Turbine: Building Model 2+1 Web Applications

http://jakarta.apache.org/turbine

Martin Pöschl - mpoeschl@apache.org
Outline

- What is Model 1?
- What is MVC?
- What is Model 2?
- What is Turbine (Model 2+1)?
- How do I build applications with Turbine? (TDK)
What is Model 1?

- Model 1 is the current method of creating sites using JSP/Servlets. A linear and non-expandable approach.

- Flow works like this:
  - Request comes in
  - JSP page executes Beans to do processing
  - Result is returned
Picture of Model 1
What is MVC?

- Model-View-Controller
- Abstraction of each layer of development
  - Model - Application Logic
  - View - Presentation Logic
  - Controller - Business Logic
Model - Core of the Application

- Application Logic
  - database accesses
  - data crunching/munging
  - Separation of UI from application

- Not dependent on input or output formats
View - Presentation Layer

- Responsible for displaying the combined work of the Controller and Model
- UI portion of the page
- Formats data to be presentable to the end user
Controller - Input Mechanism

- Listens to what the User requests
- Supplies and communicates information to the Model and the View
A Picture of MVC

Web Application Domain

Copyright Lutris Technologies
What is Model 2?

- It is a server side implementation of MVC using Java Server Pages (JSP)
- Flow works like this:
  - Request comes in
  - Servlet (Controller) is executed to build a Context (JavaBean)
  - Context is passed to JSP template to provide the View
  - Result is returned
- Listens to what the User requests
- Supplies and communicates information to the Model and the View
Picture of Model 2
Problems with Model 2

- Does not address the issue of providing a consistent View across the application.
- Does not mention a framework for reusable code.
- Does not address issue of tying the Controller to the View.
- Does not address issue of receiving input from the View.
- Needs to be more web application friendly.
A JSP solution to Model 2 Problems

- For those of you who refuse to use anything other than JSP, the Struts project is the right way to go for now.

- Http://jakarta.apache.org/struts/
What is Turbine?

- Turbine is a servlet based framework that allows experienced Java developers to quickly build secure web applications.

- A web based application is an application where users use their favorite web browser in order to access secure business logic.
Problems Turbine Solves

- Extensible base framework for development
- Re-usable web application code
- Module architecture
- Turbine is trying to be the wheel. Anything else is re-inventing that wheel.
Turbine Features 1

- Integration with template systems: **Velocity**, WebMacro, Cocoon, FreeMarker, JSP, ECS
- Utility code for working with Velocity/WM, such as a SelectorBox class for building `<select>` boxes
- Single Entry Point Servlet model for optimal security and control
- Singleton based Database Connection Pool (JDBC) with built in support for all of the major databases
- Parameter Parsing for GET/POST/PATH_INFO
- Event based Action handling!
Turbine Features 2

- Strict MVC guidelines implemented through many interfaces and abstract classes as well as through the template systems.
- Integration with 4 Object-Relational tools: DatabaseMap/Peers, Village, Town, Castor
- IDBroker that abstracts the autoinsert/sequence usage from the database
- DatabaseMap generation tool that will read a schema and generate the OR Java classes for you
- Failsafe Job based scheduler system (ie: a Java based cron)
Turbine Features 3

- GlobalCache (Singleton based system for caching data across servlets and requests)
- DateSelector (utility for building the HTML for popup date menus)
- File Upload API
- Generic Services API for creating Singletons
- Castor Service Integration
- XML-RPC Service Integration
- Localization Services API
- JNDI Services API
Turbine Features 4

- Visitor/Member API for managing users
- Temporary and Permanent storage/management for users session data
- ACL (Access Control List) based security system that uses Roles and Permissions (and the database schemas)
- BrowserDetector class for determining which browser someone is using
- GenerateUniqueId class for getting a unique number (ie: for sessions)
- Logging (Log4j)
Turbine Features 5

- Property file reader tools and a TurbineResources service to make it easy to retrieve properties
- Integration with JavaMail to make sending email painless
- Integration with JavaMail and WebMacro/Velocity to allow you to send processed templates as email!
- Built using Ant
- Works cleanly with Servlet API 2.0 and higher
- Works with JDK 1.1.x and higher - 100% Pure Java
Turbine’s Community

- Turbine has a mailing list with over 430 people on it
- There are over 35 people contributing code and documentation
- Used by people all over the world (many foreign contributors)
- Used as the basis for other large OSS projects like Jetspeed
Turbine’s Size

- Turbine now has almost 300 classes (and 5 slides of features!)
- Not all parts of Turbine have to be used
- Pick the parts you want and ignore the rest…until you find you need it. :-)

07.05.01 Martin Pöschl <mpoeschl@apache.org>
What is Model 2+1?

- Solves the problems of Model 2.
- Is not Model 3, but is instead Model 2 plus feature additions.
How is Turbine Model 2+1?

- Adds lots of reusable code
- Interfaces (and implementations) for Security, User Management and Services
- Turbine Modules for tying MVC together
  - Actions and ActionEvents
- Designed to be used as a webapp framework
Turbine and MVC

- Turbine blurs the MVC model a bit
- Turbine provides the overall container for building MVC applications
- View is managed by one of the template solutions, but controlled by Turbine
Picture of Turbine Modules
Picture Turbine Execution
Template Directory Structure

- templates/
  - screens/
    - MyScreen.vm
  - actions/
    - MyAction.vm
  - navigations/
    - TopNav.vm and BottomNav.vm
  - layouts/
    - Default.vm
Java Directory Structure

- modules/
  - screens/
    - HelloWorld.java
  - actions/
    - MyAction.java
  - navigations/
    - No file needed right now.
  - layouts/
    - No file needed right now.
public class HelloWorld extends WebMacroSiteScreen
{
    public void doBuildTemplate( RunData data, Context context )
        throws Exception
    {
        // the context object has already been setup for you!
        context.put ("hello", "this is a test...");
    }
}
Template: HelloWorld.wm

$page.setTitle("Welcome to jugat")
This is an example Velocity template!
<p>
Here is what was put into the context:
<p>
<b>$hello</b>
Template: default.vm

$page.setStyleSheet("/myapp/style.css")
$page.setBgColor("#ffffff")
$navigation.setTemplate("/TopNav.vm")

<table width="100%">
    <tr>
        <td>$screen_placeholder</td>
    </tr>
    <tr>
        <td>$navigation.setTemplate("/BottomNav.vm")</td>
    </tr>
</table>
Template: TopNav.vm

This is the Top Navigation
<hr>
Template: BottomNav.vm

This is the Bottom Navigation
Example Output

```
This is the Top Navigation

This is an example WebMacro template!

Here is what was put into the context

this is a test...

This is the Bottom Navigation
```

Welcome to Comdex 2000

[Image of a computer window with the text shown in the natural text above]
<form action="$link.setPage("Next.vm")" method=post>
<input type="hidden" name="action" value="MyAction">
<input type="text" name="f1" value="">
<input type="submit" name="eventSubmit_doSubmit" value="Hit me!">
</form>
public class MyAction extends WebMacroSiteAction
{
    public void doSubmit (RunData data, Context context)
    throws Exception
    {
        String text = data.getParameters().getString("f1", "blank");
        context.put("username", text);
        data.setMessage("Done!");
    }

    public void doPerform(RunData data, Context context)
    throws Exception
    {
        data.setMessage("Button not found!");
    }
}
Peers

- One to one mapping with a Database Table
- DatabaseMap’s define the database
  – One Map peer database
- Created by MapBuilders
- Peer Objects control access to the database
- Data Objects - Represent one row in a table
Peers (continued)

- Criteria - abstraction of what it takes to build a select statement
  - Handle 90% of all select cases
  - You can still use straight SQL for more complex cases

- IDBroker - creates unique keys for doing inserts
  - database independent
  - used by the Peer objects for inserts
Criteria crit = new Criteria();
crit.add (ItemPeer.CATEGORY_ID,2);
Vector v = ItemPeer.doSelect (crit);
for (Enumeration e = v.elements(); e.hasMoreElements();)
{
    Item item = (Item) e.nextElement();
    int id = item.getId();
}
Peers (inserts)

Item itm = new Item();
itm.setName("New Item");
itm.setPrice(100);
itm.setCategoryId(1);
Object o = ItemPeer.doInsert(itm);
Peers (inserts2)

Item itm = new Item();

data.getParameters().setProperties(itm);

Object o = ItemPeer.doInsert (itm);
Criteria crit = new Criteria();

crit.add(CategoryPeer.NAME,"New Category");

Object o = CategoryPeer.doInsert (crit);
Peers (updates)

Criteria crit = new Criteria();
crit.add (ItemPeer.ITEM_ID,3);
crit.add (ItemPeer.NAME,"My Item");
ItemPeer.doUpdate(crit);
Peers (deletes)

Criteria crit = new Criteria();
crit.add (ItemPeer.ITEM_ID,3);
ItemPeer.doDelete (crit);
Turbine Developers Kit (TDK)

- Everything you need to start developing webapps with Turbine out of the box
- Everything is pre-configured. Just start Tomcat and go.
- Includes:
  - Tomcat 4.0
  - Turbine
  - Lots of .jars
  - Documentation
TDK Benefits

- Easily create a webapp with the directory structure and sample pages already in place
- All the external .jars are already there
- Tomcat WebServer/Servlet engine
- Easily upgraded
- 100% Free
URL’s

- http://jakarta.apache.org/turbine/
- http://jakarta.apache.org/turbine/tdk/
- http://jakarta.apache.org/velocity/