COSC 617 – Advanced Web Development

Siddharth Kaza
Towson University, Computer and Information Sciences

Outline

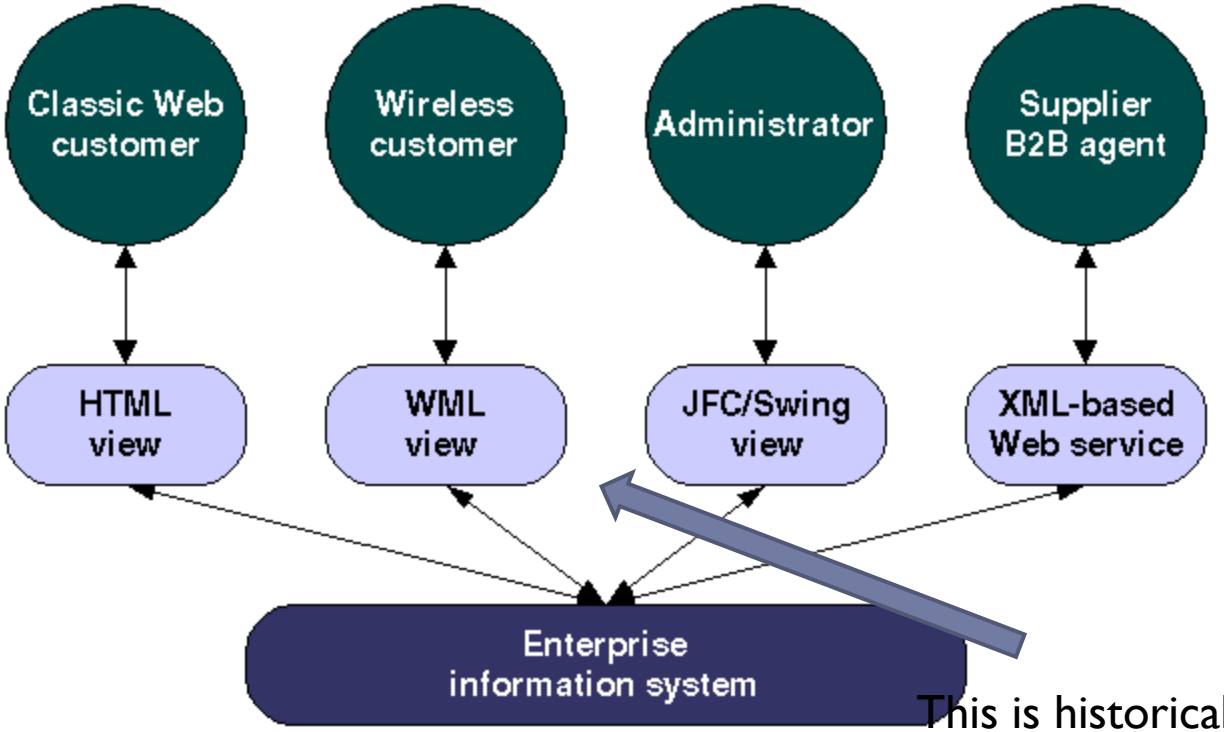
- Model-view-controller
- Rails Demo
  - Introduction to our VM, Eclipse Environment, and Rails environment
- MVC in other languages/environments
- Discussion on project topics and teams
  - Handout previous projects
  - Paper presentations
- Lab 1 – First Rails project

Model-View-Controller: Motivation

- Object oriented design principals
  - One class per "concept"
  - Each class does one thing well
    - Minimize coupling
    - Separation between classes is clear
  - Support re-use, modification, etc.
- Make classes into components.
- MVC coined in 1979 for use in Smalltalk

MVC on the Web: Motivation

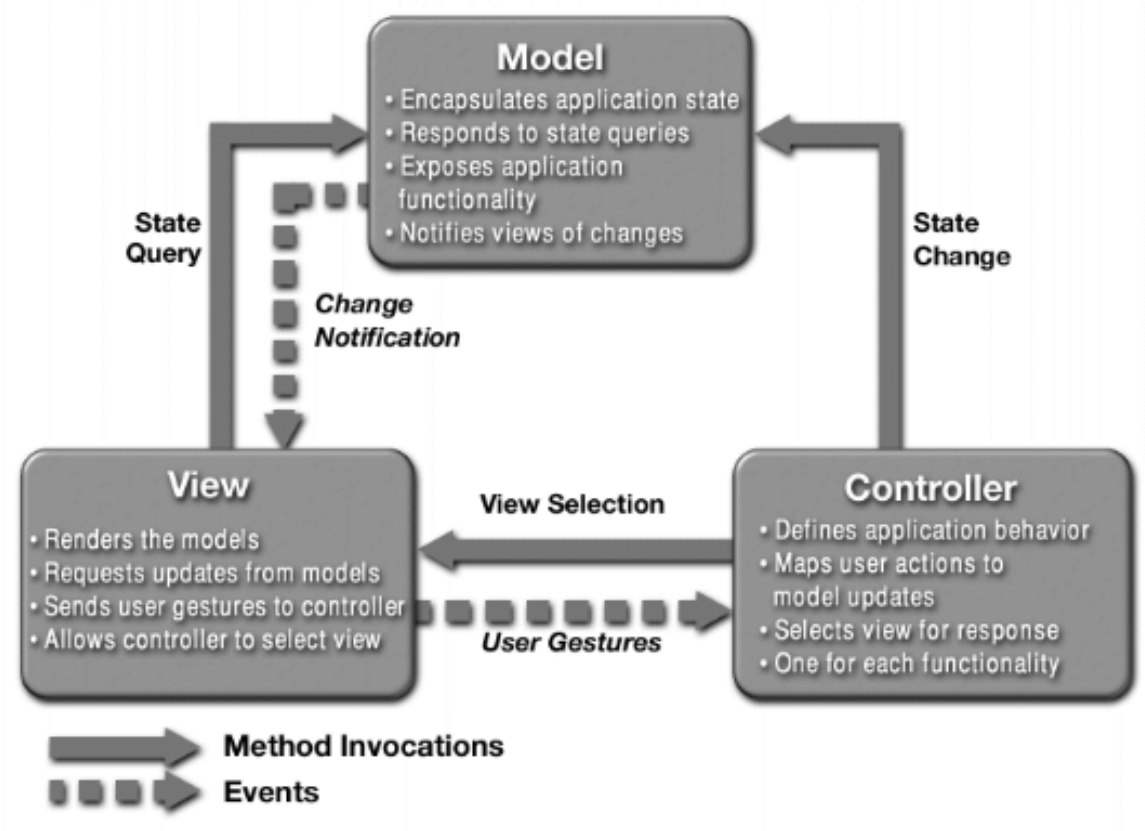
- Enterprises need to support multiple "views" of the same data



This is historical now, but you still need a mobile view.

- From: http://java.sun.com/blueprints/patterns/MVC-detailed.html

Sun's definition of MVC for Web



- From: http://java.sun.com/blueprints/patterns/MVC-detailed.html

Sun's definition of MVC for Web (cont.)

- Model: application data and rules.
  - "business logic"
- View: renders model. Gets data from model and specifies and presents it.
  - Views responsibility to show the most current data
- Controller: translates interactions with user into actions to be performed by the model.

- From: http://java.sun.com/blueprints/patterns/MVC-detailed.html

MVC and Rails

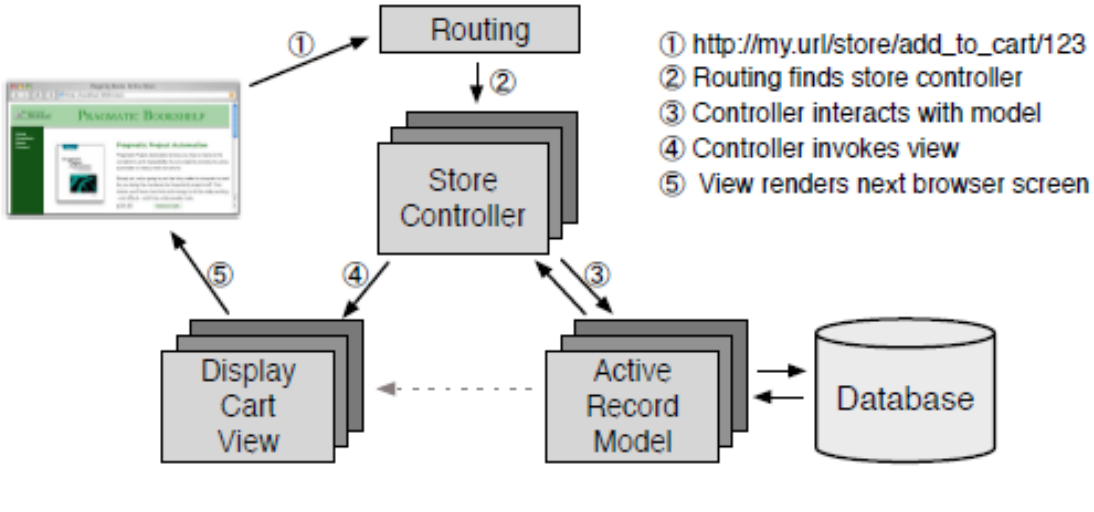


Figure 2.2: Rails and MVC

- So why use MVC?
  - Easier to develop, easier to maintain
  - Makes intuitive sense
- Why not?
  - 3x classes
  - Separation may not be clean

What's a model? What's a controller?

- Model – some object, or class of objects in a system or domain
  - **Nouns**
  - Maintains state of the application
  - Data + business rules

- Controller – collection of methods/actions that operate on a model
  - **Verbs**
  - Similar to methods in OO
  - Each model class has methods for setting and querying state
  - Verbs in controllers are things that web site might do to a model

Two types of models

- Persistent
  - Those things that we want to store for the long term in a database
  - User accounts
- Transitory
  - Information relating to the current session
- So don't think Models = DB Tables, it more than that
  - Classes
  - Arrays
  - Hashes

MVC in Rails

- Lets do a small demo of rails
  - while we do this, think of MVC architecture and how it is manifesting itself

- A list of instructors and their courses.

Demo – Our VM

- The VM is Ubuntu 14
- Passwords on the blackboard in the VM document
- In the settings of VMWare Player/Workstation please assign your VM at least 3GB+
- Ruby 2.0, Rails 4.2, Rubymine 7
- Rails is a gem for Ruby along with other gems. So you install Rails after you install Ruby.
- It uses the Ruby Version Manager called rbenv
  - You may have to make sure that version manager is pointing to right version of ruby.
  - To check versions, use 'rails -v' and 'ruby -v'

- If you want to transfer files between your host machine and the VM, you may have to use a USB drive

Demo - Bundler

- One of the first problems you may face is certain dependencies missing
- Dependencies are gems that your application depends on to function
  - A gem is a packaged Ruby application or library. It has a name (e.g. **rails**) and a version (e.g. **3.2.8**).
- Very important to specify exactly which gems and what versions your application depends on
  - so it can function on servers with different S/W configurations
- The *bundler* takes care of this and installs gems that are missing.
  - See the Gemfile in your project folder
  - Add the lines to the file (if needed, mostly in windows)

    gem 'therubyracer'

    gem 'execjs'
    - Or uncomment it if its already there

Demo – actual app

- I will mostly go along with the book, but the application is slightly different.

- Steps
  - Generate a controller and views
    - rails generate controller main index goodbye time
  - Notice the directory structure and the structure of the controller file
    - See the link methods to views and how that functions
    - See routes.rb
  - See the URL structure
  - Add Course model
  - Link the main and the goodbye page together
  - To start the server
    - rails server
    - rake db:migrate

Demo

- Notice the Don't Repeat Yourself (DRY) and convention over configuration philosophies
  - Look at the size of the code vs. a Java-based application
- Initial impressions?
  - No complex configurations
  - Quick delivery of working software
  - Easy to test with the framework already in place
  - Code in many places, non-trivial to understand the first time
  - Ruby – the language is different, sometimes hard to understand

MVC in other languages/environments

- Various frameworks in Java – Tapestry, Struts, Spring,, Stripes
  - Nice introduction - https://struts.apache.org/birdseye.html
  - Hello World - https://struts.apache.org/docs/hello-world-using-struts-2.html
- .Net – ASP .Net MVC
- Microsoft Web Matrix

MVC in Java : Model

```
package com.jbossatwork;

public class CourseBean {
        private String instructor;
        private String name;
        private int number;

        public CourseBean(String instructor, String name, int number) {
                this.instructor = instructor;
                this.name = name;
                this.number = number;
        }

        public String getInstructor() { return instructor; }
        public void setInstructor(String instructor)
                { this.instructor = instructor; }
...
}
```

- Code example modified from jbossatwork.com "JBoss at Work: A Practical Guide, by Tom Marrs and Scott Davis. Copyright 2005 O'Reilly Media, Inc., 0-596-00734-5."

MVC in Java : Controller

```
If (VIEW_COURSE_LIST_ACTION.equals(actionName)) {
        List courseList = new ArrayList();
        courseList.add(new CourseBean(“Kaza", "Intro", 236));
        courseList.add(new CourseBean(“Kaza",”Web”, 617));
        courseList.add(new CourseBean("Dierbach", "Intro", 175));
        request.setAttribute("courseList", courseList);
        destinationPage = "/courseList.jsp";
}
else {
        String errorMessage = "[" + actionName + "] is not a valid action.";
        request.setAttribute(ERROR_KEY, errorMessage);
}

// Redirect to destination page.
RequestDispatcher dispatcher =
        getServletContext().getRequestDispatcher(destinationPage);
dispatcher.forward(request, response);
```

- Code example modified from jbossatwork.com "JBoss at Work: A Practical Guide, by Tom Marrs and Scott Davis. Copyright 2005 O'Reilly Media, Inc., 0-596-00734-5."

MVC in Java : View

```
<body>
<ul>
<c:forEach items='${courseList}' var='course'>
<li> ${course.instructor}, ${course.name} ${course.number}</li>
</c:forEach>
</ul>

</body>
</html>
```

- Code example modified from jbossatwork.com "JBoss at Work: A Practical Guide, by Tom Marrs and Scott Davis. Copyright 2005 O'Reilly Media, Inc., 0-596-00734-5."

Ruby vs. Java versions

- Is it just a matter of dividing the code into pieces? What more does Rails give us?

- Model – similar
  - Ruby more concise.
  - Tools might generate Java setter/getter pairs for you

- View – almost identical

- Controller?
  - Boilerplate handled by Ruby with appropriate defaults
    - Error message
    - Message-handling
  - Dispatch
  - Most defaults can be over-ridden as needed.

Projects

- What are you thinking about doing?

- Examples: think about common sites that you see and plan to develop a scaled down version:
  - Ebay
  - Linkedin
  - Wikipedia
  - Twitter
  - …

Ruby

- Resources
  - Look at Lab 1
  - http://www.ruby-doc.org
  - Dave Thomas, et al. Programming Ruby, The Pragmatic Bookshelf.
    - Source of many of the examples in these slides
- Object oriented (no primitives like Java)
- Interpreted
- Code blocks "….end" delimited