the POWER of JAVA™

Creating Good, Simple, Single-Server Web Applications

Jayson Falkner

CTO
Amberjack Software LLC
http://www.jspinsider.com

TS-9542

Goal of This Talk

Learn how to set up and design, implement, and maintain a single-server Java™ technology-powered website that can grow with your needs

Agenda

What you'll learn if you stay

Introduce the speakers

Setting up core software

- Tomcat, Ant, NetBeans™ software, hsqldb

Making your web application

- Model 1$^{1/2}$, DB Access, SSL, Filters

Avoiding growing pains

- Bigger DB, More Developers, Multiple Servers

Conclusion

Links to the stuff I've covered: http://weblogs.java.net/blog/jfalkner/

Introduce the Speakers

Jayson Falkner and Casey Kochmer

- Amberjack Software LLC
- Java Technology Authors
  - Servlets and JavaServer Pages™ specification; the J2EE™ Web Tier
- DevelopMentor Java Technology Instructor
- Several Websites
  - jspinsider.com, jspbook.com, proteomecommons.org
- JavaServer Pages Specification Expert Group
- Long time open-source proponents

Links to the stuff I've covered: http://weblogs.java.net/blog/jfalkner/

Agenda

What you'll learn if you stay

- Introduce the speakers
- **Setting up core software**
    - Tomcat, Ant, NetBeans™ software, hsqldb
- Making your web application
    - Model 1$^{1/2}$, DB Access, SSL, Filters
- Avoiding growing pains
    - Bigger DB, More Developers, Multiple Servers
- Conclusion

Links to the stuff I've covered: http://weblogs.java.net/blog/jfalkner/

What Exactly Is the “Core” Software?

Your basic development environment

- Web Server: HTTP server with JSP™ software/ Servlet support
  - Stick to the Java-based web tier standards
- Build Tool: Automate tedious tasks
  - Compiling, copying, restarting, etc.
- IDE: Make it easy to work with code
  - Highlighting, refactoring, testing
- Database: A suitable data store for your project

Setting Up the Core Software

Your basic development environment

- JDK™ software: java.sun.com
- Web Server: Jakarta Tomcat
  - tomcat.apache.org
- Build Tool: Jakarta Ant
  - ant.apache.org
- IDE: NetBeans IDE
  - netbeans.org
- Database: hsqldb
  - hsqldb.org

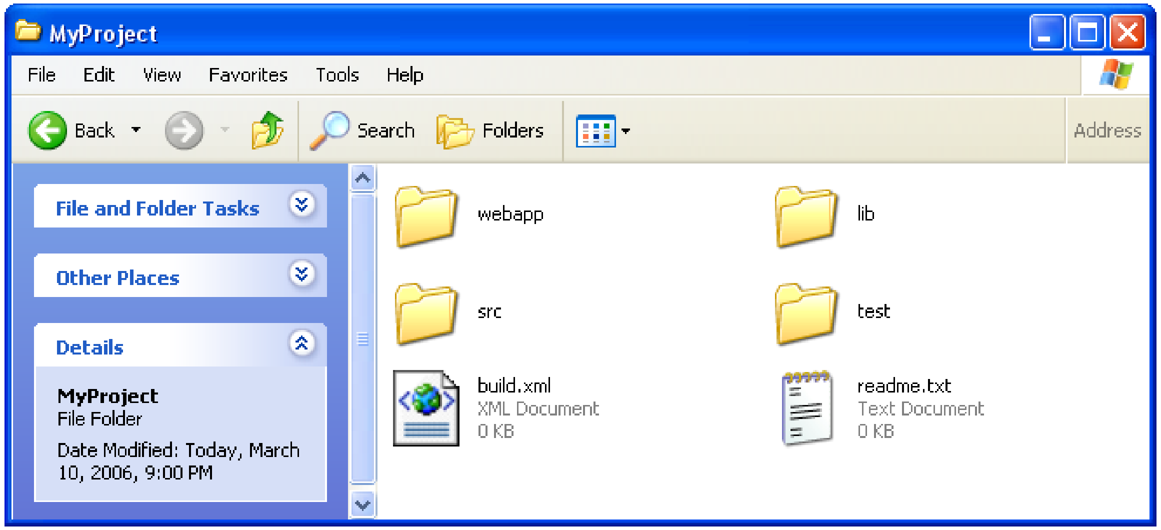
DEMO

Installing the JDK, Tomcat, Ant, and Netbeans

Layout Your Project Files

Empty web app, code, libraries, build file

- Make a directory to organize all your files
- Keep misc files out of the webapp
- Don’t let an IDE take over



MyProject
File Folder
Date Modified: Today, March 10, 2006, 9:00 PM

webapp
lib
src
test
build.xml
XML Document
0 KB
readme.txt
Text Document
0 KB

Links to the stuff I’ve covered: http://weblogs.java.net/blog/jfalkner/

Make an Empty webapp

This is what Tomcat will run

- You only need WEB-INF/web.xml
- A default webpage (index.jsp) helps too
- What else? See the specifications
  - Servlet: http://java.sun.com/products/servlet/
  - JSP software: http://java.sun.com/products/jsp

Links to the stuff I've covered: http://weblogs.java.net/blog/jfalkner/

Empty web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app version="2.4">

  <!-- optional name and description -->
  <display-name>My web application</display-name>
  <description>Webapp for J1 2006.</description>
</web-app>
```

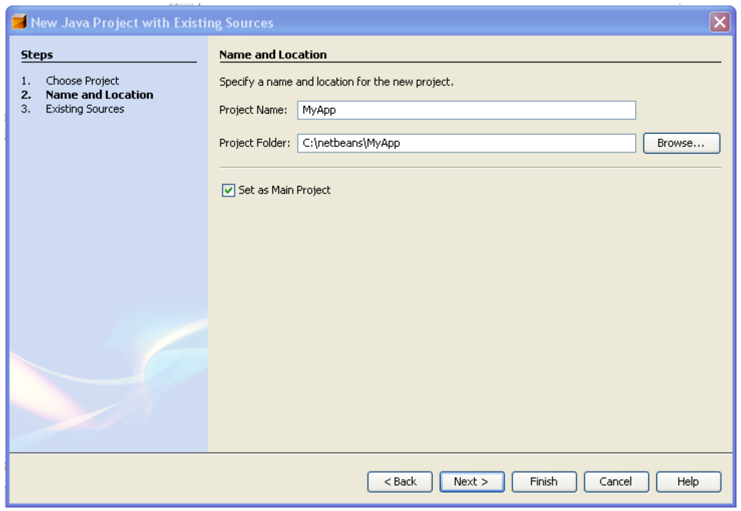
Make an index.jsp or index.html

```
<html>
  <h1>Welcome to my app</h1>
</html>
```

Point Your IDE to the Source-code

Empty web app, code, libraries, build file

- Make a directory to organize all your files
- Keep misc files out of the webapp
- Don't let an IDE take over



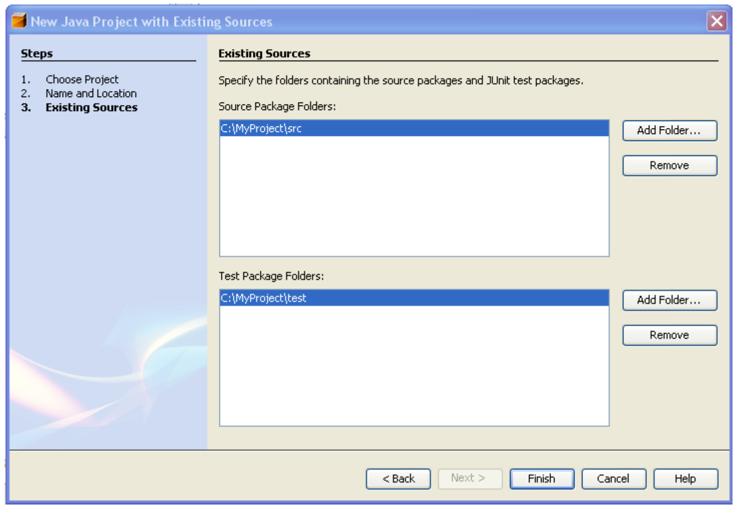
Name and Location

Specify a name and location for the new project.

Project Name: MyApp

Project Folder: C:\netbeans\MyApp Browse...

[✓] Set as Main Project



Existing Sources

Specify the folders containing the source packages and JUnit test packages.

Source Package Folders: C:\MyProject\src

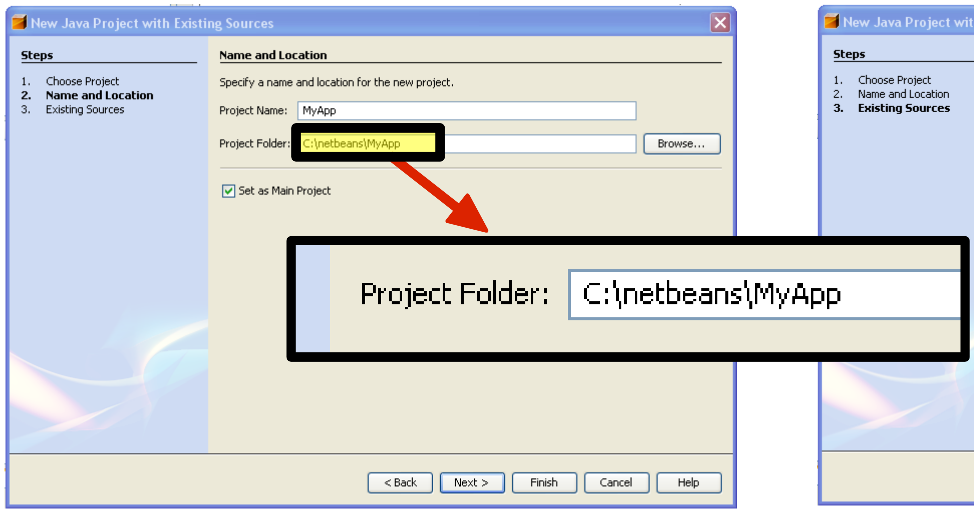
Test Package Folders: C:\MyProject\test

Netbeans Import Existing Code: http://www.netbeans.org/kb/50/import_j2se.html

Point Your IDE to the Source-code

Empty web app, code, libraries, build file

- Make a directory to organize all your files
- Keep misc files out of the webapp
- Don't let an IDE take over


New Java Project with Existing Sources

Steps
1. Choose Project
2. **Name and Location**
3. Existing Sources

Name and Location

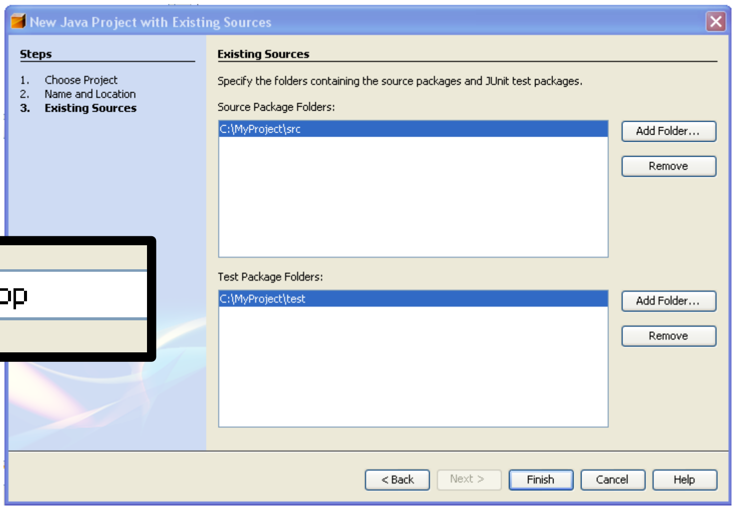
Specify a name and location for the new project.

Project Name: MyApp

Project Folder: C:\netbeans\MyApp

☑ Set as Main Project

Project Folder: C:\netbeans\MyApp


New Java Project with Existing Sources

Steps
1. Choose Project
2. Name and Location
3. **Existing Sources**

Existing Sources

Specify the folders containing the source packages and JUnit test packages.

Source Package Folders:
C:\MyProject\src

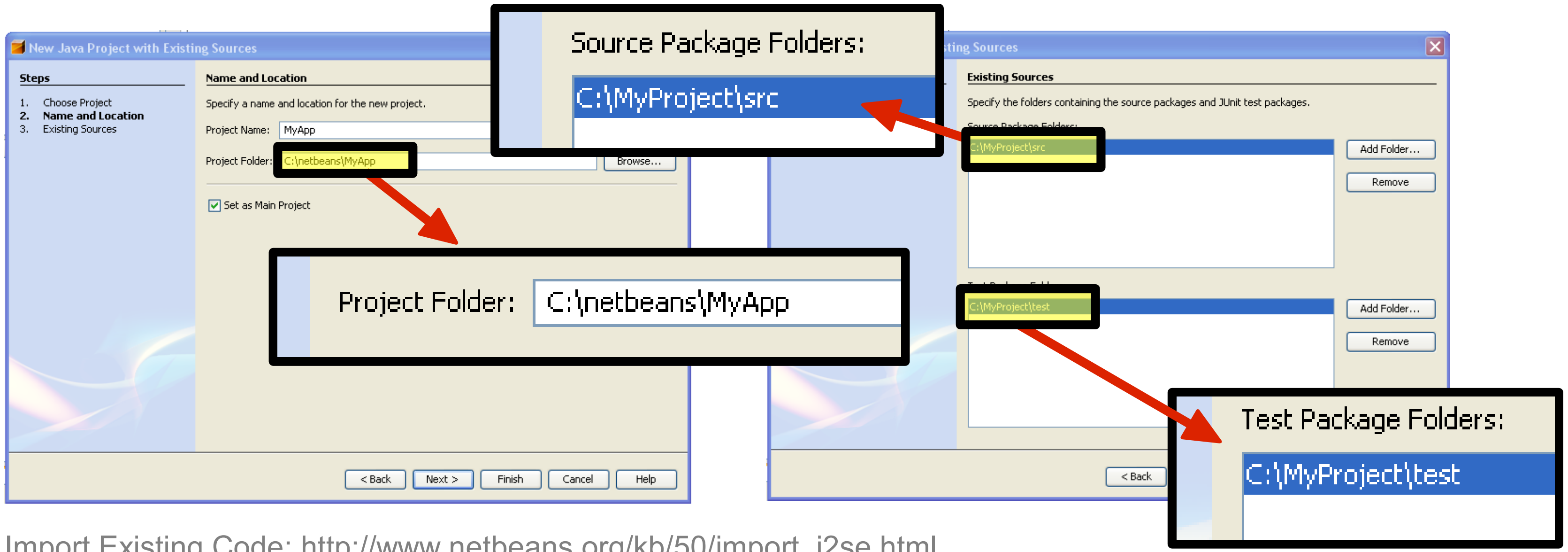
Test Package Folders:
C:\MyProject\test

Netbeans Import Existing Code: http://www.netbeans.org/kb/50/import_j2se.html

Point Your IDE to the Source-code

Empty web app, code, libraries, build file

- Make a directory to organize all your files
- Keep misc files out of the webapp
- Don't let an IDE take over



New Java Project with Existing Sources

Steps
1. Choose Project
2. Name and Location
3. Existing Sources

Name and Location

Specify a name and location for the new project.

Project Name: MyApp

Project Folder: C:\netbeans\MyApp

☑ Set as Main Project

Existing Sources

Specify the folders containing the source packages and JUnit test packages.

Source Package Folders: C:\MyProject\src

Test Package Folders: C:\MyProject\test

Netbeans Import Existing Code: http://www.netbeans.org/kb/50/import_j2se.html

Automating With Ant

Always automate the simple, tedious tasks

- Clean up old code and temporary files
- Compiling code
- Generate documentation
- Creating a WAR for easy deployment
- Deploying/Reloading a WAR with Tomcat
  - Tomcat has Ant tasks
  - http://www.jroller.com/page/srinivas/20050428

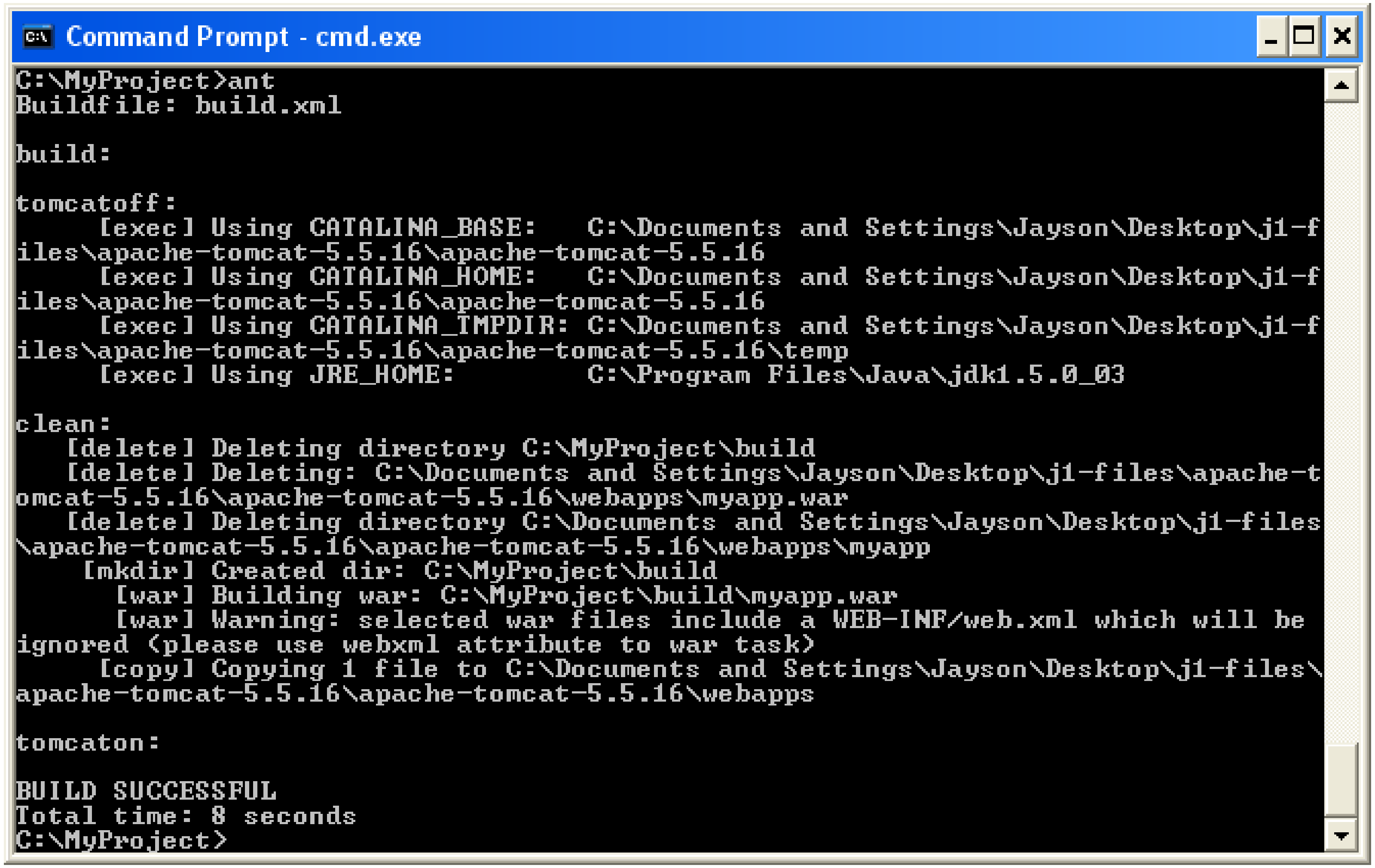
Ant User's Manual: http://ant.apache.org/manual/index.html

Ant’s build.xml

```
<project name="MyApp" default="build" basedir=".">
...
   <target name="build">
      <antcall target="tomcatoff"/>
      <antcall target="clean"/>
      <mkdir dir="${build}" />
      <war destfile="${build}/myapp.war"
           webxml="${webapp}/WEB-INF/web.xml">
            <fileset dir="${webapp}"/>
      </war>
      <copy file="${build}/myapp.war"
            tofile="${appdir}/myapp.war"/>
      <antcall target="tomcaton"/>
   </target>
...
</project>
```

This file is at: http://weblogs.java.net/blog/jfalkner/

Ant at Work



```
C:\MyProject>ant
Buildfile: build.xml

build:

tomcatoff:
     [exec] Using CATALINA_BASE:   C:\Documents and Settings\Jayson\Desktop\j1-f
iles\apache-tomcat-5.5.16\apache-tomcat-5.5.16
     [exec] Using CATALINA_HOME:   C:\Documents and Settings\Jayson\Desktop\j1-f
iles\apache-tomcat-5.5.16\apache-tomcat-5.5.16
     [exec] Using CATALINA_TMPDIR: C:\Documents and Settings\Jayson\Desktop\j1-f
iles\apache-tomcat-5.5.16\apache-tomcat-5.5.16\temp
     [exec] Using JRE_HOME:        C:\Program Files\Java\jdk1.5.0_03

clean:
   [delete] Deleting directory C:\MyProject\build
   [delete] Deleting: C:\Documents and Settings\Jayson\Desktop\j1-files\apache-t
omcat-5.5.16\apache-tomcat-5.5.16\webapps\myapp.war
   [delete] Deleting directory C:\Documents and Settings\Jayson\Desktop\j1-files
\apache-tomcat-5.5.16\apache-tomcat-5.5.16\webapps\myapp
    [mkdir] Created dir: C:\MyProject\build
      [war] Building war: C:\MyProject\build\myapp.war
      [war] Warning: selected war files include a WEB-INF/web.xml which will be
ignored (please use webxml attribute to war task)
     [copy] Copying 1 file to C:\Documents and Settings\Jayson\Desktop\j1-files\
apache-tomcat-5.5.16\apache-tomcat-5.5.16\webapps

tomcaton:

BUILD SUCCESSFUL
Total time: 8 seconds
C:\MyProject>
```

We Now Have a Website

You are ready to code some pages

- Tomcat is serving up content via HTTP
- Development is done in a single directory
- You know what files are where
- Ant automates most everything
- A WAR is produced that is easily portable

You can get this webapp shell at: http://weblogs.java.net/blog/jfalkner/

DEMO

Now we have a website

Agenda

What you'll learn if you stay

- Introduce the speakers
- Setting up core software
  - Tomcat, Ant, NetBeans™ software, hsqldb
- **Making your web application**
  - Model 1$^{1/2}$, DB Access, SSL, Filters
- Avoiding growing pains
  - Bigger DB, More Developers, Multiple Servers
- Conclusion

Links to the stuff I've covered: http://weblogs.java.net/blog/jfalkner/

Making Your Web Application

We’re looking at toast not the bread and butter

- HyperText markup language for content
  - http://www.w3.org/MarkUp/
- Cascading styles sheets for style
  - http://www.w3.org/Style/CSS/
- Java technology hooks for dynamic pages
  - Servlets: Good for bytes and code
  - JavaServer pages software: Good for text and style

You can get this webapp shell at: http://weblogs.java.net/blog/jfalkner/

Where to Put Your Effort

Two critical abstractions you must have

- Abstract code from formatting
  - If you change formatting, it shouldn’t break code
  - Servlets shouldn’t produce HTML
  - JSP-based code shouldn’t mix HTML and Java code
- Abstract your database connections
  - Database changes != broken formatting
  - Database changes minimally impact your Java code
- Hitting moving targets requires flexibility

You can get this webapp shell at: http://weblogs.java.net/blog/jfalkner/

Model 1 ½: Code Then Formatting

```
<% @page ... %>
<%
  String s = request.getParameter("ticker_symbol");
  double value = TickerTool.getValue(s);
  request.setAttribute("s", s);
  request.setAttribute("value", value);
%>
<!-- The line no code shall cross -->
<html>
<h1>Today's Market</h1>
<table>
  <tr><th>Ticker Symbol</th><th>Value</th></tr>
  <tr><td>${s}</td><td>${value}</td></tr>
</table>
</html>
```

This file is at: http://weblogs.java.net/blog/jfalkner/

How Does Model 1 ½ Let Me…

Mix HTML with tag libraries to avoid code

- Java STL does a lot
  - Iteration, conditionals, formatting, i18n, and more
  - http://java.sun.com/products/jsp/jstl/
- JSP Expression Language nicely displays values
  - Part of the JSP 2.0+ specifications
- Abstract complex DHTML/JavaScript™ technology
  - http://ajaxtags.sourceforge.net/
  - http://java.sun.com/javaee/javaserverfaces/index.jsp
- Easier maintenance and collaboration

More links at: http://weblogs.java.net/blog/jfalkner/

Abstracting Database Connections

Don’t embed SQL in random places

- Minimize the impact of your database choice
- Restrict who has raw access to the database
- Data Access Objects (DAO)
    - http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html
- Flexibility to choose and change data persistence
    - http://www.hibernate.org/
    - http://www.springframework.org/
    - http://java.sun.com/products/ejb/
- Easy to unit test your DAO

More links at: http://weblogs.java.net/blog/jfalkner/

DAO: What Not to Do!

```
<% @page ... %>
<%
   Class.forName("com.mysql.jdbc.Driver").newInstance();
   Connection con = DriverManager.getConnection
       ("jdbc:mysql://localhost/weather","a","b");
   Statement st = con.createStatement();
   ResultSet rs = st.executeQuery
      ("SELECT name, age FROM arun");
   ...
%>
<html>
<h1>My Webpage</h1>
  ...
</html>
```

This file is at: http://weblogs.java.net/blog/jfalkner/

DAO: Abstract Database Calls

```
...
public class TickerTool {
  public double getValue(String sym) {
    ...
    Statement st = con.createStatement();
    ResultSet rs = st.executeQuery
      ("SELECT value FROM tickers WHERE sym="+sym);
    ...
  }
}
```

All Other Code Uses

```
TickerTool tt = new TickerTool();
double value = tt.getValue(“SUN”);
```

This file is at: http://weblogs.java.net/blog/jfalkner/

Database Cleanup

Obligated to mention a few other things

- hsqldb is an ideal starting point
  - http://hsqldb.org
- Connection pool for high throughput
  - http://tomcat.apache.org/tomcat-5.0-doc/jndi-datasource-examples-howto.html
- Free heavy duty databases
  - http://mysql.com
  - http://www.postgresql.org/

More links at: http://weblogs.java.net/blog/jfalkner/

SSL for Security

If encryption is needed, it is easy to have

- You must purchase a trusted certificate
  - Chained versus root signed
  - $30 per year: http://www.godaddy.com
- Tomcat supports SSL

  http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html
- Simply add https:// instead of http://
- Servlet specification for web.xml config

More links at: http://weblogs.java.net/blog/jfalkner/

Add servlet Filters for Features

Trivial to install and they can be very handy

- Drop a JAR in WEB-INF/lib, edit web.xml
- Caching and Compression
  - http://www.onjava.com/pub/a/onjava/2003/11/19/filters.html
  - http://www.onjava.com/pub/a/onjava/2004/03/03/filters.html
- User tracking and logging
  - http://opensymphony.com/

More links at: http://weblogs.java.net/blog/jfalkner/

Agenda

What you'll learn if you stay

- Introduce the speakers
- Setting up core software
  - Tomcat, Ant, NetBeans™ software, hsqldb
- Making your web application
  - Model $1^{1/2}$, DB Access, SSL, Filters
- **Avoiding growing pains**
  - Bigger DB, More Developers, Multiple Servers
- Conclusion

Links to the stuff I've covered: http://weblogs.java.net/blog/jfalkner/

Avoiding Common Growing Pains

Having a popular site is both good and problematic

- This is why the critical abstractions are key
  - Clean separation of code and formatting
  - Clean separation of data access and code
- Upgrading the database
- Multiple developer issues
  - Java technology’s portability helps greatly with the rest
    - WARs are easy to deploy
    - All tools are cross platform
- Localization of project files aids collaboration

More links at: http://weblogs.java.net/blog/jfalkner/

Upgrading the Database

SQL and DAO hide the change from your code

- Common growing pains
  - Database/Webserver is too slow
  - More space is needed for the data
  - Multiple servers need to access
- Why it isn’t a problem
  - SQL can largely be reused
  - Only your DAO can break
    - All formatting and most code is fine
  - It is easy to unit test DAO

More links at: http://weblogs.java.net/blog/jfalkner/

Multiple Developer Issues

Sharing code isn't hard if you know where it is

- Common growing pains
  - Everyone needs access to the code
  - Can't standardize on tools
  - What is the deliverable?
- Why it isn't a problem
  - Trivial to share the simple project directory
  - All tools are already cross platform
    - IDE isn't tied to the project
  - A working WAR is the deliverable

More links at: http://weblogs.java.net/blog/jfalkner/

Multiple Developer Issues

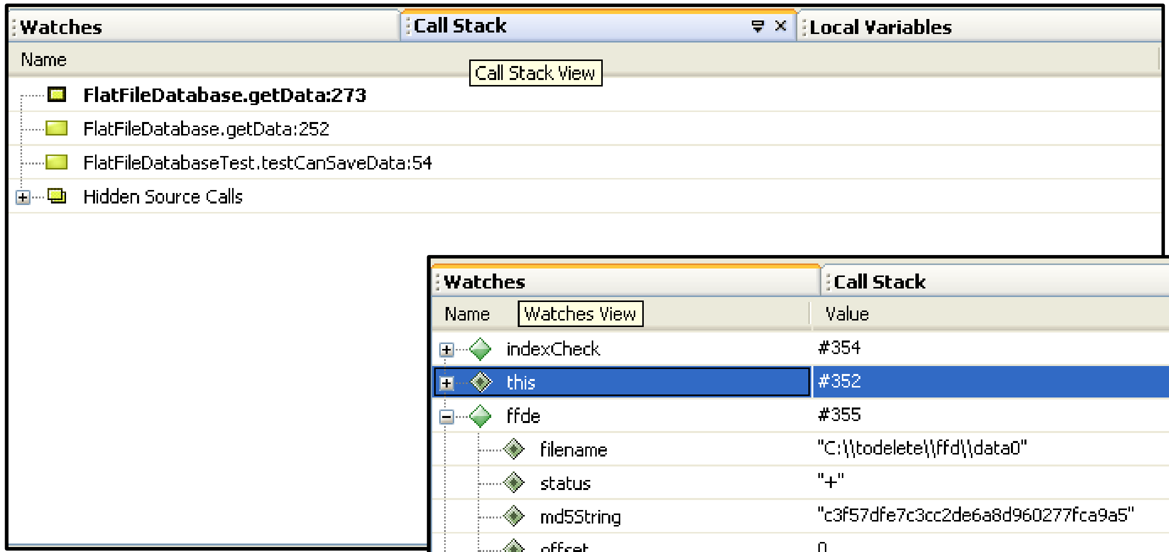
How do you code together?

- Common growing pains
  - How do I know that the code works?
  - Quickly analyzing other's code
- Why it isn't a problem
  - Unit tests check functionality
    http://junit.netbeans.org/
  - Java technology has great debuggers and profilers
    http://debugger.netbeans.org
    http://profiler.netbeans.org

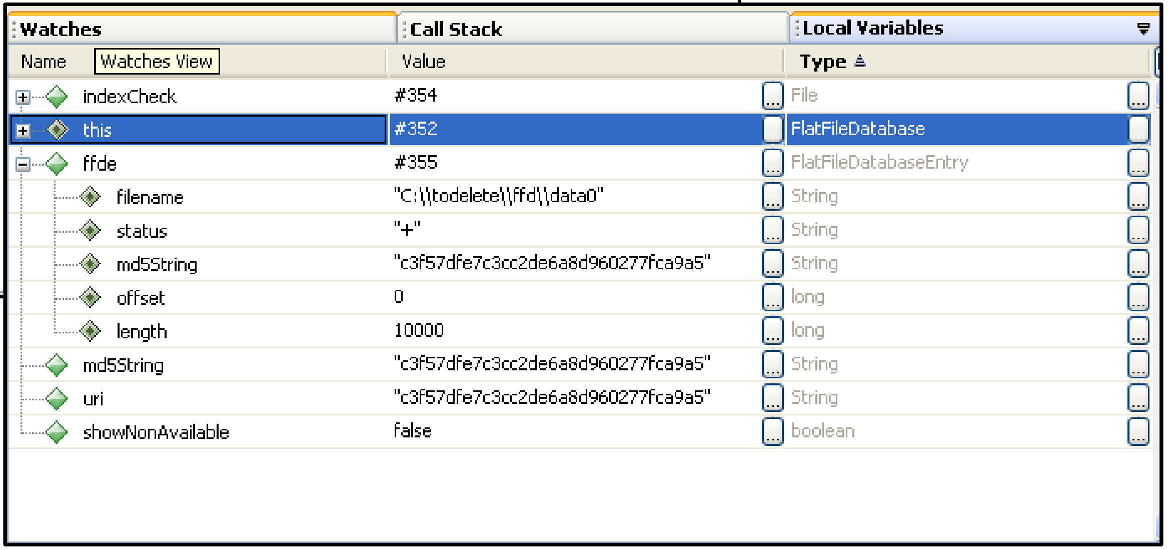
More links at: http://weblogs.java.net/blog/jfalkner/

NetBeans Software Debugging

Easily seeing what goes on in the Java VM



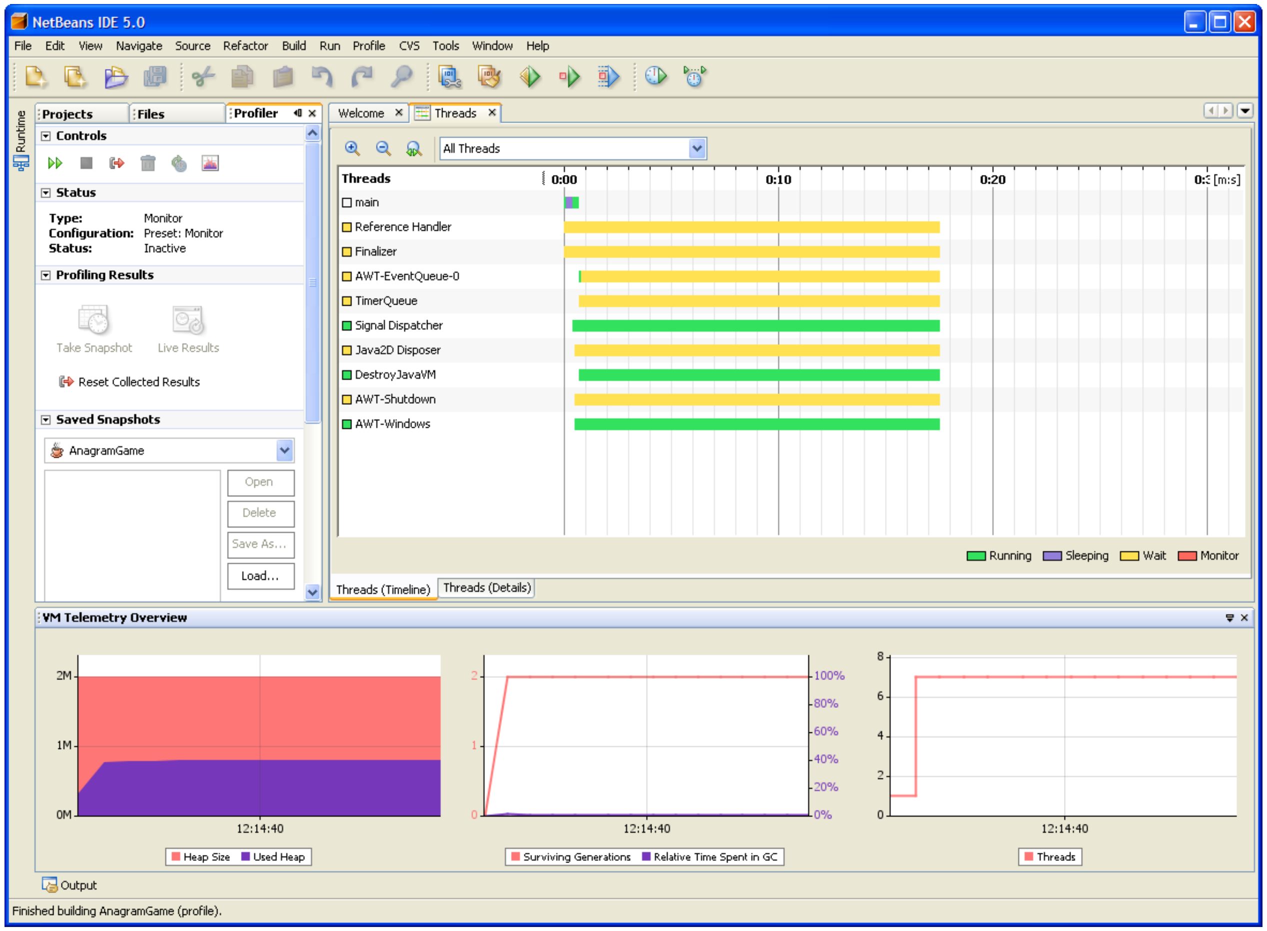
Name
FlatFileDatabase.getData:273
FlatFileDatabase.getData:252
FlatFileDatabaseTest.testCanSaveData:54
Hidden Source Calls



Name	Value	Type
indexCheck	#354	File
this	#352	FlatFileDatabase
ffde	#355	FlatFileDatabaseEntry
filename	"C:\\todelete\\ffd\\data0"	String
status	"+"	String
md5String	"c3f57dfe7c3cc2de6a8d960277fca9a5"	String
offset	0	long
length	10000	long
md5String	"c3f57dfe7c3cc2de6a8d960277fca9a5"	String
uri	"c3f57dfe7c3cc2de6a8d960277fca9a5"	String
showNonAvailable	false	boolean

NetBeans Software Profiling

Easily seeing what goes on in the Java VM



Type: Monitor
Configuration: Preset: Monitor
Status: Inactive

AnagramGame

All Threads

Threads	0:00	0:10	0:20	0:[m:s]
main				
Reference Handler				
Finalizer				
AWT-EventQueue-0				
TimerQueue				
Signal Dispatcher				
Java2D Disposer				
DestroyJavaVM				
AWT-Shutdown				
AWT-Windows				

🟩 Running 🟪 Sleeping 🟨 Wait 🟥 Monitor

Finished building AnagramGame (profile).

Multiple Servers

Growing beyond a single content server

- Quite possibly the trickiest problem
- Simple case, deploy the WAR lots of places
  - DNS can round robin requests
  - Session management issues
- Tomcat clustering

  http://tomcat.apache.org/tomcat-5.5-doc/cluster-howto.html
- The rest of J2EE™ platform

  http://java.sun.com/j2ee

Links to the stuff I've covered: http://weblogs.java.net/blog/jfalkner/

Agenda
What you'll learn if you stay

- Introduce the speakers
- Setting up core software
  - Tomcat, Ant, NetBeans™ software, hsqldb
- Making your web application
  - Model $1^{1/2}$, DB Access, SSL, Filters
- Avoiding growing pains
  - Bigger DB, More Developers, Multiple Servers
- **Conclusion**

Links to the stuff I've covered: http://weblogs.java.net/blog/jfalkner/

Conclusion

Take these points home

- The software is free, pay for people
- Use those critical abstractions
  - Formatting apart from code
  - Code apart from database
- You want to be able to grow
  - Bringing on more people
  - Upgrading your database
  - Multiple servers

Links to the stuff I've covered: http://weblogs.java.net/blog/jfalkner/

Q&A

Jayson Falkner and Casey Kochmer

Creating Good, Simple, Single-Server Web Applications

Jayson Falkner

CTO
Amberjack Software LLC
http://www.jspinsider.com

TS-9542