Index

This wiki is dedicated to documenting the **Wicket Java application framework**. Wicket takes simplicity, separation of concerns and ease of development to a whole new level. The wiki is currently open to new users and contributors; see the contribution page for more information. To download Wicket, please visit the Wicket site.

Table of contents

About Wicket

What is Wicket?

Wicket is one of the most recent in a long line of Java web development frameworks and stands on the shoulders of many that have come before it. Wicket is a component-based framework, which puts it in stark contrast to some of the earlier solutions to the sometimes monotonous task of web programming. Like other frameworks, Wicket builds on top of Sun's servlet API; however, unlike frameworks like Struts or Spring MVC, the developer using Wicket is mostly removed from the request/response nature that is inherent with the web and Servlets. Instead of building controllers that must service many users and threads simultaneously, taking in requests, returning responses, and never storing any state, the Wicket developer thinks in terms of stateful components. Instead of creating a controller or action class, he or she creates a page, places components on it, and defines how each component reacts to user input.

This may all sound familiar to those with desktop GUI experience; Microsoft's Visual Studio line, Sun's Swing API, and Borland's Delphi are all popular desktop GUI development tools that use component models. Using components allows the developer to spend less time on the visual tier of his or her app and more time implementing the core functionality. Even more important is how extensible this makes component-based GUIs. Adding additional functionality is simply a matter of adding one more component, which can act independently or in cooperation with other components in the view. These advantages have not been lost on web developers. In fact, many web framework projects have attempted to leverage the productivity and scalability of desktop applications. Apache Jakarta's Tapestry and Microsoft's own ASP.NET as well as Sun's Java Server Faces specification all present solutions to component-based development over the web and bring new ideas to the table. All of these technologies separate the page layout into a template file. JSF uses Sun's JSPs, ASP.NET uses ASP, and Tapestry use's it's own templating system based on standard HTML markup. These pages are rendered on each request, and as they are rendering, they make calls into a backing

class to support dynamic content. As much as the word "template" would seem to suggest otherwise, this makes the page template king. Backing classes tends to be a series of listener methods, at the total mercy of the page template that is supposed to be merely defining the placement of components.

This works fine, and it is definitely a step up from a model 2 controller singleton class. Instead of a giant if block, we have well defined methods. Instead of being stateless, we can have instance variables. But now our Java code is a second-class citizen, existing merely to provide the page with the information it needs while it renders itself. Also, this backing class is always aware of the request-response cycle. It knows that getMember() is going to be called on every request, and reacts accordingly by getting a fresh copy from the database. Sure, the developer needs to deal with these details, but does it really have to be dealt with at the very top level of the application? Wicket allows the developer to build a page in Java (you remember Java right? It's like OGNL, but with even more cool features) that uses and manipulates an HTML file, not the other way around. This page exists across requests, and does not even need to be aware of the request/response cycle. But something needs to know when a new request is starting and when the last one has finished rendering, right? All kinds of problems crop up when an object/relational mapper tries to work with an object from the last request, or the user is looking at an object that no longer represents what's actually in the database, or when you merely try to store all that data in the session. The solution here is to not make the page aware of all the request details, but the data itself. That's what Wicket tries to do.

Introduction to Java web applications

- Introduction on Java web applications A must read for anyone developing web applications with Java. We presume you know this before you start working with Wicket.

Why Wicket

- An introduction by Jonathan Locke (father of Wicket)

Framework Comparisons

- Wicket and Struts
- Wicket and Tapestry
- Wicket and JSP / Spring MVC / WebFlow (1.0)
- Wicket and JSF (blog post)
- Wicket and GWT (blog post)
- Wicket and Seam / JSF (blog post)
- Wicket, Tapestry 5 and Grails (blog post)

Who is using Wicket

- Websites based on Wicket
- Wicket Products
- Related Projects and Tools adding additonal value to the World of Wicket (WoW)

Where to (get) help

IRC

- For those interested in a more direct support, please join IRC: ##wicket at irc.freenode.net. See Wicket IRC for more information

Community initiatives

- Meetups

Communities on social networking sites

- Wicket @ LinkedIn You are welcome to join.
- Wicket @ Xing Same applies here, join.

Contribute to Wicket

- Contributing to Wicket - Report a bug, Build a quickstart, Submit a patch...

Commercial Services

- Companies that provide services such as training, consultations and support

What's next

Wish List for Next Version

- Wicket Wish List
- ~~Wicket Ajax rewriting~~

Migrations

- Migration from Wicket 1.5 to Wicket 6.0 ~~*(6.0 work in progress; not yet released)*~~
- Migration from Wicket 1.4 to Wicket 1.5 *(1.5 is the current stable release)*
- Migration from Wicket 1.3 to Wicket 1.4 *(1.4 is the previous stable release)*
- Migration from Wicket 1.2 to Wicket 1.3 *(1.3 is an old stable release)*
- Migration to Wicket 1.2 *(1.2 is an old stable release)*
- List of API changes from 1.1.1 to 1.2 *(Updated to compare with the released 1.2 jar)*
- ~~Migration from 1.2 to Wicket 2.0~~ *(2.0 as described here is discontinued)*

More about Wicket...

Videos, Talks, Screencasts

- "Wicket Quickstart" - comprehensive screencast about how to use wicket-archetype-quickstart by Al Maw
- "Generic Bean Editing with Apache Wicket - Presentation & Code" (short talk by Al Maw)
- Video of an interview with Nick Heudecker on TheServerSide.com
- Slides and presentations

Wicket Press & User Stories

- Articles about Wicket
- Wicket Blogs
- User Stories

Companies Hiring Wicket Developers

- Employers seeking Wicket developers

External Links

- Wicket Spirals teaching method by Dzenan Ridjanovic
- Online Maven Book Get help understanding Maven

Using Wicket

Development Environment

Java

- Setup JDK - Java Development Kit

Build tools

- Setup Maven - Maven (build, reporting and documentation)

IDE

- Setup Eclipse and plugins
- Setup IntelliJ IDEA
- Netbeans IDE

Application server

- Setup Tomcat - servlet engine / application server
- JBoss AS 7 Wicket Quickstart
- Red Hat OpenShift + Wicket - HOWTO Free hosting for Wicket applications on JBoss AS 7

Portal server

- Setup Liferay - enterprise opensource portal server

Database

- Setup HSQLDB - in-memory database during development
- Setup PostgreSQL - enterprise ready opensource database

Development

Wicket User Guide

Learn building web applications with Wicket from scratch reading its 200+ page user guide. The guide gradually introduces you to the various features of the framework with many real-world examples. It covers subjects such as models, behaviours, testing and integration with other projects.

The guide is available as PDF or html file for the following versions:

Wicket User Guide - Wicket 6.x, Wicket 7.x, Wicket 8.x

Framework Documentation

- Documentation Index - An attempt to provide a one-page overview as to where to look for information.
- Windows Guide to Installing Wicket on Eclipse with Maven - How to get a Wicket development environment installed and running.
- New User Guide (*A work-in-progress but a reasonable starting point.*)
- Reference library - A selection of reference guides and how-tos.
- Best Practices and Gotchas
- Wicket FAQs
- Release notes

GUI-Components and Widgets

Here is the reference for the standard Wicket GUI components: Component Reference

You are looking for some additional additional cool Widgets, mainly Javascript/AJAX style?
Then have a look here:

- Wicketstuff - Wicket Stuff provides components that complement the Wicket framework.
- Wicket-Bootstrap - Simple and flexible Wicket Components, HTML, CSS, and Javascript for popular user interface components and interactions.
- wiQuery - wiQuery integrates jQuery and jQuery UI into the Apache Wicket framework.
- visural-wicket - a set of open-source Wicket components, Apache 2.0 licensed.
- Tabs:
    - Article "Advanced Wicket tabs with jQuery"
- Tree:
    - Project "wicket-tree"

Wicket Component Packaging

- Wicket Component JAR Metadata

Portlet Development

- Examples: Wicket Examples as portlets - run the wicket example webapplication as (collection of) portlets

Development Aids

- Serialization Checker

Testing

- Settings for testing
- Choosing the overall testing approach:
    - Selenium - tips for testing Wicket apps with Selenium
    - WicketTester - Mock the browser and the container. Check the states of the components and the models.
    - Wicket Page Test - Use a real browser and a real container. Mock the service objects. Check the HTML DOM elements (supporting AJAX).