - Even in multiple tabs
- Stateful components
- Model abstractions
- Internationalization
- Friendly URLs
- Debugging

Disadvantages

- Not a full RIA
- 'final' used a lot
  - Inhibits overriding
- Interaction with business logic tier and GUI logic are not separated

System Architecture

- Servlet filter initiates flow
  - Request for static files can go through
- Interaction with business tier through Java
- Components reference a model object
- Mapping of Java controllers and HTML templates through naming conventions

Components

- Java Class
  - » Inheritance
  - » Composition (Panels)
- Optional associated markup
  - » Inheritance
  - » Composition
- Page aware
  - » Header contribution
- Behaviors

Setup

- Create layout
  - mvn archetype:create
- Create a class extending Application
  - getHomePage() returns initial class
  - Can extend SpringApplication, DataApplication etc.
- Create class extending WebPage and matching HTML page

Hello World (! DEMO)

```
|-- pom.xml
|-- src
|   `-- main
|       |-- java
|       |   `-- com
|       |       `-- tikalk
|       |           `-- wicket
|       |               |-- HomePage.html
|       |               |-- HomePage.java
|       |               `-- WicketApplication.java
|       |-- resources
|       |   `-- log4j.properties
|       `-- webapp
|           `-- WEB-INF
```

Hello World

- Application class

```
public class WicketApplication extends
   WebApplication
{
   public WicketApplication() {}

   public Class getHomePage() {
      return HomePage.class;
   }
}
```

Hello World

- Template

```
<html>
  <head>
    <title>World Homepage</title>
  </head>
  <body>
    <strong>World Homepage</strong><br/><br/>
    <span wicket:id="message">message will be
  here</span>
    <form wicket:id="form">
      <input type="submit" wicket:id="btn"/>
    </form>
  </body>
</html>
```

Hello World

- Java Class

```
public class HomePage extends WebPage {
  public HomePage(final PageParameters
  parameters) {
     add(new Label("message", "Hello World"));
     add(new Form("form"){
       protected void onSubmit() {
         System.out.println("hello");
       }
     }.add(new Button("btn")));
}
```

View Flow

- Find the page object for the URL
- Tell it to render
  - » Find the HTML template and parse
  - » For each tag with wicket:id
    - • Find matching component
    - • Tell component to render
- Component manipulates HTML snippet
  - » Label: replace the body with the given text

View Flow

path/hello.html

HelloPage

Label("msg","Hello")

```
<html>
  <head>
   ...

 <span wicket:id="msg">Msg</span>

  </body>
</html>
    </body>
</html>
```

```
<html>
  <head>
   ...
   <span wicket:id="msg">Hello</span>

  </body>
</html>
```

Form Submit

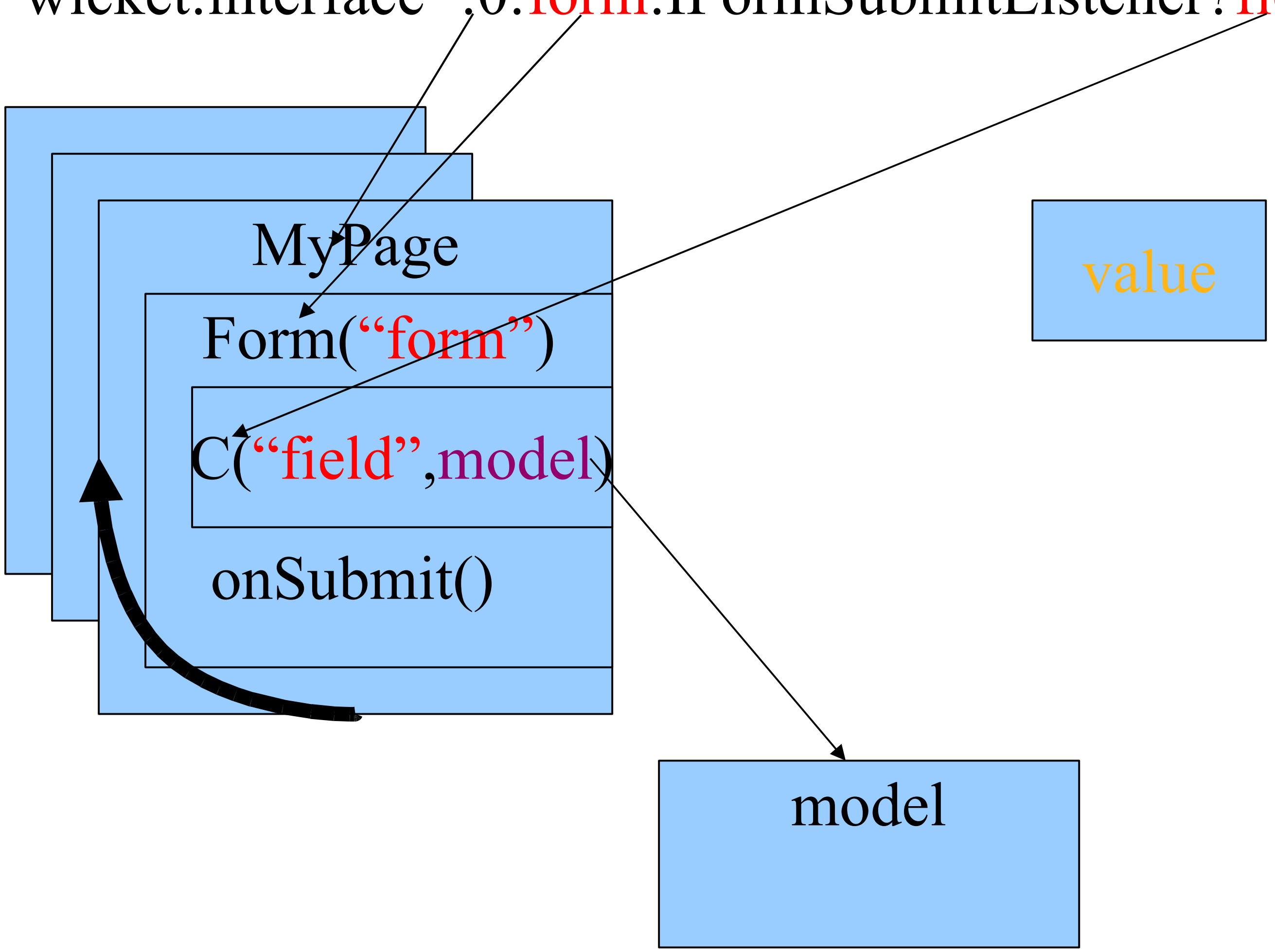
- action="?wicket:interface=:0:form::IFormSubmitListener::"
- Find form from path (:0:form)
- Call each component to read input
- User additional handling
- Redirect to page

- Register RequestListenerInterface by Name of the interface name
- Find by name in request
- GetPage from path
- Get component from path in page
- Invoke method on component
  - » Method from interface
- dispatchEvent(getPage(),url) ??

- IFormValidator

Form Submit

wicket:interface=:0:form:IFormSubmitListener?field=value



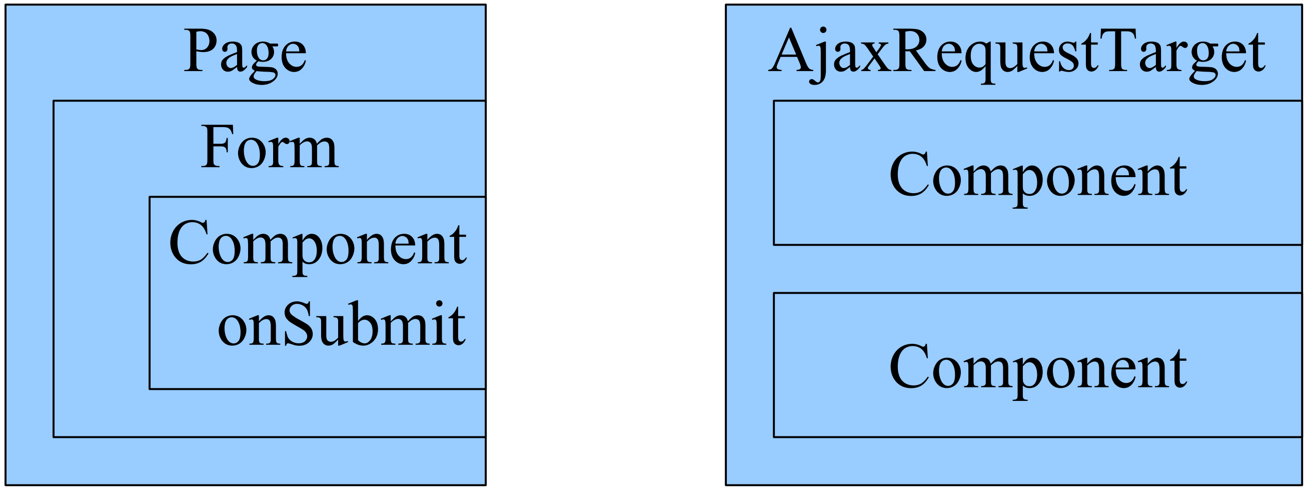
- Validate
- Convert

Ajax

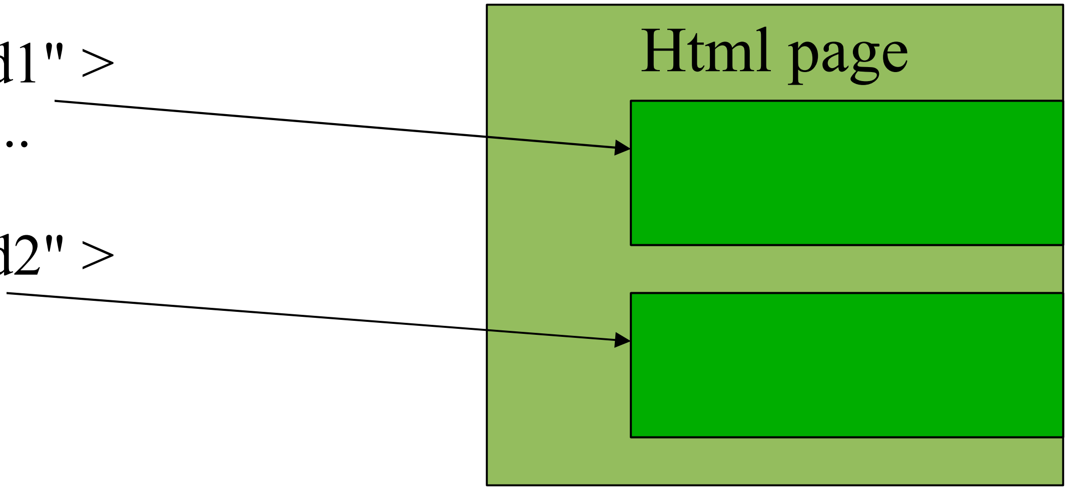
- Component is rendered to call ajax javascript
- Component is called
- Adds components to the 'target'
- Response lists each component's render with id
- Javascript replaces HTML according to ids

Ajax

```
<input ... onclick="var wcall=wicketSubmitFormById('addComment2'...
```



```
<ajax-response>
    <component id="id1" >
      <![CDATA[<div...
    </component>
    <component id="id2" >
      ....
    </component>
</ajax-response>
```



JBUG Amazing Brilliant Blog Application

- Or JABBA
- Simple entry list
- Links list
- Nice URLs
- Add comment
- Ajax add comment
- Search
  - » Panel, date picker, auto complete

Simple List

```
<html>
  <body>
    <strong>All entries:</strong>
    <span wicket:id = "entries">
      <p>
        <span wicket:id="date">
1/1/2004</span>
        <span wicket:id = "title">Title goes
here</span>
       </p>
    </span>
  </body>
</html>
```

Simple List

```
public HomePage(final PageParameters
   parameters) {
   super(parameters);
   add(new ListView("entries", getEntries()){
     protected void populateItem(ListItem
   item) {
         Jabba.Entry entry =
           (Jabba.Entry)item.getModelObject();
         item.add(new Label("date",
           format(entry.getDate())));
         item.add(new Label("title",
           entry.getTitle()));
     }
   });}}
```

Linked List

```
<a wicket:id="titleLink" href="#"><span
  wicket:id="title">Title</span></a>


item.add(new Link("titleLink") {
  {add(new Label("title",
      entry.getTitle()));}

   public void onClick() {
     setResponsePage(
       new EntryPage(entry));
   }});
}
```

Linked List

```
public class EntryPage extends WebPage {
  public EntryPage(Entry entry) {
    super();
    setModel(new CompoundPropertyModel(entry));
    add(new Label("head-title", entry.getTitle()));
    add(new Label("title"));
    add(new Label("date") {
      public IConverter getConverter(Class type) {
        return new DateConverter();
      }});
    add(new MultiLineLabel("text"));
    add(new ListView("comments"){
      protected void populateItem(ListItem item) {
        item.add(new MultiLineLabel("comment",
                                  item.getModel()));
      }});
  }}
```

Linked List

http://localhost:8080/jabba2/?wicket:interface=:1::::

Friendly URLs - EntryPage

```
public EntryPage(PageParameters params) {
  super(params);
  if (params.size() == 0) {
    setResponsePage(HomePage.class);
    return;
  }
  Entry entry =
    getEntry(params.getString("title"));
```

Friendly URLs - HomePage

```
PageParameters params = new PageParameters();
params.add("title", entry.getTitle());
params.add("year",
    String.valueOf(entry.getDate().getYear()+
      1900));
params.add("month",
   String.valueOf(entry.getDate().getMonth()));



item.add(new BookmarkablePageLink("title",
   EntryPage.class, params).
      add(new Label("title", entry.getTitle())));
```

Friendly URLs - Application

```
public JabbaApplication() {
   String[] paramNames = {"year", "month",
   "title"};
   MixedParamUrlCodingStrategy entriesUrls =
   new
     MixedParamUrlCodingStrategy("entries",
   EntryPage.class, paramNames );
   mount(entriesUrls);
}
```

Friendly URLs

http://localhost:8080/jabba3/entries/2007/11/Welcome To my blog/

Add Comment

```
<div>
  Leave a comment:
  <form wicket:id="addComment">
    <textarea wicket:id="comment"></textarea>
    <input type="submit" value="Submit"/>
  </form>
</div>
```

Add Comment

```
private Comment comment = new Comment();

add(new StatelessForm("addComment") {
  {
  add(new TextArea("comment", new
    PropertyModel(comment, "comment")));
  }
  protected void onSubmit() {
    entry.addComment(comment.comment);
    commentsListView.modelChanged();
    setResponsePage(EntryPage.class,
       Common.createParameters(entry));
  }
});
```

Ajax Comment

```
add(container = new
   WebMarkupContainer("ajaxContainer") {
   {
    add(commentsListView = new ListView("comments") {
...
add(new StatelessForm("addComment") {
{
    add(new TextArea("comment", new
   PropertyModel(comment, "comment")));
    add(new AjaxButton("submit"){
      protected void onSubmit(AjaxRequestTarget target,
   Form form) {
         entry.addComment(comment.comment);
         target.addComponent(container);
        }});
}
```

Ajax Comment – HTML

```
<div wicket:id="ajaxContainer">
  <div wicket:id="comments">
    <p wicket:id="comment">Comment</p>
  </div>
</div>
<div>
  Leave a comment:
  <form wicket:id="addComment">
    <textarea wicket:id="comment"></textarea>
    <input type="submit" wicket:id="submit"
  value="Submit"/>
  </form>
</div>
```

AjaxButton

- Extends Button
- Add AjaxFormSubmitBehavior
  - Inherits AbstractDefaultAjaxBehavior:
    - Adds <script> elements to <head>
  - Adds 'onclick' tag to the <input>
  - On callback, calls onSubmit

Search Posts - EntriesPanel

- Extracted from HomePage

```
public EntriesPanel(String id, List entries)
  {
  super(id);
  add(new ListView("entries", entries){
....
```

- HomePage:

```
 add(new BookmarkablePageLink("search",
  SearchPage.class));
 add(new EntriesPanel("entriesPanel",
  getEntries()));
```

Search Posts

```
<html xmlns:wicket>
  <body>
  <wicket:panel>
    <span wicket:id = "entries">
      <p>
        <span wicket:id = "date">1/1/2004</span>
          <a wicket:id = "titleLink"
  href="#"><span
  wicket:id="title">Title</span></a>
      </p>
    </span>
  </wicket:panel>
  </body>
</html>
```

Search Posts - HomePage

```
<a href="#" wicket:id="search">Search</a>
<strong>All entries:</strong>
<div wicket:id="entriesPanel"/>
```

Search Page Template

```
<html xmlns="http://www.w3.org/1999/xhtml">
<body>
  <form wicket:id="search">
    Date: <input type="text" wicket:id="date"
  size="10" />
    Title: <input type="text" wicket:id="title"
  size="20" />
    <input type="submit" wicket:id="submit"/>
  </form>
  <div wicket:id="results"></div>
</body>
</html>
```

Search Posts - SearchPage

```
private static class SearchTerm implements Serializable{
  Date date;
  String title;
}
public SearchPage() {
  super();
   Label label = new Label("results",
     "Enter some values in form");
   label.setOutputMarkupId(true);
   final String id = label.getMarkupId();
   add(label);

   final SearchTerm st = new SearchTerm();
   Form form = new StatelessForm("search",
       new CompoundPropertyModel(st));
   add(form);
```

SearchPage()

```
DateTextField.forShortStyle("date");
dateSearch.add(new DatePicker());
form.add(dateSearch);

AutoCompleteTextField titleSearch =
  new  AutoCompleteTextField("title") {
    protected Iterator getChoices(String input){
      return getEntryTitlesByPrefix(input);
    }
  };
form.add(titleSearch);
```

SearchPage()

```
form.add(new AjaxButton("submit") {
  protected void onSubmit(AjaxRequestTarget
   target, Form form) {
    EntriesPanel results =
      new EntriesPanel("results",
          getEntries(st));
    SearchPage.this.replace(results);
    target.addComponent(results, id);
  }
}
```

DatePicker

- A behavior
- Adds <script> elements
- Provides images:
  - resources/org.apache.wicket.extensions.yui.calendar.DatePicker/icon1.gif
  - icon1.gif in the same folder as DatePicker
- Writes HTML code
- Works with YUI

AutoCompleteTextField

- Extends TextField
- Adds AutoCompleteBehavior
  - » Adds javascript reference
  - » Render javascript code – key press
- On request:
  - » Find behavior (listener)
  - » Call getChoices with req. parameter
  - » Render results

Available Components

- DataTable (Toolbars)
- Wizard
- PageableListView, PagingNavigator
- TabbedPanel, AjaxTabbedPanel
- Dojo, yui, mootools, scriptaculous
- Tree
- Comet
- Swarm – component authorization/authentication

Thank You!

Questions?

AutoComplete

- AutoCompleteTextField
  - » getChoices(input)
- AutoCompleteBehavior
  - » wicket-autocomplete.js
    - Wicket.Ajax.Request(callbackUrl+&q=..)
  - » OnRequest
    - Render choices to target
    - requestCycel.setRequestTarget
  - » element.innerHtml=response

- Pagination?
- Completion? How does it work?
- wicket:link
- Fragements
- Triplets
- wasp