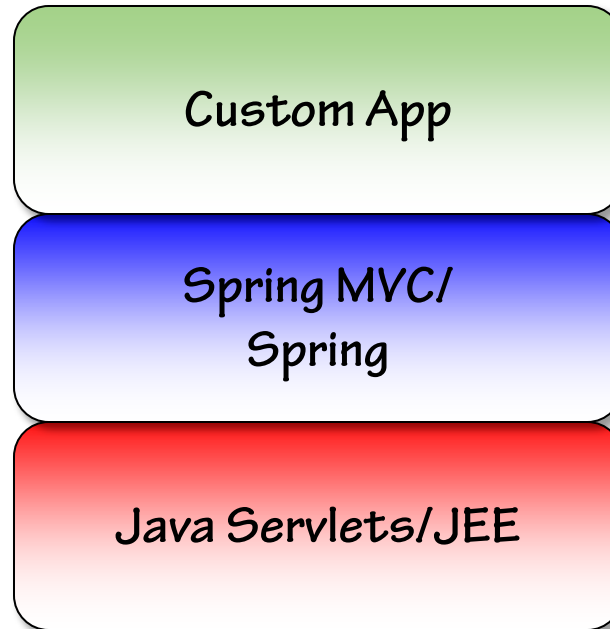


# **Introduction to Spring MVC**

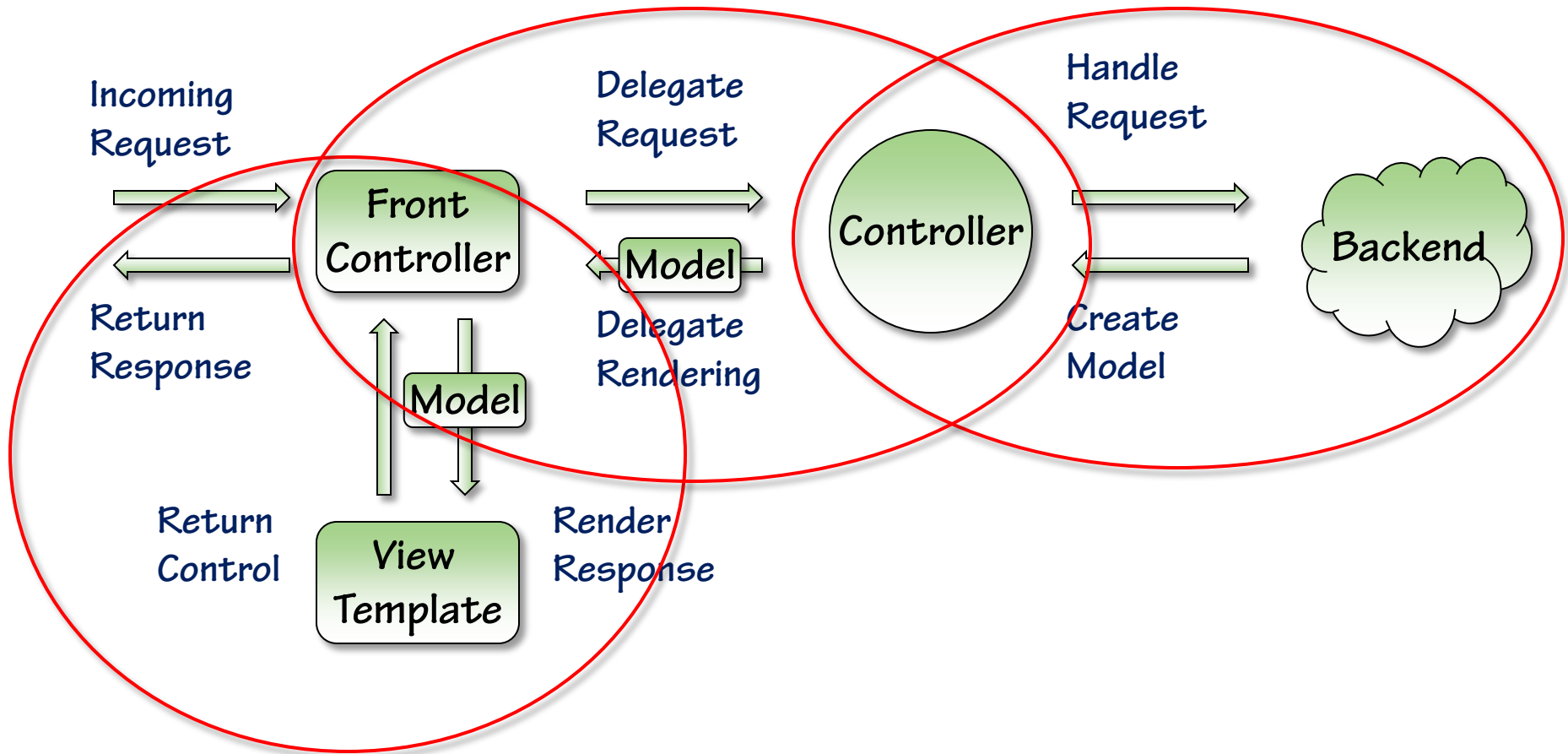
# What is Spring MVC?

- **A web framework built around the principles of Spring**
- **POJO based and Interface driven**
- **Based on a Dispatcher Servlet / Front Controller pattern**
  - MVC stands for Model-View-Controller
- **Very lightweight and unobtrusive compared to other frameworks**
- **Built from the shortcomings of Struts 1**
- **Support for:**
  - Themes
  - Locales/i18n
  - Restful services
  - Annotation based configuration
  - Seamless integration with other Spring Services/Beans

# Architecture



# Request / Response Lifecycle



# Vocabulary

- **DispatcherServlet** – The entry / configuration point for the Application
- **Controller** – Command pattern object that handles the request and determines which view to route to
- **RequestMapping** – The url and request type that a method is tied to
- **ViewResolver** – Used to locate JSP pages or whatever view we are using
- **Servlet-config** – Configuration file per DispatcherServlet
- **POJO** – Plain Old Java Object
- **Bean** – A Spring configured POJO

# App Features

- Allow for entering minutes exercising
- Show total minutes
- Set goal for the day
- Notify when goal is reached



# maven

- Using Maven we only need three dependencies

- Dependencies:

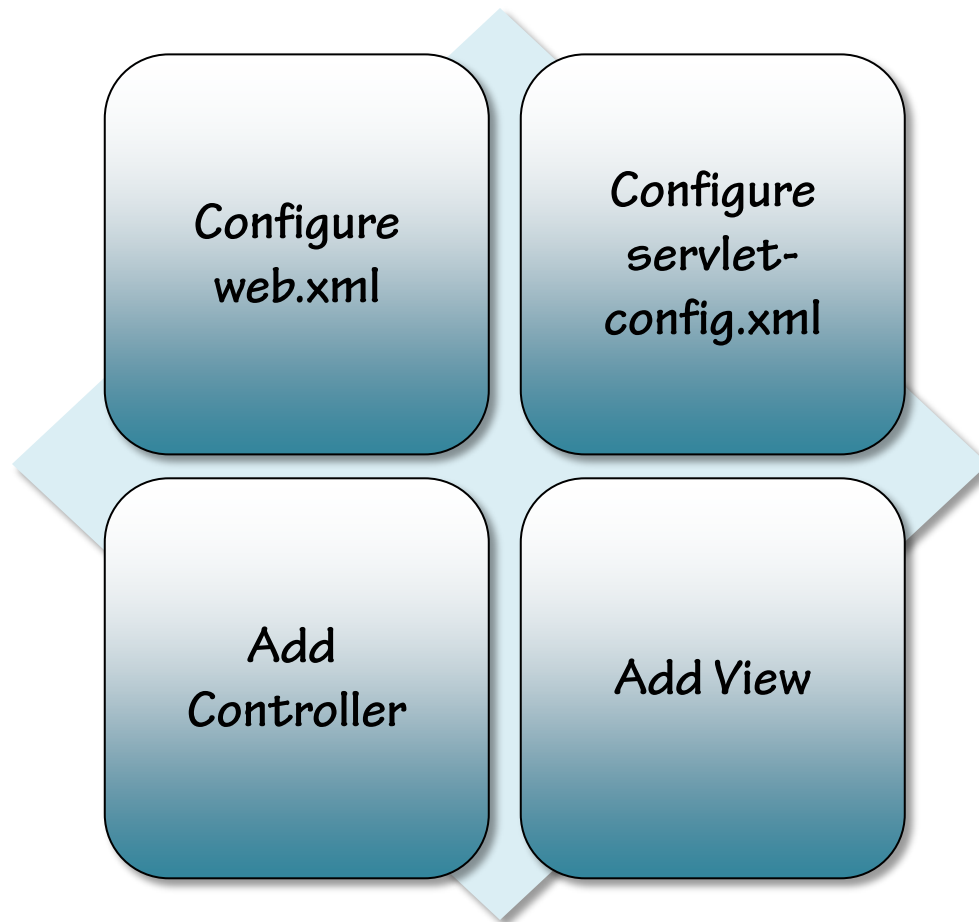
- spring-webmvc
- servlet-api
- jstl

```
<dependency>  
  <groupId>org.springframework</groupId>  
  <artifactId>spring-webmvc</artifactId>  
  <version>3.1.2.RELEASE</version>  
</dependency>
```

```
<dependency>  
  <groupId>javax.servlet</groupId>  
  <artifactId>servlet-api</artifactId>  
  <version>2.5</version>  
  <scope>provided</scope>  
</dependency>
```

```
<dependency>  
  <groupId>javax.servlet</groupId>  
  <artifactId>jstl</artifactId>  
  <version>1.2</version>  
  <scope>provided</scope>  
</dependency>
```

# Spring MVC Configuration





# Tomcat

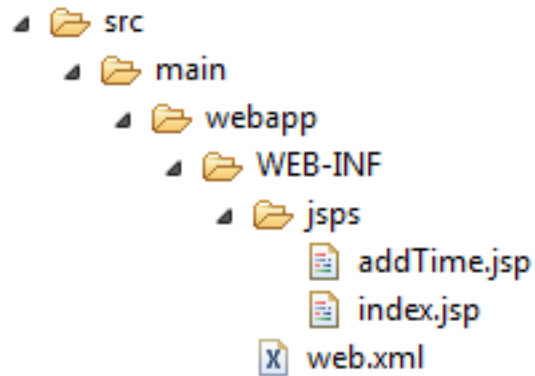
FitnessTracker.war

Tomcat  
Web Container



Apache Tomcat

# Views



`InternalResourceViewResolver`

`addTime.jsp = http://localhost:8080/fitness/addTime`

# Controllers

- **Annotation Based**
  - @Controller
- **Named whatever you want**
  - Typically named around business domain
- **Path set using annotation**
  - @RequestMapping
- **TimeController.java**



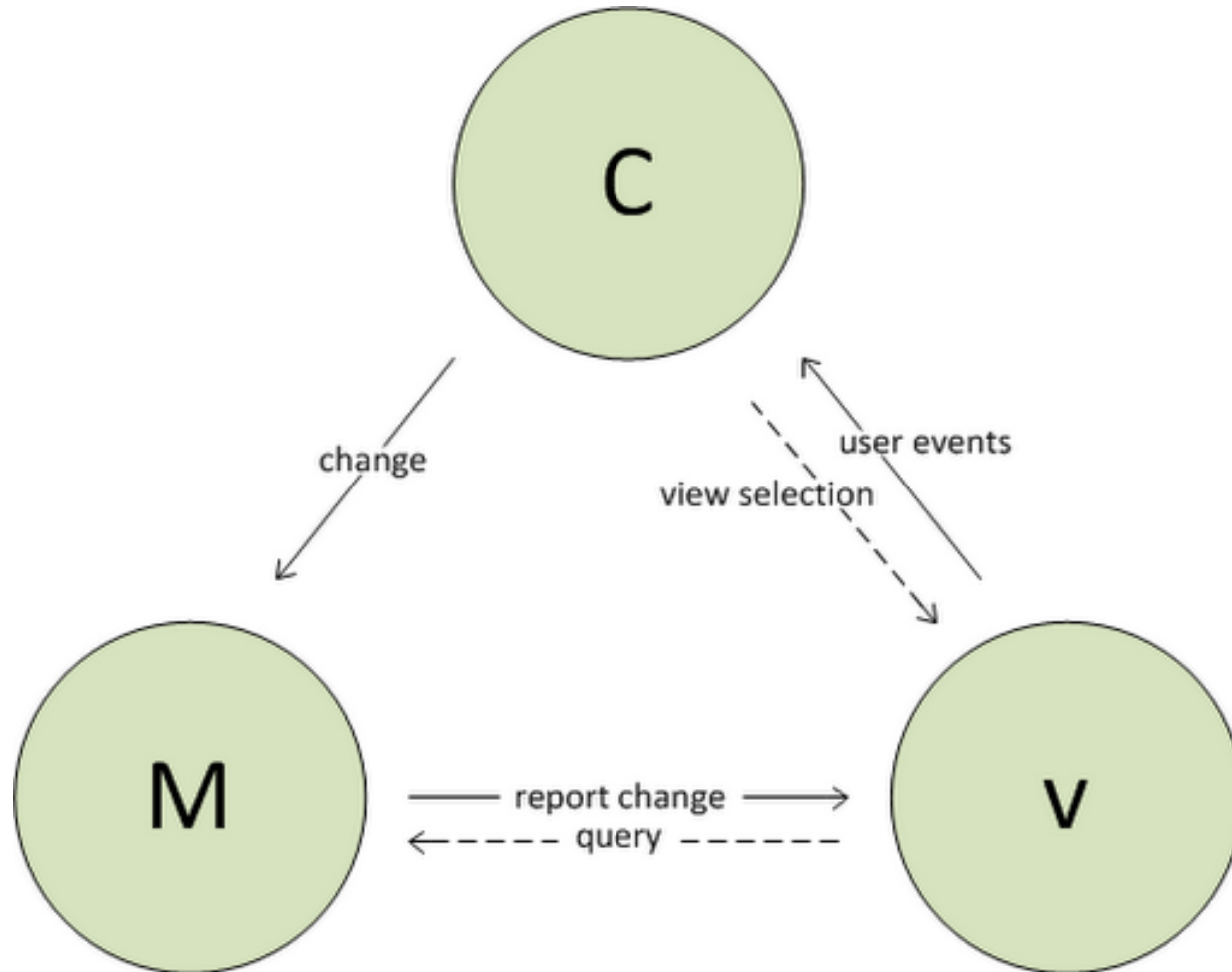
# Namespace

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:p="http://www.springframework.org/schema/p"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:jee="http://www.springframework.org/schema/jee"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xmlns:mvc="http://www.springframework.org/schema/mvc"
  xsi:schemaLocation="
    http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.5.xsd
    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd
    http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-jee-3.0.xsd
    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.0.xsd
    http://www.springframework.org/schema/task http://www.springframework.org/schema/task/spring-task.xsd
    http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd">
```

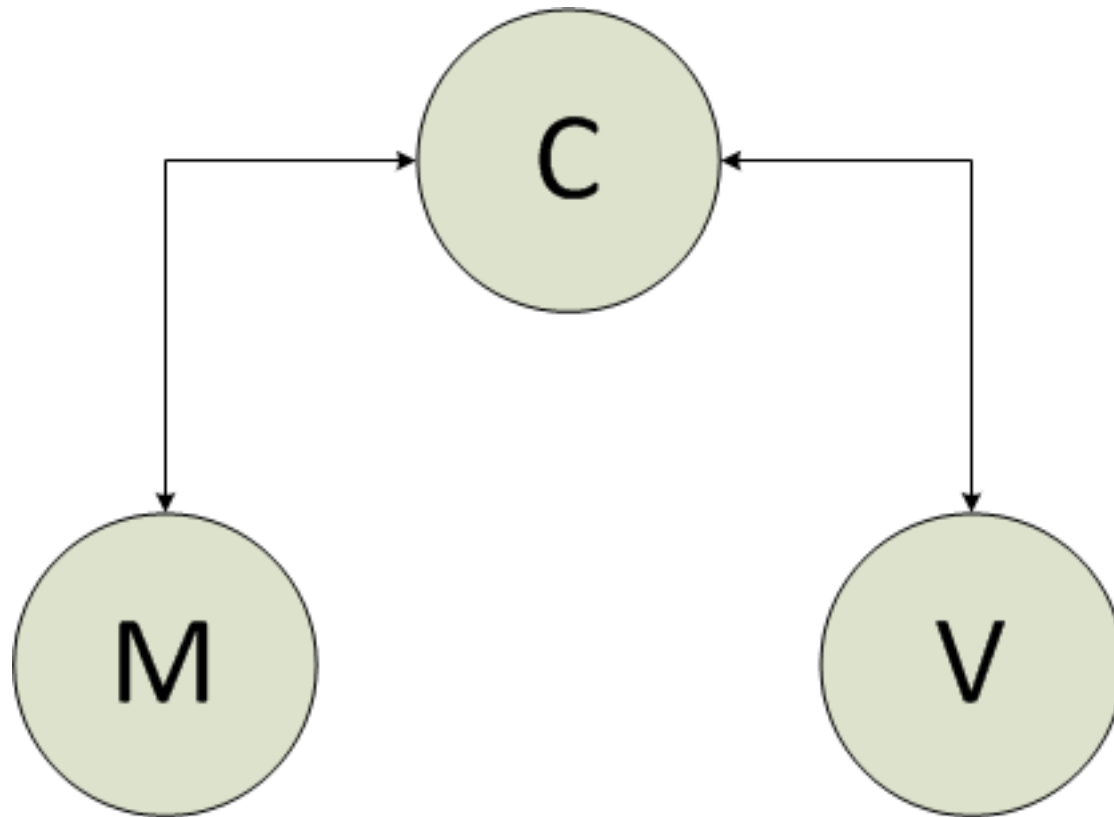
```
  <mvc:annotation-driven />
  <context:component-scan base-package="com.training.controller"/>

  <bean class="org.springframework.web.servlet.view.BeanNameViewResolver"/>
```

# MVC

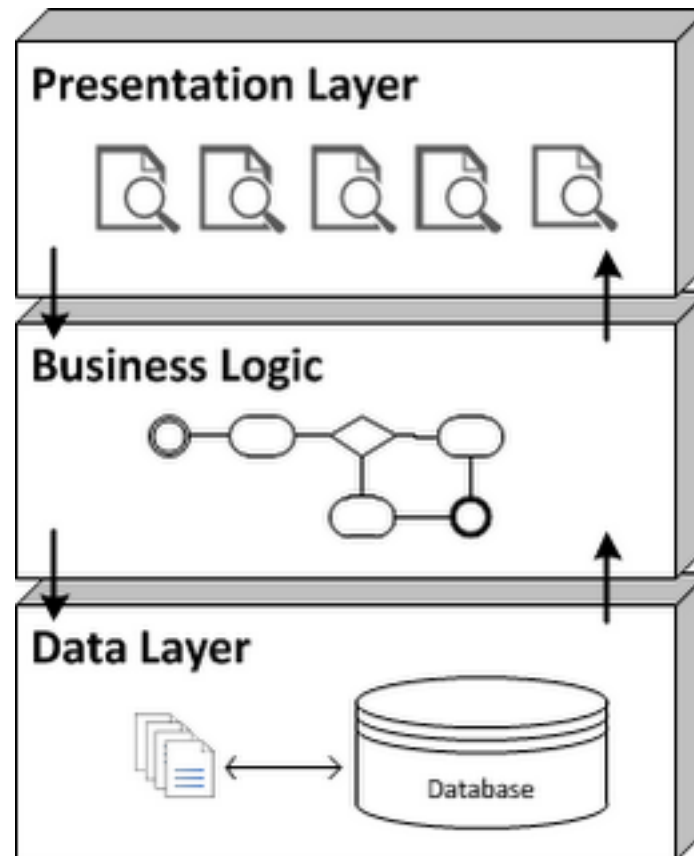


# MVC Web

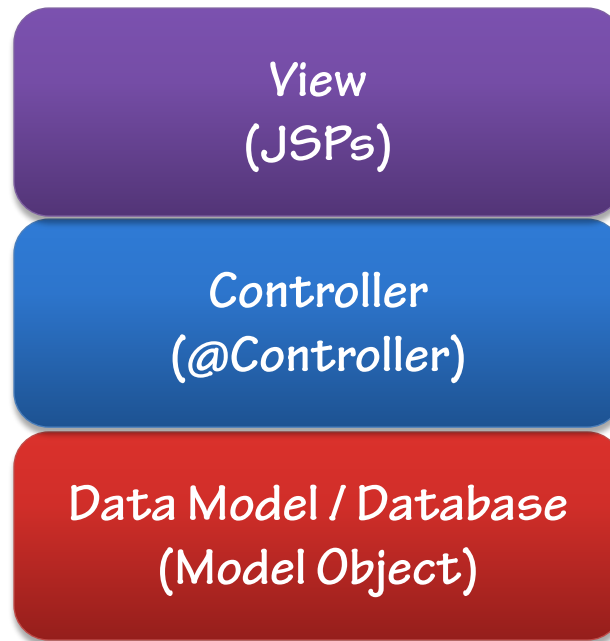


# Tiered Architectures

- Separation of concerns
- Reusable layers
- Maintenance

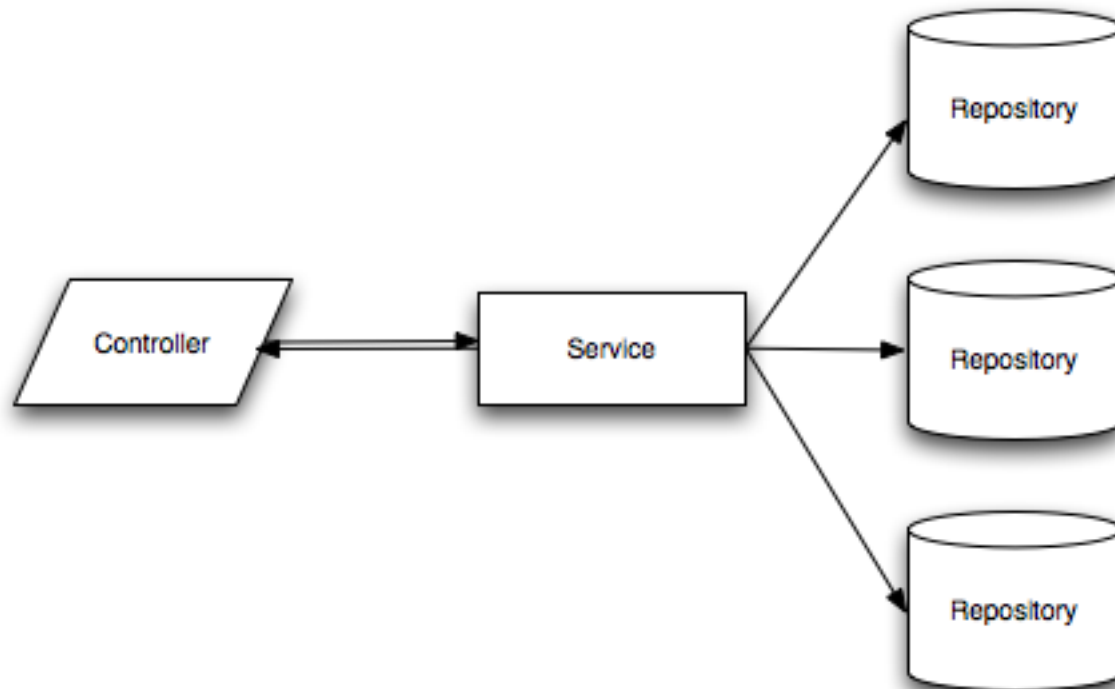


# Layers





# Components



# Controller

- Handles incoming requests and building the response
- Business logic should not be handled here
- Works with the Service and Repository tier for business logic and data gathering
- Annotated with `@Controller`
- Handles Exceptions and routes the view accordingly



# Service

- Annotated with `@Service`
- The Service tier describes the verbs (actions) of a system
- Where the business logic resides
- Ensures that the business object is in a valid state
- Where transactions often begin (two phase commits)
- Often has the same methods as the Repository, but different focus



# Repository

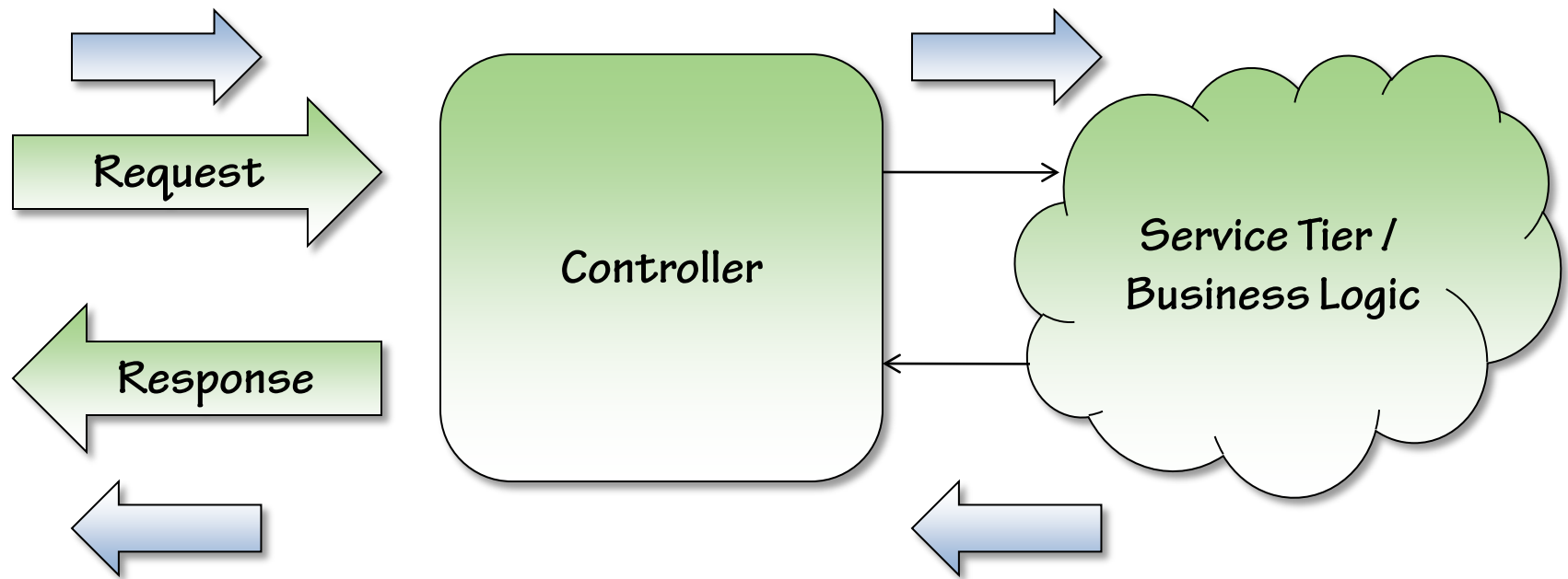
- Annotated with `@Repository`
- The Repository tier describes the nouns (data) of a system
- Focused on persisting and interacting with the database
- One-to-one mapping with an Object
- Often a one-to-one mapping with a database table



# What is a Controller?



# What is a Controller?



# Responsibilities

- Interpret user input and transform to input to a model
- Provide access to business logic
- Determines view based off of logic
- Interprets Exceptions from the business logic / service tier

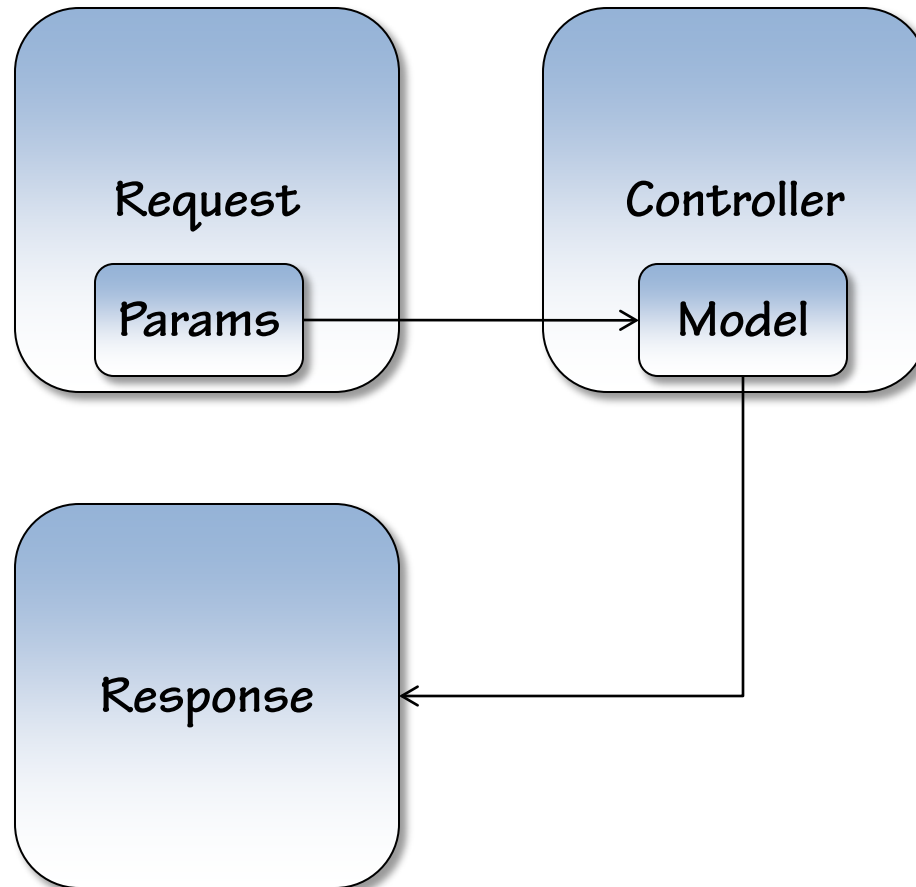


# @Controller

```
@Controller  
public class HelloController {  
    @RequestMapping(value = "/greeting")  
    public String sayHello (Model model) {
```



# Passing Params



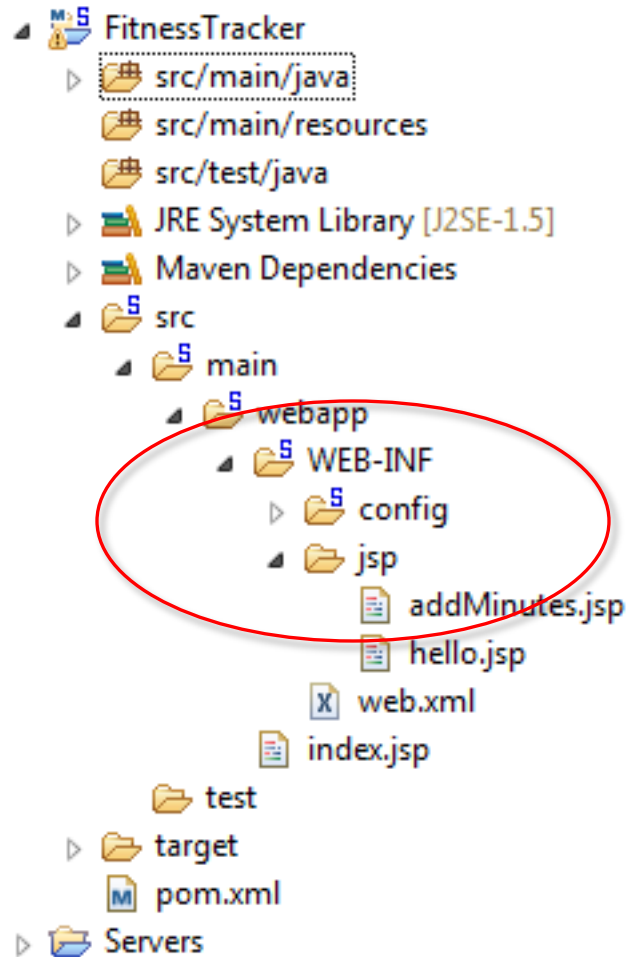
# **@ModelAttribute**

- Used with an HTTP GET
- Used with an HTTP POST
- Works with POJOs
- Can be validated with a Binding Result

# Views



# Conventions



# Resolving a View

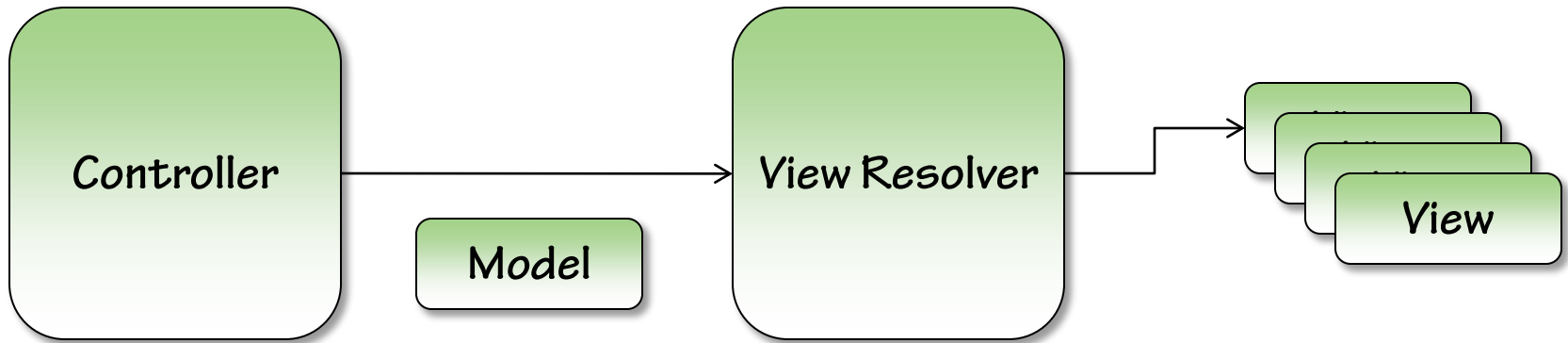
```
@Controller
public class MinutesController {

    @RequestMapping(value = "/addMinutes")
    public String addMinutes(@ModelAttribute ("exercise") Exercise exercise) {

        System.out.println("exercise: " + exercise.getMinutes());

        return "addMinutes";
    }
}
```

# Resolving a View



# View Resolvers (some)

BeanNameViewResolver

ContentNegotiatingViewResolver

FreeMarkerViewResolver

InternalResourceViewResolver

JasperReportsViewResolver

ResourceBundleViewResolver


TilesViewResolver

UrlBasedViewResolver

# Spring Tag Libraries

A light green rounded rectangle with a black border and a subtle drop shadow. The text "spring.tld" is centered inside.

spring.tld

A light green rounded rectangle with a black border and a subtle drop shadow. The text "spring-form.tld" is centered inside.

spring-form.tld

<http://static.springsource.org/spring/docs/current/spring-framework-reference/html/spring.tld.html>

<http://static.springsource.org/spring/docs/current/spring-framework-reference/html/spring-form.tld.html>



# spring.tld

bind

escapeBody

hasBindErrors

htmlEscape

message

nestedPath

theme

transform

url

eval

# Interceptors

- Registered and part of the request lifecycle
- Have the ability to pre-handle and post-handle web requests
- Callback methods used to override or change values
- Commonly used for Locale Changing

```
<bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor">  
    <property name="paramName" value="language" />  
</bean>
```

## spring-form.tld (some)

checkbox

checkboxes

errors

form

hidden

input

label

option

password

select

# Tags

- All of the form tags have an error class associated with them
- There is a specific errors tag for displaying validation errors

```
<form:form>
  <table>
    <tr>
      <td>First Name:</td>
      <td><form:input path="firstName" /></td>
      <td><!-- Show errors for firstName field -->
      <td><form:errors path="firstName" /></td>
    </tr>

    <tr>
      <td>Last Name:</td>
      <td><form:input path="lastName" /></td>
      <td><!-- Show errors for lastName field -->
      <td><form:errors path="lastName" /></td>
    </tr>
    <tr>
      <td colspan="3">
        <input type="submit" value="Save Changes" />
      </td>
    </tr>
  </table>
</form:form>
```

# Validator Interface

- **Validator Interface**
- **ValidationUtils class**
- **BindingResult class**
- **SimpleFormController**



# JSR-303

- Standard for validation
- Annotation based
- Hibernate Validator
- POJO based



**REST**



# Verbs

- What does CRUD stand for?
- POST
- GET
- PUT
- DELETE





# ContentNegotiatingViewResolver

- **Accepts**
  - XML
  - JSON
  - HTML
- **Multiple View Resolvers**
- **Additional JARs needed for the Marshaller**

# **web.xml**

- **Dispatcher Servlet needs to open allow access for various request types.**
- **HTML**
- **PDFs**
- **JSON**
- **XML**

# JQuery

- Javascript framework with multiple plugins
- Easily consumes JSON
- Download JQuery from [http://docs.jquery.com/Downloading\\_jQuery](http://docs.jquery.com/Downloading_jQuery)

