Spring MVC的简单介绍

# Spring MVC学习步骤介绍

## 什么是MVC框架？

## Spring MVC概述

**Spring MVC** 是Spring框架最重要的的模块之一。它以强大的**Spring IoC容器**为基础，并充分利用**容器的特性**来简化它的配置。

## DispatcherServlet(Spring控制器)介绍

## Spring MVC入门示例

Spring MVC HelloWorld

## Spring MVC的配置文件

web.xml和spring-servlet.xml文件配置

## 使用@RequestMapping 映射请求

## 映射请求参数&请求头

## 处理模型数据

## 视图和视图解析



# 什么是MVC框架？

## Model-View-Controller：MVC

**模型-视图-控制器（MVC）**是一个众所周知的以设计界面应用程序为基础的设计模式。它主要通过**分离模型、视图及控制器**在应用程序中的角色将**业务逻辑**从界面中解耦。

## MVC框架的三部分功能介绍

通常：

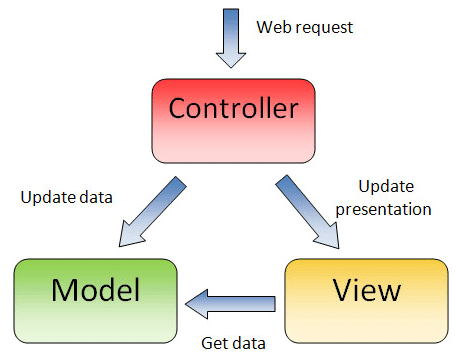
### 模型负责封装应用程序数据在视图层展示。

### 视图仅仅只是展示这些数据，不包含任何业务逻辑。

### 控制器负责接收来自用户的请求，并调用后台服务（manager或者dao）来处理业务逻辑。处理后，后台业务层可能会返回了一些数据在视图层展示。控制器收集这些数据及准备模型在视图层展示。

## MVC框架的核心思想

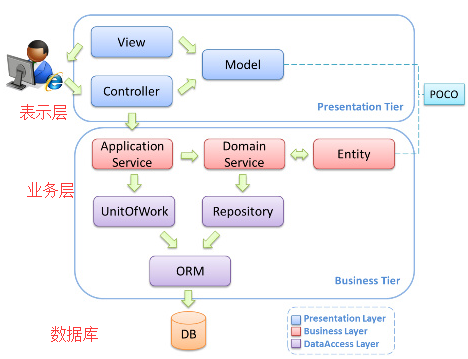
**MVC模式的核心思想**是将**业务逻辑**从界面中分离出来，允许它们单独改变而不会相互影响。



在**Spring MVC**应用程序中，模型通常由**POJO对象**组成，**它在业务层中被处理，在持久层中被持久化**。视图通常是用JSP标准标签库（JSTL）编写的JSP模板。**控制器部分是由dispatcher servlet负责**，在本教程中我们将会了解更多它的相关细节。

## Web应用3层架构：数据-业务-表示

一些开发人员认为**业务层和DAO层类**是MVC模型组件的一部分。我对此持有不同的意见。我不认为业务层及DAO层类为MVC框架的一部分。通常一个web应用是3层架构，即**数据-业务-表示**。**MVC实际上是表示层的一部分。**



# Spring MVC简单介绍

**Spring MVC框架**是一个MVC框架，通过实现**Model-View-Controller模式**来很好地将数据、业务与展现进行分离。从这样一个角度来说，Spring MVC和Structs、Structs2非常类似。Spring MVC的设计是围绕**DispatcherServlet**展开的，DispatcherServlet负责将请求派发到特定的handler。通过可配置的hander mappings、view resolution、locale以及theme resolution来处理请求并且转到对应的视图。

Spring MVC有基于注解版与基础.xml版的两种用法，不过现在的企业级开发基本都使用的是注解版，没别的原因，就是方便而已。因此后面的代码示例，都是基于注解版本的，想了解基于.xml版本的Spring MVC的朋友可以自行上网查询。

# Spring MVC

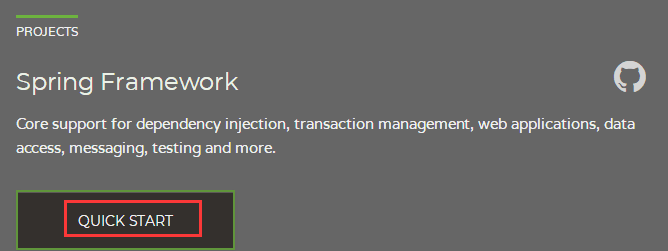
## 引入jar包或Maven引入

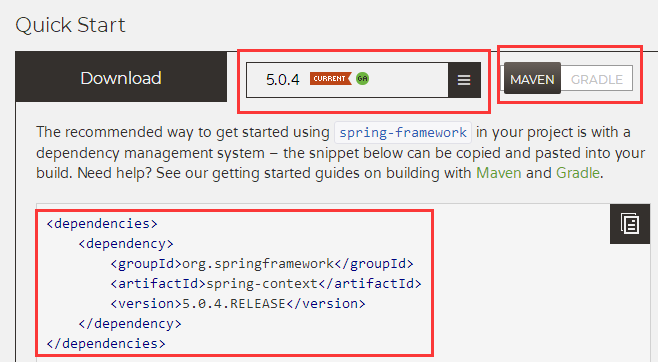
需要引入的jar包：

要搭建一个基础功能的**Spring MVC环境**，必须引入的**jar包**是**beans、context、core、expression、web、webmvc以及commons-logging**。

Maven引入：

<https://projects.spring.io/spring-framework/>





注意：这个dependency包括**beans、context、core、expression、aop、jcl这5个包。**

**Web包、web-mvc需要单独引入。**

**从maven的Repository网站搜索：http://mvnrepository.com/**

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.0.4.RELEASE</version>

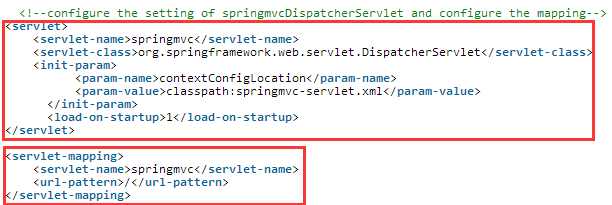
</dependency>

</dependencies>

## 配置web.xml文件

在**web.xml**文件中必须配置servlet，配置servlet同时需要配置servlet-mapping：

在**SpringMVC**中一般都采用org.springframework.web.servlet.**DispatcherServlet**，调度Servlet。初始化参数init-param：就是指定SpringMVC的配置文件位置，classpath表示当前的类路径。**url-pattern**用于开发者选择**哪些路径**是需要让**Spring MVC**来处理的，**“/”**表示所有路径都让SpringMVC来处理。**spring-servlet.xml**文件的名字可以随意起。Spring框架默认使用的配置文件名称为**applicationContext.xml**。

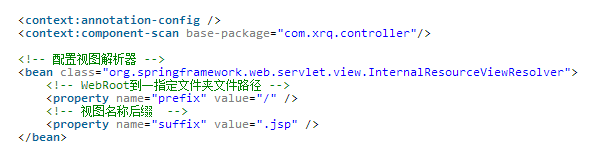


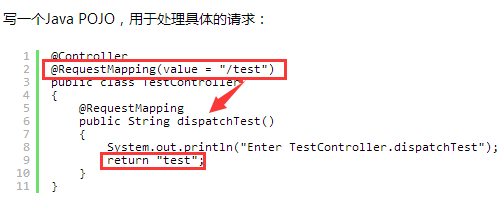
## 创建SpringMVC的配置文件：spring-servlet.xml文件

Context:component-scan配置注解的扫描范围;

配置视图解析器，一般采用的视图解析器是org.springframework.web.servlet.view.**InternalResourceViewResolver**，其中prefix属性限定文件路径，suffix限定后缀。

**视图解析器**的作用：将逻辑视图映射到物理视图。





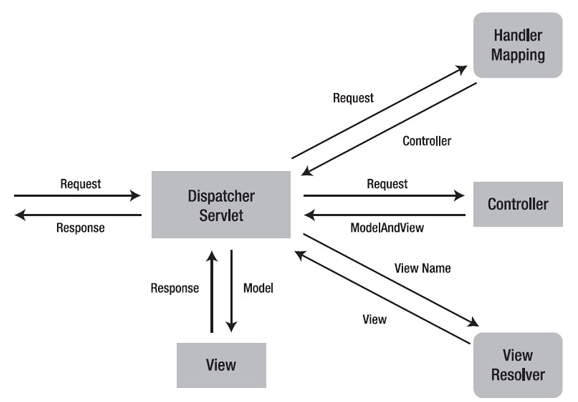
@RequestMapping(value=”/xxx”)既可以用于注解类，也可以注解方法。

由于在**springmvc-servlet.xml**中配置了prefix和suffix，因此return的时候就可以方便一些，不需要写**前缀和后缀**，Spring MVC**默认**会转发到（请求转发是Spring MVC默认的页面跳转方式）”/test.jsp”路径下。

# Dispatcher Servlet(Spring控制器)

在最简单的Spring MVC应用程序中，控制器是唯一的你需要在Java web部署描述文件（即**web.xml**文件）中配置的Servlet。**Spring MVC控制器 ——通常称作Dispatcher Servlet**，**实现了前端控制器设计模式**。**并且每个web请求必须通过它以便它能够管理整个请求的生命周期**。

当一个web请求发送到Spring MVC应用程序，**dispatcher servlet首先接收请求**。然后它组织那些在Spring web**应用程序上下文配置**的（例如实际请求处理控制器和视图解析器）或者**使用注解配置的组件**，所有的这些都需要处理该请求。

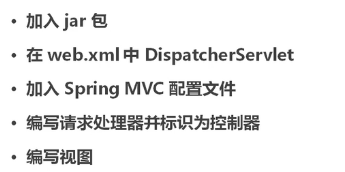


**在Spring3.0中定义一个控制器类，这个类必须标有@Controller注解。**当有@Controller注解的控制器收到一个请求时，它会寻找一个**合适的handler方法**去处理这个请求。这就需要控制器通过**一个或多个handler映射**去把每个请求映射到**handler方法**。为了这样做，**一个控制器类的方法需要被@RequestMapping注解装饰，使它们成为handler方法。**

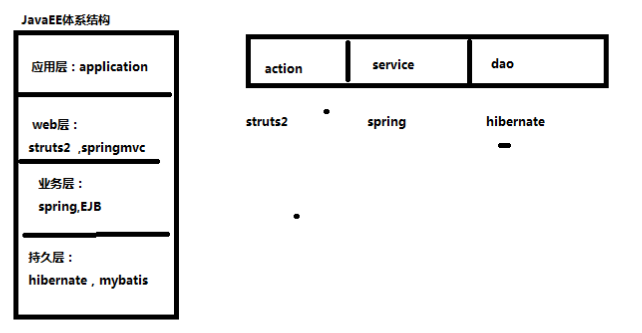
**handler方法处理完请求后，它把控制权委托给视图名与handler方法返回值相同的视图。**为了提供一个灵活的方法，**一个handler方法的返回值并不代表一个视图的实现而是一个逻辑视图**，即没有任何文件扩展名。你可以将这些逻辑视图映射到正确的实现，并将这些实现写入到上下文文件，这样你就可以**轻松的更改视图层代码甚至不用修改请求handler类的代码**。

为一个逻辑名称匹配正确的文件是**视图解析器**的责任。一旦控制器类已将一个视图名称解析到一个视图实现。它会根据视图实现的设计来渲染对应对象。

# Spring MVC的简单实例



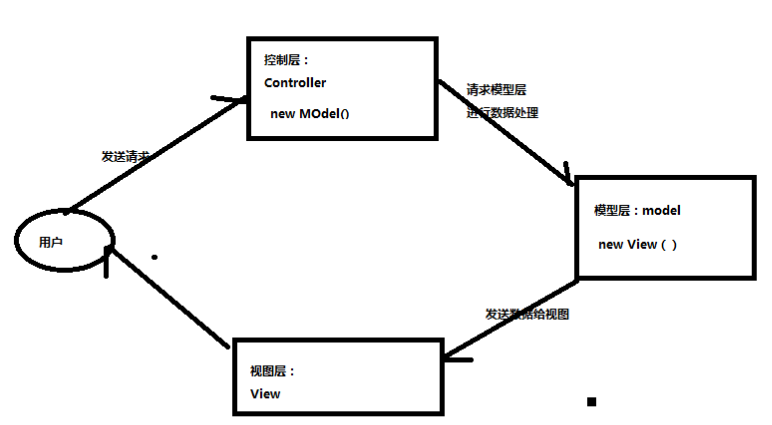
# JavaEE的体系结构



# 面试经常问到：什么是MVC？

# springMVC出现之前的设计模式：

这种方式各层之间通过对象连接，耦合度较大，一层发生改变，则其他层也需要改变。

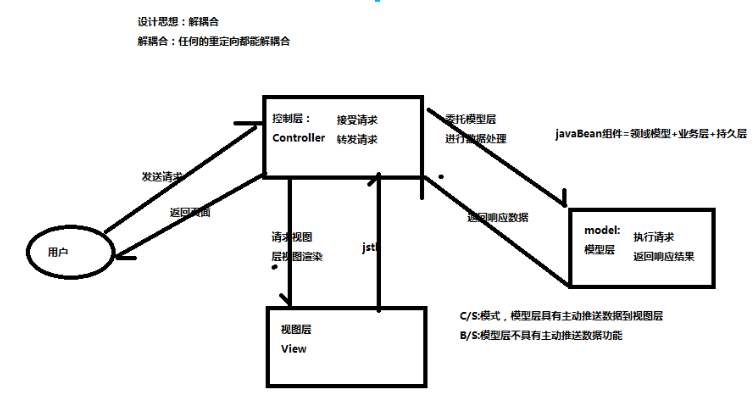


# MVC的设计思想：解耦合。任何的重定向都能解耦合。

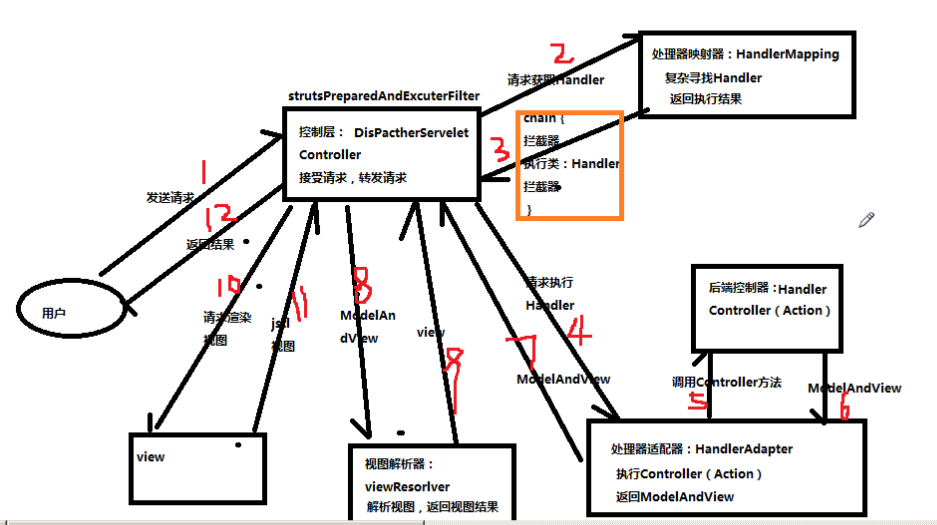
1. 控制层Controller： **接受请求，转发请求**。
2. 模型层Model：**执行请求，返回响应结果**。
3. 视图层View：。

C/S模式：模型层会主动推送数据到视图层，已经过时；

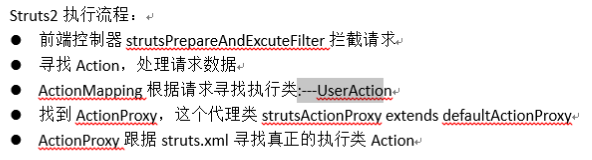
B/S模型：模型层不会主动推送数据到视图层。



1. springMVC的执行流程：



1. struts的执行流程：



1. springmvc的执行流程：

