Internet Computing Workshop

Part4: Wicket
Contents

- A Birdseye view
- Why Wicket?
- Study of sample code?
- Wicket framework details
- Integration to Hibernate, Spring and Acegi
A high level view of the course

• Aim: to learn engineering of multi-tired web based systems
A Birdseye view of our progress

Next, the presentation layer using Wicket
Serverside vs Client side programming

Model:
- POJO
- Hibernate
- ORM
- Spring
- DAO
- Business Services
- Persistent operations

Wicket
Presentation tier
Acegi
Access Control
Presentation layer

You can do your presentation layer in any GUI form, but we are interested in **Web Application**: program that runs in a browser.

**Software as a Service (SaaS)**

**What is SaaS?**

**A brief review:**

Web browsers use **HTTP**: application layer protocol to allow interaction in a request/response form.
Presentation layer—a brief review

“50 first dates” HTTP is a stateless protocols (Designed to scale well!)
What is a state?
values of some of the variables— as well as the (sub-)pages used, user input values,…
Maintaining state is of crucial importance to us:
Yes! to benefit from the rest of the framework when doing a task (say using a transaction)?
So how does, say Servlets, maintain states?
What is Servlet?
Presentation layer—A brief review

What is Servlet?
- Java class conforming to `javax.servlet` API
- for adding Dynamic contents to a Web server
- created by JSP or Velocity Engine
- ....)

... back to question:

So how does Servlet deals with state?
Servlets can maintain state in session variables using cookies, or URL rewriting.
Presentation layer—a brief review

- There are all sorts of controversies about cookies: tracking users, security,…

Encoding states in URL:

- Security risk
- Does not scale (every form must know information about all other forms and their fields)
- Encoding from/to object (remove semantics information) is costly
Presentation layer—a brief review

Other challenging problems with JSP/Servelt:

• session support is via HttpSession objects implemented through hashmap
  – Key collision
  – low level - hard to modify

• Mixing of business logic and presentation—you can have dynamic code loop/if/… within your HTML, so business logic is in different places (hard to maintain)
What is wicket?

Because of the above shortcomings: in particular mismatch between stateless HTTP protocol and OO programming

Wicket:

Component based declarative Web application framework which allows state management and complete separation between presentation and logic.
What is wicket?

In Wicket:

1. Presentation part of the Web application, layout, forms, tables, … are defined using HTML without any dynamic elements

2. Components (pages, widgets, …) are programmed in Java constructs
Inspecting code

SimpleWicket application (see the webpage)

-SimpleWicket

|- src
  |- model
    |- Appointment.java

|- page

  |- Hello.java // pairs of file
  |- Hello.html

|- service

|- web // various CSS files
Three main parts of Wicket

1. Markup
2. Java Components
3. Model (we will see about this later)
Markup

**Markup** HTML code that describes widgets show to the user - Matches Wicket:id attributes and attaches Java Components to the tags in which the attribute is defined

```html
<html>
  <head>
    <title> Example page </title>
  </head>
  <body>
    <span wicket:id="message">
      <!-- message goes here -->
    </span>
  </body>
</html>
```

What is span anyway?
Java component

Extends **Component** to represent behaviour

```java
public class Hello extends WebPage {
    public Hello() {
        add(new Label("message","Hello World"));
    }
}
```

**Label component**, produces the string "Hello World" and replaces it in the body of `<span>`
Java component

Compare the two previous slides!
Framework matches Java Component and the Markup.
Key characteristic of Wicket is separation of Markup and Behaviour!
Java component

A Wicket Component class encapsulates the behavior and characteristics of the widgets:

- how they’re rendered
- how models are managed (we will see models later)
- how authorization is enforced
- whether a component should be displayed for a given context.
Java component

Check API for Wicket Components:
But briefly explaining
Part of Component Hierachy

Component

WebComponent
  Label

MarkupContainer
  children

WebMarkupContainer

FormComponent
  formComponents
  ...getInput()...

Form
  formComponents
  ...onSubmit()...

TextField
What happened to Markup for Label?

Only a few component classes (page, panel, border) have associated Markup files. The rest are assigned Markup snippet of code in their parents Markup file.

Example for Label with id "message":

```html
<span wicket:id="message">
   <!-- message goes here -->
</span>
```

⚠️ A very common mistake is matching of nesting in Markup and Java file!
What happened to Markup for Label?

🤔 A very common mistake is matching of *nesting* in Markup and Java file!

If you are adding Components inside a component, the Markup snippet of code must follow the nesting of Java file.
Model in Wicket

Model in Wicket is about where the data comes from. This is similar to the use of word **Model in MVC** and **NOT** models in Hibernate.
Model in Wicket

For sample see MakeAppointment.java
We will study the models later

Description of the Architecture- draw on the board

Now we will look at Java Components
Wicket components form a tree

For details of wicket components see the API
At the root we have Page (browser window)
org.apache.wicket.markup.html.WebPage;
Everything is added in a nested manner
The Label component is used to display text. It is bound to a <span> tag. See line 17-29 Hello.java Label(identifier, Model); Model can come from a Model object or a String of data
TextField

Single line processing of text
See line 64 of MakeAppointment.java

    TextField date = new TextField("appointmentDate",
        dateModel, Date.class);
    date.setRequired(true);
    date.add(new DatePicker());

Observe the use of Date picker().

Date specified in line 44

Notice TextFields can be used for SQL Injection - take care if using JDBC
Forms

You can also add various Forms
You need to implement `onSubmit()` for the behaviour
See line 39 of `MakeAppointment.java`
Announcements

Assessment 4 on Wicket will be held on Friday 19 March

Assessment 5 on Hibernate, Spring, Acegi, Wicket is in form of a project:
1) Will be online from next week
2) Viva on Wed 28 April in the Lab

This is the final assessment.
Indirections in accessing object

Models in Wicket are not Models in conventional sense!

Models are *Model Locators*
Why locators?

when object is not available at the time of component construction and instead needs to be retrieved from somewhere else at runtime.

Idea of wicket:

- extracts the value from the model
- render the corresponding component and sets its value when the containing HTML form is submitted

Let us see this in details:
Wicket is an event-driven framework

Look at onSubmit() method of form Line 41 of MakeAppointment.Java

How does it work?

Stages involved are:

1. Wicket HTML components register themselves as listeners
   1. defined through various Wicket listener interfaces
   2. Listeners listen to requests originating from the client browser.

2. When a client activity results in a request on a component, Wicket calls the corresponding listener method.
Rendering process

Remember:

1. All Wicket pages extend the WebPage class.
2. There is a one-to-one correspondence between the HTML widgets with a `wicket:id` attribute and the Page components.
When you click the OK button this happens behind the scene:

1. Page rendering process starts by calling the Page.render() method.

2. The Page finds the corresponding markup template and begins iterating over the HTML tags, converting them into an internal Java representation in the process.

3.1 If a tag without wicket:id is found, it is rendered as is.

3.2 If a tag with wicket:id is found, the corresponding Wicket component in the Page is located, and the rendering is delegated to the component.

5. The Page instance is then stored in an internal store called PageMap. Wicket maintains one PageMap per user session.

Draw picture
Rendering process

View the source code
First part is the java script
Go to line 78.
Observe the action which is taking place in line 99
Observe the rendered page after completion of an action
Links are used statically to link between pages with <a href> tags to allow navigation from one point to another. But what if there is a page which require direct access.

**BookmarkablePageLink**: a component to give others direct access to locations inside the application.
BookmarkablePageLink does not require navigation in conventional sense (where the user has been)

BookmarkablePageLink is good for things that we wish to share across users

Login pages
Pages related to choice
Pages to be visited often

See SWBorder.java in SimpleWicket
Links for performing actions

Links are good for performing actions
Link is an abstract class requiring implementation of onClick()
Which is called when the user clicks the link!
Hiding part of a page

We often need to hide part of a page because some condition is not met. For example, we don’t show the checkout until user has products in the shopping basket. Or when a user is logged in, login related links are often removed.

In some framework we can add a component to hierarchy, but in wicket it will not work.
Why?
Hiding part of a page

Why?
Because correspondence between markup and components must be preserved

But you can hide a component!

loginLink.setVisible(false);

See line 42-47 of AppointmentBorder, java of SimpleHibSprWicAce
Simplest Wicket project to start with

Five steps are involved

MyApp // this is the root
    |- src // to put all the source files
          |- myapp.hello
          |- Hello.java
          |- Hello.html
          |- MyApp.java // explained above
    |- JRE System Library
    |- context
          |- WEB-INF
                |- lib // to make Wicket jars available at runtime
                |- web.xml // deployment descriptor
Simplest Wicket: pages

You need a pair of Hello.java

```java
package myapp.hello;

import org.apache.wicket.markup.html.WebPage;

public class Hello extends WebPage{
    //nothing in it
}
```

and Hello.html

```html
<html> Hello World </html>
```
Simplest Wicket: pages

MyApp is to display your page- extend WebApplication

Must implement getHomePage() to say where is the homepage of the application

```java
public Class getHomePage() {
    return Hello.class;
}
```
Make jarfile available at Runtime

Make the library file available at runtime by copying them to context/WEB-INF/lib
Deployment descriptor

This is web.xml in context/WEB-INF

Observe the code:

<init-param>
    <param-name>
        applicationClassName
    </param-name>
    <param-value>
        myapp.hello.MyApp
    </param-value>
</init-param>
Registering Application with Tomcat

Create a file MyApp.xml in TOMCAT_HOME\conf\Catalina\localhost\MyApp.xml

With contents

```xml
<Context docBase="\$dir$/MyApp/context"
  reloadable="true"/>
```

• Says to Tomcat where application files are
• Reload the files when its class changes (recompile)
Running the application

You can do it within your IDE

Or simpler

1) Start tomcat by running startup.bat or startup.sh

2) Point your browser to http://localhost:8080/MyApp/app

Code a files available at SimplestWicket.zip

More complex applications require adding pairs of java/html and modifying MyApp.java