Chapter 1

Spring Web MVC Introduction

Objectives:

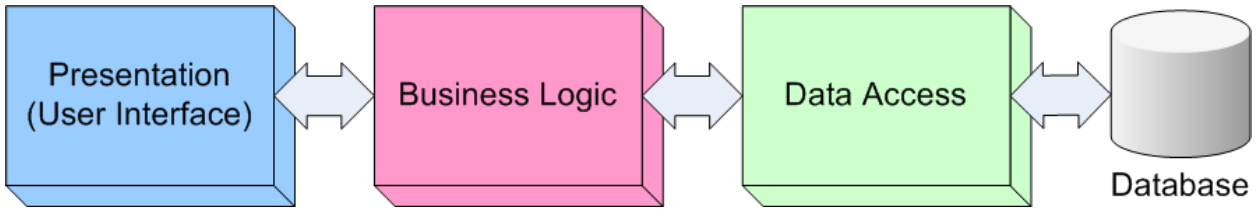
- **Understand the complexity of Web application development and the need for a framework.**
- **Learn what an MVC framework is and how it helps in the development of applications.**
- **Learn about the many Java EE MVC frameworks.**
- **Understand the impact of Spring 3 to the Web MVC framework.**

Chapter Overview

If the acronym MVC means nothing to you, this introductory chapter attempts to clarify what MVC is all about and why organizations often adopt the MVC pattern when developing Web applications. More precisely, you examine why organizations often adopt an MVC framework when developing Web applications.

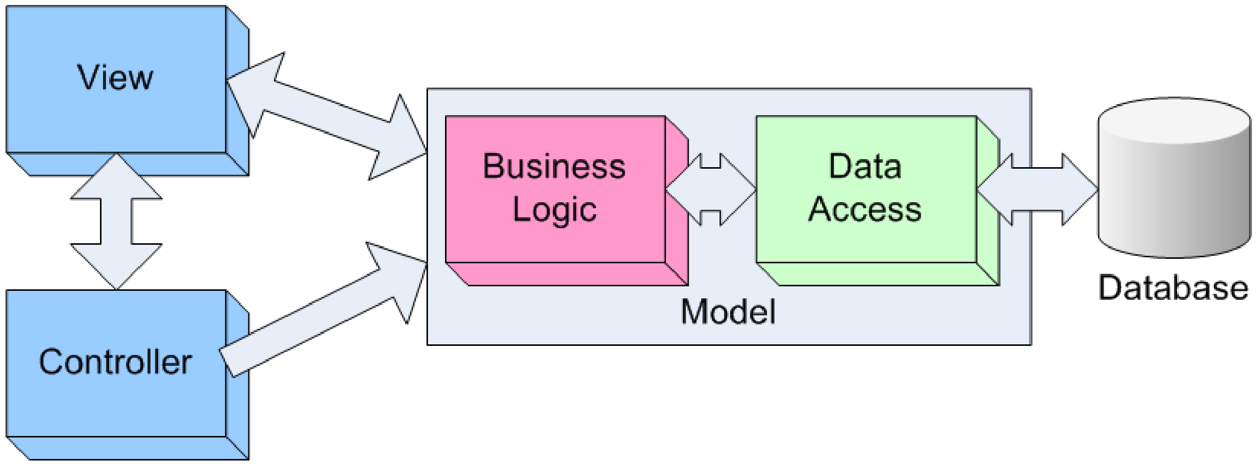
Java Web Applications

- **Building Web applications can be a difficult task.**
  - All information exchanged in HTML over HTTP is in the form of Strings.
  - This form of data exchange requires a lot of data conversion and is subject to many data validation issues.
  - The server side is usually required to have extensive knowledge of the client side UI in order to extract data from the key/value pairs in the request object.
  - HTTP is generally stateless, so tracking a user's actions and information over many requests and responses requires state management activities – usually via session.
  - Working with HTML can be tedious, error prone, and difficult to create graphically pleasing user interfaces.
  - Web site navigation and workflow can be difficult to orchestrate and even harder to change/adapt to new business needs.
- **Most software engineers soon learn that it is important to divide any application (Web or otherwise) into separate layers.**
  - Specifically, most significant applications are divided into presentation (or user interface), business or domain logic, and data access layers.



  - When implemented correctly, applications divided in this manner allow each layer to be significantly modified, or even replaced, without affecting the others.
  - For example, the data access layer can be changed to get data from a different source without causing the domain logic or UI to change.
  - Loose coupling between the layers provides for flexibility and ease of maintenance.

- **The Model View Controller (MVC) pattern carries this idea further.**
  - In MVC applications, the presentation layer is further divided into view and controller layers.
  - In an MVC application, the business or domain logic and data access layers are seen collectively as the model.



  - The model encapsulates the raw data and business logic that operate on that data. The model should also notify observes when it has changed.
  - The controller responds to events, typically user actions, and instructs the model and the view to perform actions based on the events.
  - The controller may invoke changes on the model.
  - The view renders information supplied by the model in a form suitable for user interaction. Multiple views may exist for different displays of the same model.
  - The view may also serve to capture events to send to the controller.

- **In the Java EE arena, several frameworks provide assistance to developers in implementing MVC Web applications.**
  - While servlets, JSPs, JSP tag libraries, and JavaBeans provide the raw ingredients for developing MVC Web applications, there is still a lot of work to accomplish.
  - A framework provides a lot of the application infrastructure, or "plumbing", allowing developers to concentrate on business code, not infrastructure code.
  - For example, a framework often provides infrastructure code for handling data validation.
  - A framework also helps to formalize the use of the Java Web technologies making the application easier to develop and maintain.
  - For instance, frameworks often dictate how servlets, in the role of controller, react to events and call on the next view to be displayed.
  - Most frameworks also provide common out-of-the-box services and functionality that Web developers often need.
  - These include, but are not limited to, data conversion, internationalization/localization, populating form fields, etc.
- **The table below lists some of the more popular MVC Web frameworks in Java.**

MVC Frameworks
Spring MVC
Struts
JavaServer Faces
Cocoon
Tapestry

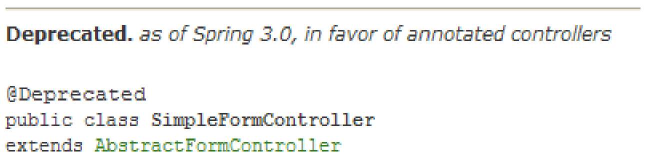
- **In this class, you explore the Spring MVC Framework.**
    - The Spring Web MVC framework is based on servlets, JSPs, tag libraries, and Java Beans – just like many of its competitors.
    - As you might imagine, Spring Web MVC also relies on the power of the Spring container and other Spring APIs.
- **In the Spring MVC world, there are many components to orchestrate each request/response to the user.**
    - Each component has a specific task.
    - While this might seem to add complexity, it allows for maximum flexibility and customization.
    - Each component can be configured or re-engineered for specific application needs.
- **As you have probably come to expect with Spring, it can integrate with "best of breed" solutions in all aspects of application development.**
    - In the Web MVC world, Spring holds to this premise.
    - While not covered in this class, the Spring Framework integrates to several popular MVC frameworks like Struts and JSF.
    - In addition, Spring MVC can be used with several popular templating and MVC ancillary technologies like Velocity, FreeMarker, and Tiles.
    - A Spring MVC Portlet Framework is also provided for those that need to build Web portals.
    - The Spring MVC Portlet Framework mimics the architecture and style of the Spring MVC Framework.

Spring 3 and Web MVC

- **If you have used the Spring Web MVC framework in past (prior to Spring 3), recognize that Spring 3 brought major changes to the Web MVC framework.**
  - In many ways, the Spring 3 Web MVC framework is simpler than the old Web MVC framework.
  - Rod Johnson, the creator of the Spring Framework, feels the new framework is much better than the old version.

```
We view the Annotation based Spring MVC usage model that we introduced in
Spring 2.5 is being a far superior replacement.[1]
```

  - Unfortunately, the new framework is quite different and requires, in most cases, a rewrite of applications using the old framework.
  - In fact, many of the Spring 1.x and 2.x Web MVC components are deprecated in Spring 3.



Deprecated. *as of Spring 3.0, in favor of annotated controllers*

```
@Deprecated
public class SimpleFormController
extends AbstractFormController
```

- **The new Spring 3 Web MVC framework is based on building Plain Old (or Ordinary) Java Objects to create Web applications.**
  - The old framework required implementing or extending Web MVC specific interfaces or classes – violating the loose coupling philosophy of Spring.
  - The new Spring 3 Web MVC framework typically requires fewer components and less coding - deferring more of the work to the framework.

[1] http://www.infoq.com/interviews/Spring-3.0-Rod-Johnson

- **The new Spring 3 Web MVC framework is also based heavily on annotations for operation and configuration.**
  - This requires developers to have a firm grasp on autowiring and Spring annotation configuration.

  - In order to either refresh your memory or get you up to speed, both autowiring and annotations are covered in some detail in the first chapters of class.
  - In fact, a number of new annotations were added to Spring 3.
  - So even if you are familiar with Spring and annotations, the introductory chapters will likely introduce you to many new annotations.
  - You can and likely will use these annotations in many places in your applications – to include your Spring Web MVC components.
- **Under the new Web MVC architecture, one can argue that the lines between what is a "normal" Spring component and what is a "Web MVC" component are blurred.**
  - Some other framework additions/changes to Spring 3 are likely to be used by all Web MVC applications.
  - The new type conversion system and formatting systems fall into this category.
  - However, strictly speaking, these additions are not part of the Web MVC framework.
  - In fact, these additional can be used outside of Web MVC applications/environments.
  - Therefore, in addition to Web MVC, you also learn some of these new features provided by Spring.

Lab Exercise – there is no lab for this chapter

Chapter Summary

- **Building Web applications can be a difficult task.**
  - Most software engineers soon learn that it is important to divide any application (Web or otherwise) into separate layers.
  - The Model View Controller (MVC) pattern carries this idea further.
  - In MVC applications, the presentation layer is further divided into view and controller layers.
  - In the Java EE arena, several frameworks assist developers in implementing MVC Web applications.
  - In the Spring MVC world, there are many components to orchestrate each request/response to the user.
  - Each component has a specific task. While this might seem to add complexity, it allows for maximum flexibility and customization.
- **If you have used the Spring Web MVC framework in past (prior to Spring 3), recognize that Spring 3 brought major changes to the Web MVC framework.**
  - In many ways, the Spring 3 Web MVC framework is simpler than the old Web MVC framework.
  - Unfortunately, the new framework is quite different and requires, in most cases, a rewrite of applications using the old framework.