



[java.com.sun/javaone](http://java.com.sun/javaone)

# Choosing Your Java Based Web Framework: A Comparison

Richard Pack, Lead User Interface Architect at Hyperic Inc.

TS-6457



# Learn how to choose a Java Web Framework

*Enable you to:*

- ▶ *Categorize and understand what type of UI you're building*
- ▶ *Understand the features and nomenclature of Model View Controller (MVC)*
- ▶ *Weigh the trade-offs*

GOAL

# Agenda

- Current Web Frameworks
- The Traditional Criteria
- What are the problems that users are facing?
- What are your needs?
- The Advanced Criteria
- The Frameworks
- Summary and Conclusions
- Questions

# Current Frameworks

Action Framework	DWR	JSPWidget	OpenEmcee	SwingWeb
Anvil	Echo	Jucas	OpenXava	Tapestry
Aranea	Espresso	JWAA	OXF	TeaServlet
Aurora	fleXive	JWarp	Pustefix	ThinWire
Baritus	Folium	jWic	Restlet	Trimpath Junction
Barracuda	GWT	jZeno	RIFE	Turbine
Bento				Verge
Bishop				VRaptor
Canyamo				Warfare
Cassandra	Jaffa	Maverick	Smile	WebOnSwing
Chiba	Japple	Melati	SOFIA	WebWork
Chrysalis	JATO	Mentawai	Spring MVC	Wicket
Click	JFormular	Millstone	Strecks	wingS
Cocoon	JOSSO	MyFaces	Stripes	Xoplion
Dinamica	JPublish	Nacho	Struts	ZK
Dovetail	JSF	Niggle	Struts 2	ztemplates

Over 100 Frameworks

## So why is this important

- Difficult for the stakeholders to understand and use
- End up writing lots of non-reusable code
- No unified Ajax transport
- Direct persistence to Request, Session or Context
- Limited Documentation
- Writing boilerplate code
- No unified event model
- Difficult for traditional web designers
- Configuration becomes difficult to manage when the app grows
- No way to run a debugger in the templates
- Ability for the view to be used as the controller
- JSP/JSTL becomes a nightmare to read when the page grows
- Templates may have a steep learning curve
- Speed of deployment

# The Criteria

## The traditional set

- The Hello World test
  - What does it take to write Hello World?
- Performance
  - Filter or servlet?
- View decoration style
  - Does it use Templates or JSPs/Taglibs, java control of blocks, java rendering?
- Speed of Deployment
  - What does it take to ship it?
- Speed of Development - 3 Parts
  - New - what is the learning curve?
  - Existing - how much time does it take to add simple features?
  - Agile - is changing features easy?

# The Criteria

## The traditional set - continued

- Validation
  - Does it have lots of build-in validators?
- Internationalization / Multilingualization
  - What style of properties? Varying scopes?
- Documentation
- Bookmark-ability
- Scalability

# What are my needs?

- Am I building an application?
- How big is my application?
- What is my usage scenario?
- Is this likely to change?
  - You are building for the future.
- Do I really need a 3-tier architecture?
- What is the size and competencies of my team?

# Sweet Spots

- Not a one size fits all game
  - Some requirements may have diverging implementations which creates trade-offs
- Different frameworks for different usage scenarios
  - View (Read) only. Ex - storefront, blog, online bank, corporate website
  - Write only. Ex - shopping cart, signup process
  - Both view and write
  - Highly Interactive - CRM, Application builder, Storefront/Workflow builder

# The Criteria

## The advanced set

- AJAX support
  - Integrated or after the fact
  - Does it rely on external tools
- View language
  - Java, EL, OGNL, Multiple
- Community
  - Active, continual development
- Maturity
- Push based or Pull based
- 3-tier support - EJB/Spring and hibernate
- Friendly URLs ala URL mapping
- Client/Server Javascript
  - Is there tight coupling between the languages
- Does another part of the company use the framework?
- Do your developers have experience with it?
- Does it have an Archetype

# The Frameworks

# Groovy/Grails

- Usage type: View / Write
  - Extension to the Java language
- Pros:
  - Hot-deploy
  - Simple configuration
  - Closures
  - Integrates Spring and Hibernate
  - Nice form builder
- Cons:
  - Slow compared to native frameworks

```

<!-- template code -->
<div id="table"
class="hidden">
    <%=dojoTable(id: 'Defs',
title:l.ClassicDefsTable,
url:urlFor(action: 'defData
'), schema: defSchema,
numRows:15) %>
</div>
  
```

```

//Groovy
static String dojoTable(Binding
scriptBinding, params)

def id = "${params.id}"

def res = new
StringBuffer("")
<script type="text/
javascript">
dojo.require("dojo.topic");
  
```

# Groovy/Grails

AJAX support?	Yes
View Language	Java,Groovy
Push or Pull	Push
Maturity	Young
Community	Small, but growing
3 Tier F/W support	Yes
Friendly URLs	Not built in

# Google Web Toolkit (GWT)

- Usage Type: View / Write
  - Modified Java language support
- Pros:
  - Simplicity, Swing/SWT Like
  - Lots of built in components
  - Strong binding between Java-Javascript
  - Project Archetype
- Cons:
  - Hard to control css classes
  - Built in components are fairly basic (Solved with GWT-Ext)
  - Currently no Java5/6 support (In beta)

# Google Web Toolkit (GWT)

```
public class Mail implements EntryPoint, ... {  
  
    public void onModuleLoad() {  
        topPanel.add(menuBar);  
        rightPanel.add(list);  
        mainPanel.add(table);  
  
        Window.enableScrolling(false);  
        Window.setMargin("0px");  
  
        DockPanel outer = new DockPanel();  
        outer.add(topPanel, DockPanel.NORTH);  
        outer.add(leftPanel, DockPanel.EAST);  
        outer.add(mainPanel, DockPanel.CENTER);  
  
        History.addHistoryListener(this);  
  
        RootPanel.get().add(outer);  
    }  
}
```

# Google Web Toolkit (GWT)

AJAX support?

Yes

View Language

Java generated

Push or Pull

Pull

Maturity

Young

Community

Small

3 Tier F/W support

No

Friendly URLs

Yes, with some work

# Struts 2

- Usage Type: View / Write, Highly Interactive
- Pros:
  - First class AJAX support
  - Spring container integration
  - POJOs as controllers, doesn't require inheritance
  - FreeMarker and Velocity support
- Cons:
  - Still uses **LOTS** of XML configuration
  - Still leaves the flow mapping to XML

## Struts 2

```
public class LoginAction implements Preparable {  
  
    public LoginAction(LoginService service) {  
        this.service = service;  
    }  
  
    public String execute() {  
        return Action.SUCCESS;  
    }  
  
    public String login() {  
        service.login(un, pw);  
        return execute();  
    }  
}
```

```
<s:textfield id="userName" label="UserName" name="login.firstName"/>  
<s:textfield id="password" label="Password" name="login.lastName"/>  
<s:submit theme="ajax" targets="login" notifyTopics="/login"/>
```

# Struts 2

AJAX support?	Yes
View Language	JSP, OGNL
Push or Pull	Push
Maturity	Young
Community	Medium
3 Tier F/W support	Yes
Friendly URLs	Not built in

# Tapestry

- Usage type: View / Write, Highly Interactive
- Pros:
  - Uses annotated HTML templates
  - Hot-deploy (Version 5)
  - Simple configuration
  - First class AJAX support, and lots of it
  - Nice form builder
- Cons:
  - Documentation doesn't use real world problems
  - No API backwards compatibility between major versions

# Tapestry

```

<!-- template code -->
<div jwcid="@hyperic:MessagePanel" message="message"/>
<input jwcid="@hyperic:TextField" fieldValue="userName"
  fieldTitle="message:userNameTitle" fieldLabel="message:userName"/>
<a jwcid="@hyperic:Button" listener="listener:signInButtonListener"
  label="message:signInButton" enableKeyListener="true" />
  
```

```
//Java Controller
```

```

public abstract class SignIn extends BasePage {

    @Persist()
    public abstract String getUsername();
    public abstract void setUsername(String userName);

    public ILink signInButtonListener(IRequestCycle cycle)
    {
        //button click listener
        ...
    }
  
```

# Tapestry

AJAX support?	Extensive
View Language	OGNL, Prop
Push or Pull	Pull
Maturity	Mature
Community	Large, Active
3 Tier F/W support	Yes
Friendly URLs	Yes

# Wicket

- Usage Type: Highly Interactive
- Pros:
  - Annotated HTML Templates
  - Excellent documentation
  - Components can be extended
  - Everything is done in Java
- Cons:
  - Everything is done in Java

# Wicket

//Definition

```
public class Login extends WebPage {
    public Login() {
        add(loginPanel);
    }
    public void onSubmit() {
        login();
        setResponsePage(new HomePage());
    }
}
```

```
<wicket:panel><div wicket:id="login">
    <div><span wicket:id="username"></span></div>
    <div><span wicket:id="password"></span></div>
</div></wicket:panel>
```

// Usage

```
add(new LoginPanel("login", login);
```

```
<span wicket:id="login"/>
```

# Wicket

AJAX support?	Extensive
View Language	OGNL
Push or Pull	Pull
Maturity	Moderate
Community	Medium, Active
3 Tier F/W support	Yes
Friendly URLs	Yes

# Summary

- Not all frameworks are equal
- Understand and outline your needs
- Find a few that look like they meet those needs
- Use the criteria and evaluate
- Don't take my word for it, try a "Hello World" of your own

## For More Information

- [rich.javaone@gmail.com](mailto:rich.javaone@gmail.com)
  
- These slides
  - [download.hyperic.com/pdf/WebFrameworkComparison.pdf](http://download.hyperic.com/pdf/WebFrameworkComparison.pdf)
  
- Java Framework “Sweet Spots”
  - [www.virtuas.com/files/JavaWebFrameworkSweetSpots.pdf](http://www.virtuas.com/files/JavaWebFrameworkSweetSpots.pdf)
  
- The Frameworks
  - [grails.codehaus.org/](http://grails.codehaus.org/)
  - [code.google.com/webtoolkit/](http://code.google.com/webtoolkit/)
  - [struts.apache.org/](http://struts.apache.org/)
  - [tapestry.apache.org](http://tapestry.apache.org)
  - [wicket.apache.org](http://wicket.apache.org)

# THANK YOU



## Choosing Your Java Based Web Framework: A Comparison

Richard Pack

TS-6457

