Web Frameworks and Tooling
What is it?
How do I use it?
And why should I want to?

Geertjan Wielenga
http://blogs.sun.com/geertjan
Sun Microsystems
Source Talk Tage, 27 September, 2006
Agenda

• Introduction
• 1. Why?
• 2. What?
• 3. How?
• 4. When?
• Conclusion
Once upon a time...

- JSP
- Servlet

But now...
These are currently supported in NetBeans IDE...
Plus... many more in-house frameworks...
How the IDE can help

- NetBeans module a.k.a. “plugin”
- manifest.mf
- layer.xml
- Java classes with functionality
- “Matisse” GUI Builder for design
- Easy to install in NetBeans IDE
- Easy to distribute as binary file to others
NetBeans module can provide...

- Creating a source structure
- Syntax highlighting
- Code completion
- Refactoring
- Hyperlinking
- Component Palette
- Multiview Editors
- Error annotations
...and also these:

- File templates
- Indentation engines
- Internationalization
- Options
- Project samples
- JavaHelp helpsets
- Junit testing
Web Framework Tooling...

Creating a source structure
Syntax highlighting
Code completion Options
Refactoring
File templates
Component Palette
Indentation engines
Multiview Editors
Hyperlinking
JavaHelp helpsets
Where everything begins...
Configuration Panel...

New Web Application

Steps
1. Choose Project
2. Name and Location
3. Frameworks

Frameworks
Select the frameworks you want to use in your web application.

- [ ] JavaServer Faces
- [ ] Struts 1.2.9

JavaServer Faces Configuration
JSF Servlet Name: Faces Servlet
Servlet URL Mapping: /faces/*

- [ ] Validate XML
- [ ] Verify Objects

< Back  Next >  Finish  Cancel  Help
When you click “Finish”...
Creating new artifacts...

1. Choose File Type
2. Project: WebApplication141
   Categories: Web, Enterprise, Java Classes, JavaBeans Objects, JUnit, NetBeans Module Development, Persistence, Web Services, Sun Resources
   File Types: Servlet, Filter, Web Application Listener, Tag Library Descriptor, Tag File, Tag Handler, HTML, JSF Managed Bean, Struts Action, Struts ActionForm Bean, XHTML
   Description: Creates a new managed bean class.
New JSF Managed Bean

Steps
1. Choose File Type
2. Name and Location

Name and Location

Class Name: NewJSFManagedBean
Project: WebApplication141
Location: Source Packages
Package: org.my.pkg
Created File: /Geertjan Wielenga/WebApplication141/src/java/org/my/pkg/NewJSFManagedBean.java
Configuration File: WEB-INF/faces-config.xml
Scope: request
Bean Description:

< Back  Next >  Finish  Cancel  Help

© 2006 Sun Microsystems
After you click “Finish”…
JAR Files...
Project Properties Extension...
Hyperlinking...

```xml
<?xml version='1.0' encoding='UTF-8'?>

<!DOCTYPE faces-config PUBLIC
 "-//Sun Microsystems, Inc.//DTD JavaServer Faces Config 1.1//EN"
 "http://java.sun.com/dtd/web-facesconfig_1_1.dtd">

<!-- =========== FULL CONFIGURATION FILE =========== -->

<faces-config>
    <managed-bean>
        <managed-bean-name>NewJSFManagedBean</managed-bean-name>
        <managed-bean-class>org.my.pkg.NewJSFManagedBean</managed-bean-class>
        <managed-bean-scope>request</managed-bean-scope>
    </managed-bean>
</faces-config>
```
Web Framework Tooling...

Creating a source structure

Syntax highlighting

Code completion

Options

File templates

Refactoring

Indentation engines

Component Palette

Multiview Editors

Hyperlinking

JavaHelp helpsets
Good news: Lots of help!

• 1. Great source code in NetBeans sources.
• 2. Open source projects:
  > https://nbwicketsupport.dev.java.net/
  > https://nbfaceletssupport.dev.java.net/
• 3. dev@openide.netbeans.org
• 4. Javadoc & Tutorials (let's have a look...)
• 6. Two very recent developments...
Contents

Registering a Web Framework

Prerequisite Knowledge

1. Introduction
   1.1 Preparing to Provide Support for a Web Framework
   1.2 The WebFrameworkProvider Class
   1.3 Getting Started Really Quickly
   1.4 Example: Basic Registration

2. Preparing to Work with the WebFrameworkProvider Class

3. Providing a Framework Configuration Panel
   3.1 Creating the Panel
   3.2 Example: Adding a Configuration Panel to the Web
   3.3 Coding the Panel

4. Creating a Source Structure and Files
   4.1 Preparing to Use the Extend() Method
   4.2 Example: Defining the Extend() Method
Let's browse through the chapter...
Let's put it into practise...

- platform.netbeans.org \(\text{\textit{(not www.netbeans.org)}}\)
- Look in Javadoc
- WebFrameworkProvider class (NetBeans Web API)
  \(>\) extend()
  \(>\) isInWebModule()
  \(>\) getConfigurationFiles()
  \(>\) getConfigurationPanel()
Wizard for Getting Started

1. Choose File Type
2. ...

Choose File Type

- Project: module32

Categories:
- Java Classes
- Java GUI Forms
- JavaBeans Objects
- JUnit
- NetBeans Module Development
- XML
- Ant Build Scripts
- Other

File Types:
- J2SE Library Descriptor
- Action
- JavaHelp Help Set
- File Type
- Module Installer
- Options Panel
- Project Template
- Update Center
- WebFrameworkProvider
- Window Component
- Wizard

Description:
Creates a Java Class that extends WebFrameworkProvider. Includes skeleton API methods. Must add dependencies on Web APIs, which includes the WebFrameworkProvider class. Must also register the class correctly in the layer.xml file.
Wizard for Getting Started (ctd.)
public class TapestryWebFrameworkProvider extends WebFrameworkProvider {

/** Creates a new instance of __NAME__ */
public TapestryWebFrameworkProvider() {
    super("Tapestry","Description");
}

//Empty for now, but this is where all the artifacts are created:
public Set extend(WebModule webModule) {
    return null;
}

//If true, the Project Properties dialog box will show that the framework is selected.
public boolean isInWebModule(WebModule webModule) {
    return true;
}

//Works with the framework's configuration files
public File[] getConfigurationFiles(WebModule webModule) {
    return null;
}

//Returns the lower part of the New Project wizard or the Project Properties dialog box, where the user can specify settings that are specific to your framework.
//Here, nothing is returned:
public FrameworkConfigurationPanel getConfigurationPanel(WebModule webModule) {
    return null;
}
}
...and three small little steps:

- Declare dependency on the module that provides the WebFrameworkProvider class
- Register the WebFrameworkProvider class in the layer.xml file
- Install the module.

...and then the framework is registered!
Demo!

1. Create module (plugin) project
2. Declare dependency on Web API
3. Use WebFrameworkProvider wizard
4. Register the framework provider class
5. Install the module
Summary

• Integrating support for a web framework
  > Wizards and templates
  > Pilot chapter for upcoming module development book
  > NetBeans API JavaDoc
  > Samples and Tutorials

• Need help?
  > dev@openide.netbeans.org
  > http://blogs.sun.com/geertjan

• Which frameworks do you want support for?
Questions & Answers