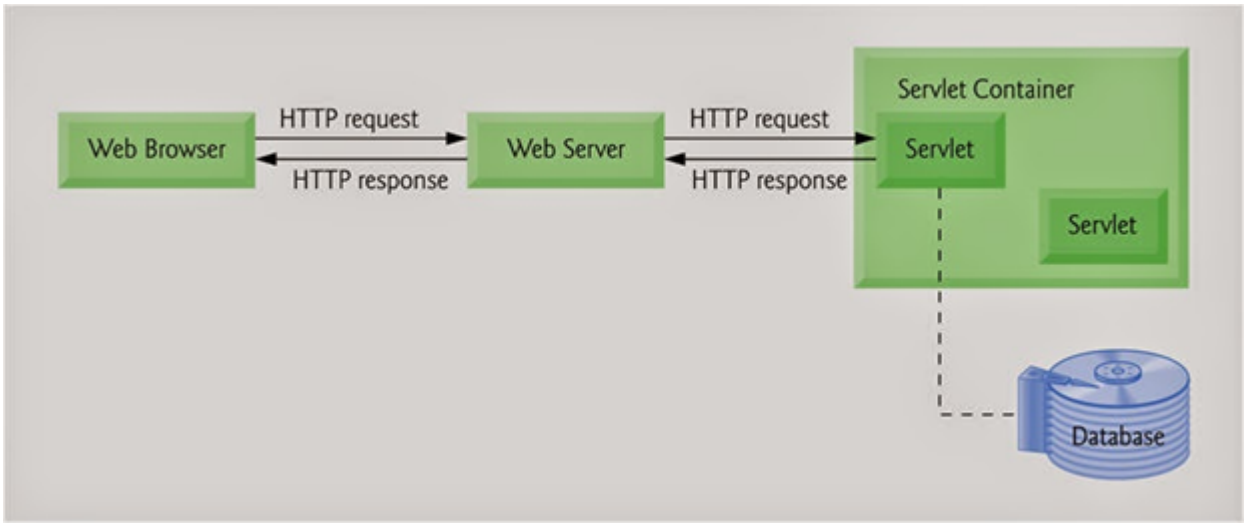
Web MVC 核心

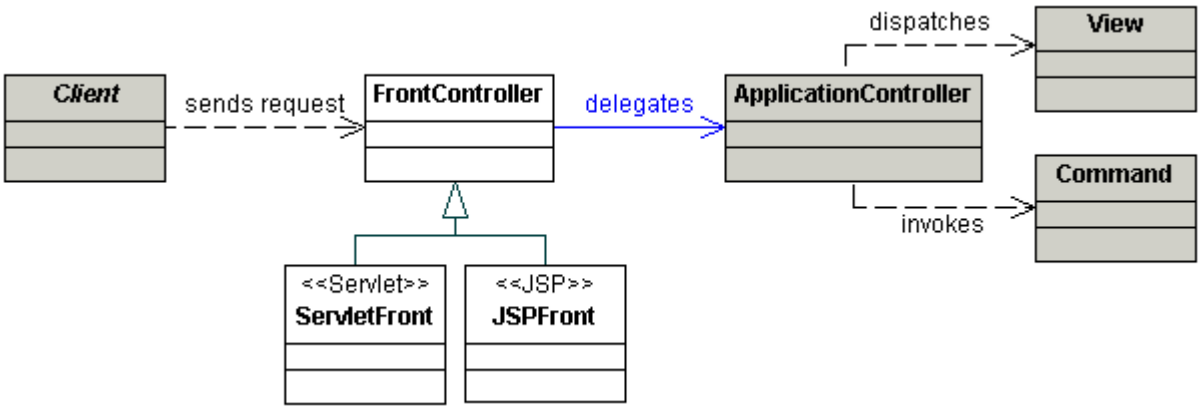
理解 Spring Web MVC 架构

基础架构：Servlet



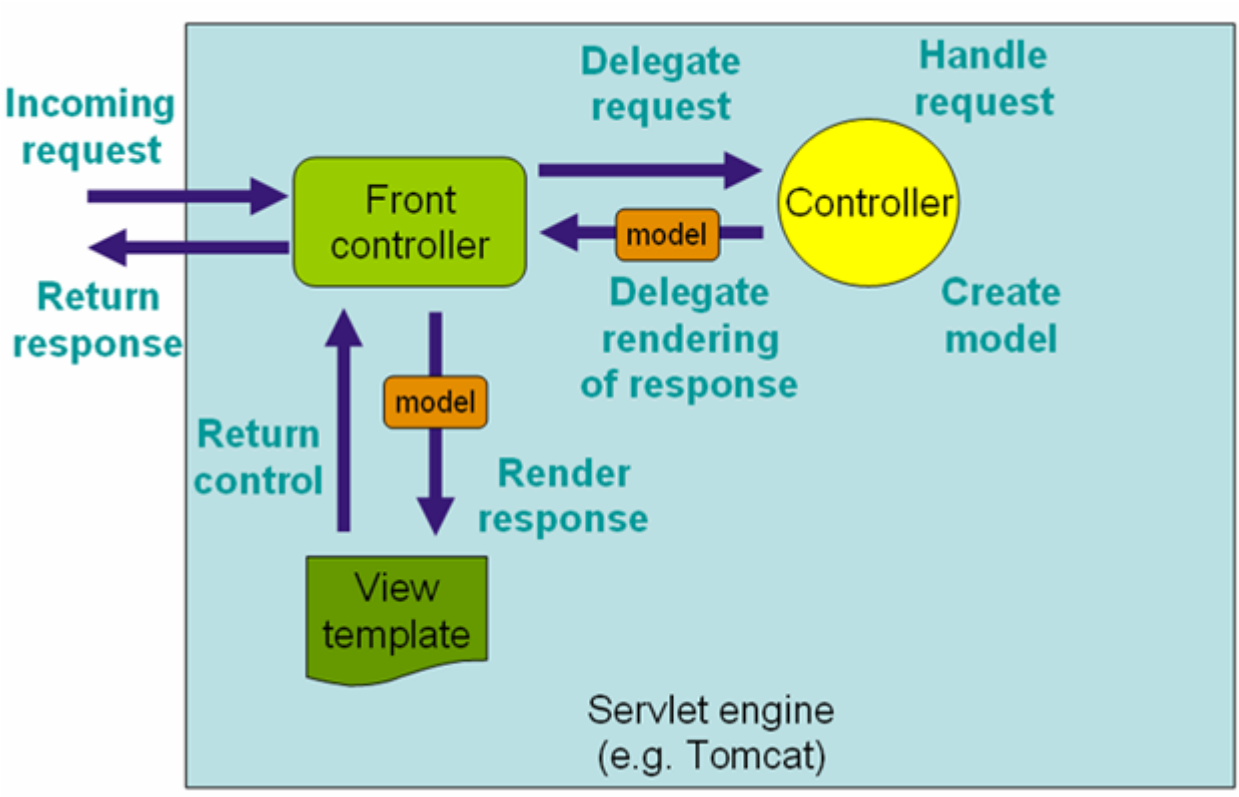
- 特点
  - 请求/响应式(Request/Response)
  - 屏蔽网络通讯的细节
- API 特性
  - 面向 HTTP 协议
  - 完整生命周期
- 职责
- 处理请求
- 资源管理（数据库连接、消息连接、其他）
- 视图渲染

核心架构：前端控制器(Front Controller)



- 资源：http://www.corej2eepatterns.com/FrontController.htm
- 实现：Spring Web MVC DispatcherServlet
  - https://docs.spring.io/spring/docs/1.0.0/javadoc-api/org/springframework/web/servlet/DispatcherServlet.html

Spring Web MVC 架构



认识 Spring Web MVC

Spring Framework 时代的一般认识

实现 Controller

```
@Controller
public class HelloWorldController {

    @RequestMapping("")
    public String index() {
        return "index";
    }

}
```

配置 Web MVC 组件

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd">

    <context:component-scan base-package="com.imooc.web"/>

    <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"/>

    <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"/>

    <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="viewClass" value="org.springframework.web.servlet.view.JstlView"/>
        <property name="prefix" value="/WEB-INF/jsp/"/>
        <property name="suffix" value=".jsp"/>
    </bean>

</beans>
```

部署 DispatcherServlet

```
<web-app>

    <servlet>
```

```
        <servlet-name>app</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>/WEB-INF/app-context.xml</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>app</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>

</web-app>
```

使用可执行 Tomcat Maven 插件

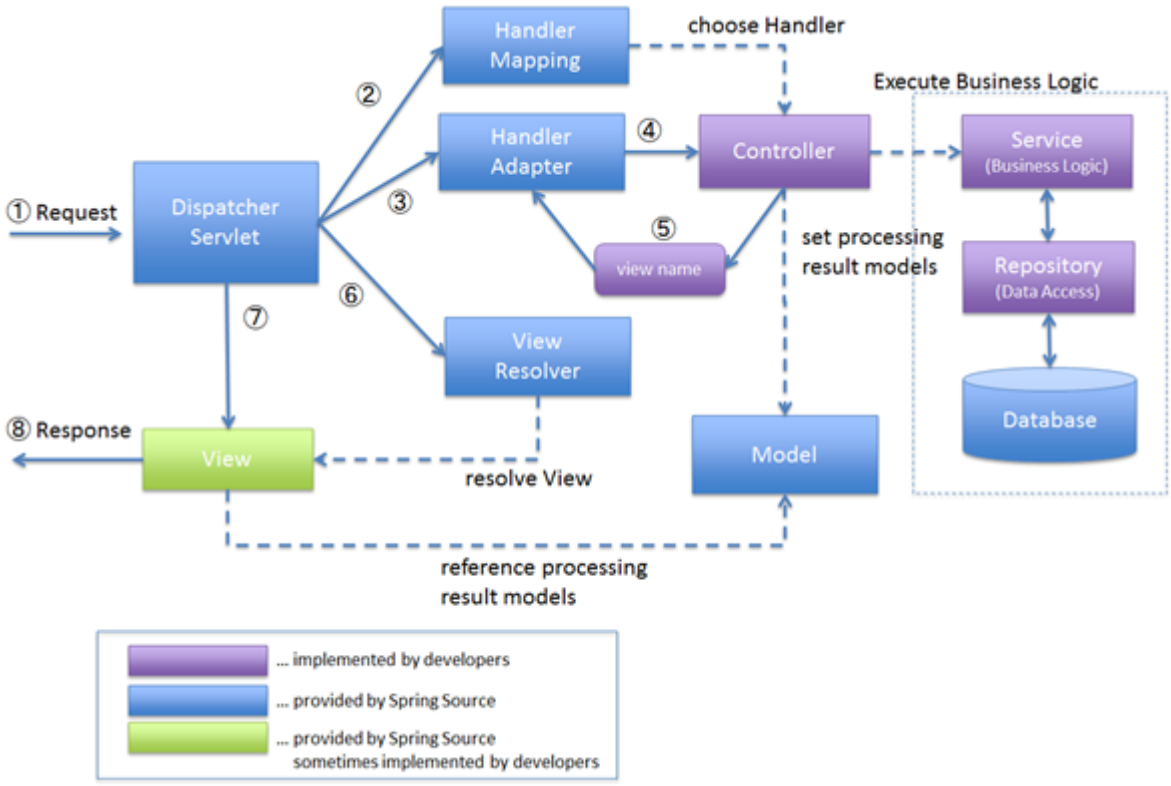
```
<plugin>
    <groupId>org.apache.tomcat.maven</groupId>
    <artifactId>tomcat7-maven-plugin</artifactId>
    <version>2.1</version>
    <executions>
        <execution>
            <id>tomcat-run</id>
            <goals>
                <goal>exec-war-only</goal>
            </goals>
            <phase>package</phase>
            <configuration>
                <!-- ServletContext path -->
                <path>/</path>
            </configuration>
        </execution>
    </executions>
</plugin>
```

Spring Framework 时代的重新认识

Web MVC 核心组件

组件 Bean 类型	说明
HandlerMapping	映射请求（Request）到处理器（Handler）加上其关联的拦截器（HandlerInterceptor）列表，其映射关系基于不同的`HandlerMapping`实现的一些标准细节。其中两种主要`HandlerMapping`实现，`RequestMappingHandlerMapping`支持标注`@RequestMapping`的方法，`SimpleUrlHandlerMapping`维护精确的URI路径与处理器的映射
HandlerAdapter	帮助`DispatcherServlet`调用请求处理器（Handler），无需关注其中实际的调用细节。比如，调用注解实现的 Controller 需要解析其关联的注解.`HandlerAdapter`的主要目的是为了屏蔽与`DispatcherServlet`之间的诸多细节。
HandlerExceptionResolver	解析异常，可能策略是将异常处理映射到其他处理器（Handlers）、或到某个 HTML 错误页面，或者其他。
ViewResolver	从处理器（Handler）返回字符类型的逻辑视图名称解析出实际的`View`对象，该对象将渲染后的内容输出到HTTP 响应中。
LocaleResolver, LocaleContextResolver	从客户端解析出`Locale`，为其实现国际化视图。
MultipartResolver	解析多部分请求（如 Web 浏览器文件上传）的抽象实现

交互流程



Web MVC 注解驱动

- 版本依赖
    - Spring Framework 3.1 +

基本配置步骤

注解配置：`@Configuration`（ Spring 范式注解 ）

组件激活：`@EnableWebMvc`（Spring 模块装配）

自定义组件 ：`WebMvcConfigurer`（Spring Bean）

示例重构

常用注解

注册模型属性：`@ModelAttribute`

读取请求头：`@RequestHeader`

读取 Cookie：`@CookieValue`

校验参数：`@Valid`、`@Validated`

注解处理：`@ExceptionHandler`

切面通知：`@ControllerAdvice`

自动装配

- 版本依赖
    - Spring Framework 3.1 +
    - Servlet 3.0 +

Servlet SPI

Servlet SPI `ServletContainerInitializer`，参考 Servlet 3.0 规范

配合 `@HandlesTypes`

Spring 适配

`SpringServletContainerInitializer`

Spring SPI

基础接口：`WebApplicationInitializer`

编程驱动：`AbstractDispatcherServletInitializer`

注解驱动：`AbstractAnnotationConfigDispatcherServletInitializer`

示例重构

简化 Spring Web MVC

Spring Boot 时代的简化

完全自动装配

自动装配 `DispatcherServlet`：`DispatcherServletAutoConfiguration`

替换 `@EnableWebMvc`：`WebMvcAutoConfiguration`

Servlet 容器 ：`ServletWebServerFactoryAutoConfiguration`

理解自动配置顺序性

绝对顺序：`@AutoConfigureOrder`

相对顺序：`@AutoConfigureAfter`

装配条件

Web 类型判断（`ConditionalOnWebApplication`）

- `WebApplicationType`
  - Servlet 类型：`WebApplicationType.SERVLET`

API 判断（`@ConditionalOnClass`）

- Servlet
  - `Servlet`
- Spring Web MVC
  - `DispatcherServlet`
  - `WebMvcConfigurer`

Bean 判断 (`@ConditionalOnMissingBean` 、`@ConditionalOnBean`)

`WebMvcConfigurationSupport`

外部化配置

Web MVC 配置

`WebMvcProperties`

资源配置

`ResourceProperties`

Spring Boot JSP 依赖

```
<!-- Provided -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-tomcat</artifactId>
    <scope>provided</scope>
</dependency>
<dependency>
    <groupId>org.apache.tomcat.embed</groupId>
    <artifactId>tomcat-embed-jasper</artifactId>
    <scope>provided</scope>
</dependency>
```