

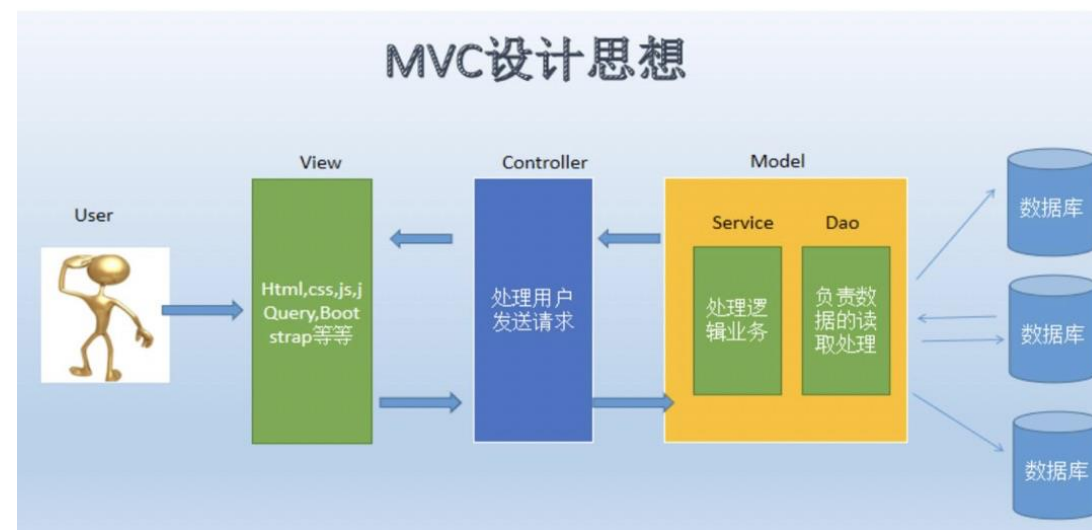
# 服务端开发-Web开发框架 (Web MVC)

陶召胜

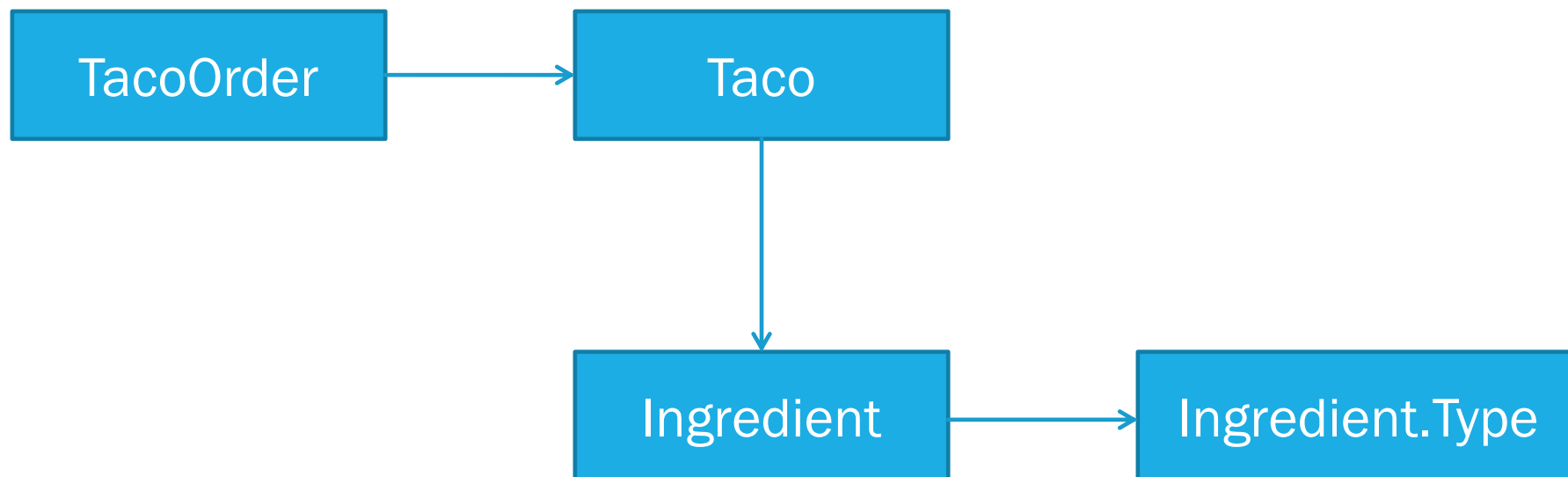
# Spring MVC

- model-view-controller
- 模型 (model) : 存储内容, 指数数据、领域类
- 控制器 (controller) : 处理用户输入
- 视图 (view) : 显示内容

MVC模式用于前后端不分离的开发场景



# 领域类



# lombok

- 依赖

```
<dependency>
```

```
    <groupId>org.projectlombok</groupId>
```

```
    <artifactId>lombok</artifactId>
```

```
</dependency>
```

- 编译期后就不需要了，要排除

```
<groupId>org.springframework.boot</groupId>
```

```
    <artifactId>spring-boot-maven-plugin</artifactId>
```

```
<configuration>
```

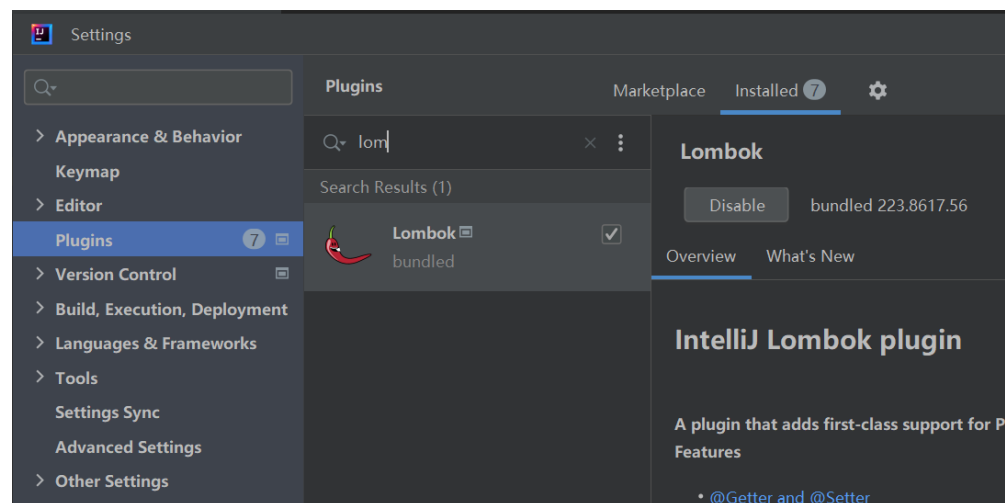
```
    <excludes>
```

```
        <exclude>
```

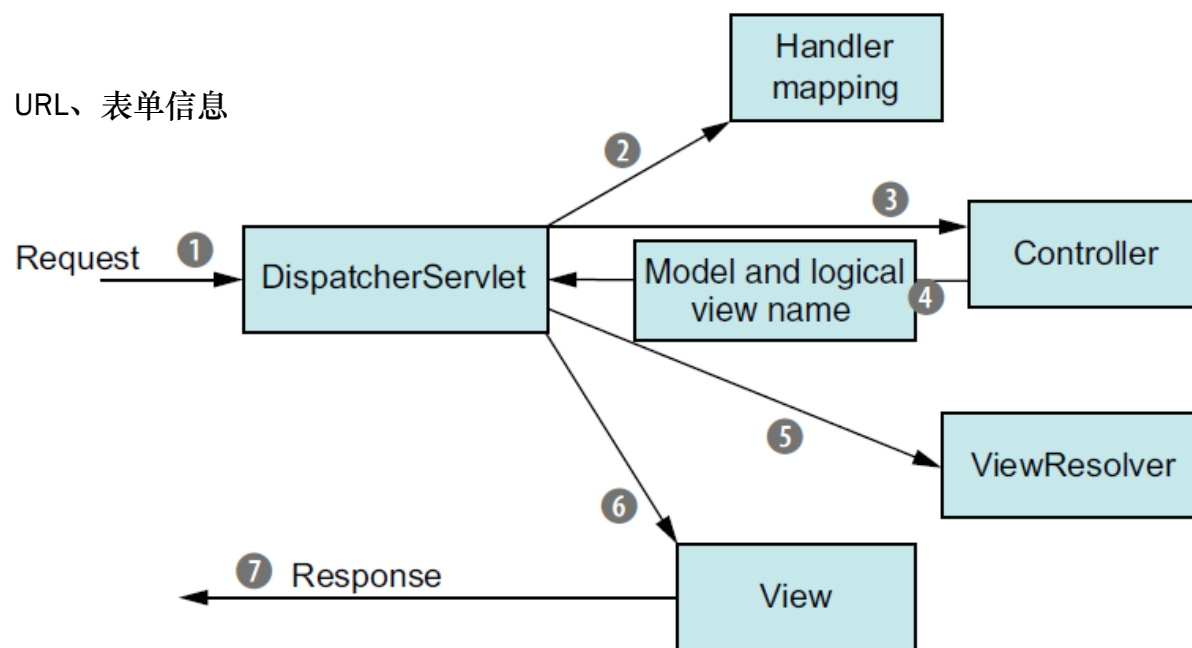
```
            <groupId>org.projectlombok</groupId>
```

```
            <artifactId>lombok</artifactId>
```

另外需要安装IntelliJ IDEA插件



# Spring MVC的请求处理过程



# DesignTacoController控制器实现

@Slf4j

@Controller

@RequestMapping("/design")

@SessionAttributes("tacoOrder")

public class DesignTacoController {

```
public class DesignTacoController {  
    private static final org.slf4j.Logger log =  
        org.slf4j.LoggerFactory.getLogger(DesignTacoController.class);
```

# Spring MVC的请求映射注解

@RequestMapping

@GetMapping

@PostMapping

@PutMapping

@DeleteMapping

@PatchMapping

# Model

- @ModelAttribute
- @SessionAttributes
- Model属性会复制到Servlet Request属性中，这样视图中就可以使用它们用于渲染页面



# 设计视图

```
<dependency>
```

```
  <groupId>org.springframework.boot</groupId>
```

```
  <artifactId>spring-boot-starter-thymeleaf</artifactId>
```

```
</dependency>
```

- Spring Boot的自动配置功能会发现Thymeleaf在类路径中，因此会为Spring MVC自动创建支撑Thymeleaf视图的Bean
- Thymeleaf与Servlet request属性协作（与spring model解耦）

# 处理表单提交

- 接口Converter实现
  - ✓ 将String转换成Ingredient
- redirect重定向
  - `return "redirect:/orders/current";`

## ▼ 常规

请求网址: `http://localhost:8080/design`

请求方法: `POST`

状态代码: 🟡 302

远程地址: `[::1]:8080`

引荐来源网址政策: `strict-origin-when-cross-origin`

## ▼ 响应标头

[查看源代码](#)

**Cache-Control:** `no-store`

**Connection:** `keep-alive`

**Content-Language:** `zh-CN`

**Content-Length:** `0`

**Date:** `Fri, 24 Feb 2023 08:38:03 GMT`

**Keep-Alive:** `timeout=60`

**Location:** `http://localhost:8080/orders/current`

# Spring MVC获取参数的几种方式

- 表单 (form) 参数, 转成model (成员类型可能会用到Converter进行类型转换), 可以使用@Valid校验
- 路径参数, @PathVariable, 例子: /book/{id}
- 请求参数 (查询参数), @RequestParam, 例子: /challenge?mode=2&id=13412431234
- json请求体, @RequestBody, 会用到HttpMessageConverter消息转换器, Rest API

# 校验表单输入

- JavaBean Validation API
- spring-boot-starter-validation (有Hibernate针对 JavaBean Validation API的实现)
- 领域类上添加校验规则
- 控制器中声明校验: @Valid
- 修改表单视图以展现校验错误

# 使用视图控制器 (View Controller)

- 如果一个控制器非常简单，不需要填充模型或处理输入
- 接口WebMvcConfigurer，用于配置
- 简单的从请求URL到视图
  - ✓ `registry.addViewController("/").setViewName("home");` //GET请求
- 接口WebMvcConfigurer也可实现到启动类中

# 更多视图模板库

- Thymeleaf
- FreeMarker
- Groovy Templates
- JSP
- Mustache

# 缓存模板

- application.\* 文件中配置：
  - ✓ `spring.thymeleaf.cache = false`
- 默认缓存打开，如果使用了DevTools则会自动关闭缓存

# 作业

- 参照本节课的例子实现一个录入通讯录的功能
- 要求：
  - ✓ 基于Spring MVC
  - ✓ 数据存入内存中即可，无需持久化
  - ✓ 对录入的数据要有校验

## Spring Boot Contacts

First Name:  名字必须至少1个字符  
Last Name:  名字必须至少1个字符  
Phone #:  无效的手机号码  
Email:  不能为空

## Spring Boot Contacts

First Name:   
Last Name:   
Phone #:   
Email:

- si li : 13345674567, lisi@163.com
- san zhang : 13312341234, zhangsan@163.com

## Spring Boot Contacts

First Name:  名字必须至少1个字符  
Last Name:  名字必须至少1个字符  
Phone #:  无效的手机号码  
Email:  不是一个合法的电子邮件地址



# 提交

- 所有源代码，压缩成一个文件。不包含编译后的class文件（删除target目录）
- 运行截图

谢谢观看！

