Web MVC REST 应用

REST 简介

基本概念

REST = RESTful = Representational State Transfer，is one way of providing interoperability between computer systems on the Internet.

历史

REST 来自于Roy Thomas Fielding 2000年的博士论文 - 《Architectural Styles and the Design of Network-based Software Architectures》

架构约束

- 统一接口（Uniform interface)
- C/S架构（Client-Server）
- 无状态（Stateless）
- 可缓存（Cacheable）
- 分层系统（Layered System）
- 按需代码（Code on demand）（可选）

统一接口（Uniform interface)

- 资源识别（Identification of resources）
  - URI（Uniform Resource Identifier ）
- 资源操作（Manipulation of resources through representations）
  - HTTP verbs：GET、PUT、POST、DELETE
- 自描述消息（Self-descriptive messages）
  - Content-Type
  - MIME-Type
  - Media Type： application/javascript、 text/html
- 超媒体（HATEOAS）
  - Hypermedia As The Engine Of Application State

Web MVC REST 支持

定义

注解	说明	Spring Framework 版本
`@Controller`	应用控制器注解声明，Spring 模式注解	2.5 +
`@RestController`	等效于 `@Controller` + `@ResponseBody`	4.0 +

映射

注解	说明	Spring Framework 版本
`@RequestMapping`	应用控制器映射注解声明	2.5 +
`@GetMapping`	GET 方法映射，等效于 `@RequestMapping(method = RequestMethod.GET)`	4.3 +
`@PostMapping`	POST 方法映射，等效于 `@RequestMapping(method = RequestMethod.POST)`	4.3 +
`@PutMapping`	PUT 方法映射，等效于 `@RequestMapping(method = RequestMethod.PUT)`	4.3 +
`@DeleteMapping`	DELETE 方法映射，等效于 `@RequestMapping(method = RequestMethod.DELETE)`	4.3 +
`@GetMapping`	GET 方法映射，等效于 `@RequestMapping(method = RequestMethod.GET)`	4.3 +
`@PatchMapping`	PATCH 方法映射，等效于 `@RequestMapping(method = RequestMethod.PATCH)`	4.3 +

请求

注解	说明	Spring Framework 版本
`@RequestParam`	获取请求参数	2.5 +
`@RequestHeader`	获取请求头	3.0 +
`@CookieValue`	获取Cookie值	3.0 +
`@RequestBody`	获取完整请求主体内容	3.0 +
`@PathVariable`	获取请求路径变量	3.0 +
`RequestEntity`	获取请求内容（包括请求主体和请求头）	4.1 +

响应

注解	说明	Spring Framework 版本
`@ResponseBody`	响应主题注解声明	2.5 +
`ResponseEntity`	响应内容（包括响应主体和响应头）	3.0.2 +
`ResponseCookie`	响应 Cookie 内容	5.0 +

拦截

注解	说明	Spring Framework 版本
`@RestControllerAdvice`	`@RestController` 注解切面通知	4.3 +
`HandlerInterceptor`	处理方法拦截器	1.0

跨域

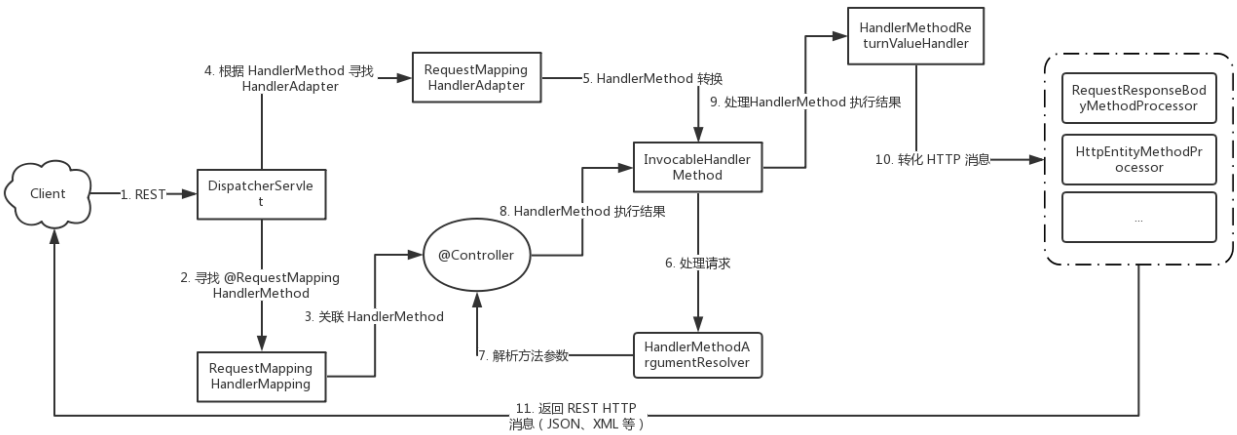
注解	说明	Spring Framework 版本
`@CrossOrigin`	资源跨域声明注解	4.2 +
`CorsFilter`	资源跨域拦截器	4.2 +
`WebMvcConfigurer#addCorsMappings`	注册资源跨域信息	4.2 +

REST 内容协商

核心组件

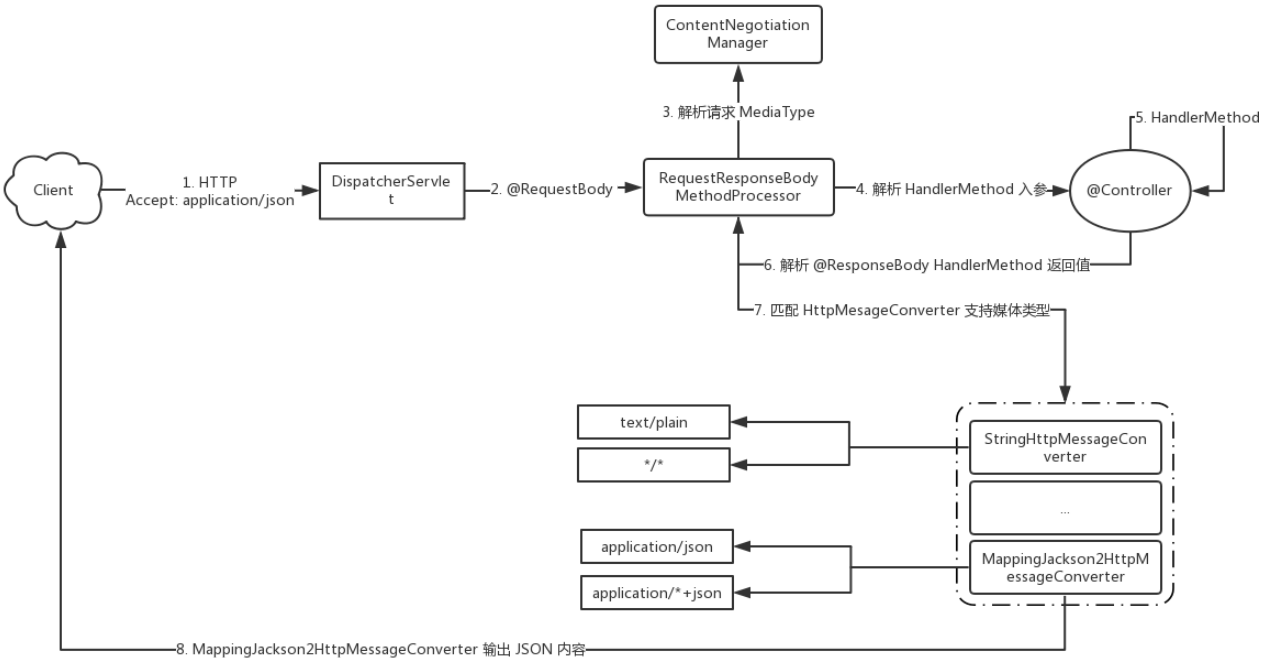
组件名称	实现	说明
内容协商管理器	`ContentNegotiationManager`	`ContentNegotiationStrategy` 控制策略
媒体类型	`MediaType`	HTTP 消息媒体类型，如 `text/html`
消费媒体类型	`@RequestMapping#consumes`	请求头 `Content-Type` 媒体类型映射
生产媒体类型	`@RequestMapping#produces`	响应头 `Content-Type` 媒体类型映射
HTTP消息转换器	`HttpMessageConverter`	HTTP 消息转换器，用于反序列化 HTTP 请求或序列化响应
Web MVC 配置器	`WebMvcConfigurer`	配置 REST 相关的组件
处理方法	`HandlerMethod`	`@RequestMapping` 标注的方法
处理方法参数解析器	`HandlerMethodArgumentResolver`	用于 HTTP 请求中解析 `HandlerMethod` 参数内容
处理方法返回值解析器	`HandlerMethodReturnValueHandler`	用于 `HandlerMethod` 返回值解析为 HTTP 响应内容

Spring Web MVC REST 处理流程



源码分析

Spring Web MVC REST 内容协商处理流程



源码分析

理解请求的媒体类型

经过 `ContentNegotiationManager` 的 `` `ContentNegotiationStrategy `` 解析请求中的媒体类型，比如：`Accept` 请求头

- 如果成功解析，返回合法 `MediaType` 列表
- 否则，返回单元素 `*/*` 媒体类型列表 - `MediaType.ALL`

理解可生成的媒体类型

返回 `@Controller` `HandlerMethod` `@RequestMapping.produces()` 属性所指定的 `MediaType` 列表：

- 如果 `@RequestMapping.produces()` 存在，返回指定 `MediaType` 列表
- 否则，返回已注册的 `HttpMessageConverter` 列表中支持的 `MediaType` 列表

理解 `@RequestMapping#consumes`

用于 `@Controller` `HandlerMethod` 匹配：

- 如果请求头 `Content-Type` 媒体类型兼容 `@RequestMapping.consumes()` 属性，执行该 `HandlerMethod`
- 否则 `HandlerMethod` 不会被调用

理解 `@RequestMapping#produces`

用于获取可生成的 `MediaType` 列表

- 如果该列表与请求的媒体类型兼容，执行第一个兼容 `HttpMessageConverter` 的实现，默认 `@RequestMapping#produces` 内容到响应头 `Content-Type`
- 否则，抛出 `HttpMediaTypeNotAcceptableException` , HTTP Status Code : 415

扩展 REST 内容协商

自定义 `HttpMessageConverter`

需求

实现 `Content-Type` 为 `text/properties` 媒体类型的 `HttpMessageConverter`

实现步骤

- 实现 `HttpMessageConverter` - `PropertiesHttpMessageConverter`
- 配置 `PropertiesHttpMessageConverter` 到 `WebMvcConfigurer#extendMessageConverters`

自定义 `HandlerMethodArgumentResolver`

需求

- 不依赖 `@RequestBody` ， 实现 `Properties` 格式请求内容，解析为 `Properties` 对象的方法参数
- 复用 `PropertiesHttpMessageConverter`

实现步骤

- 实现 `HandlerMethodArgumentResolver` - `PropertiesHandlerMethodArgumentResolver`
- ~~配置 `PropertiesHandlerMethodArgumentResolver` 到 `WebMvcConfigurer#addArgumentResolvers`~~
- `RequestMappingHandlerAdapter#setArgumentResolvers`

自定义 `HandlerMethodReturnValueHandler`

需求

- 不依赖 `@ResponseBody` ，实现 `Properties` 类型方法返回值，转化为 `Properties` 格式内容响应内容
- 复用 `PropertiesHttpMessageConverter`

实现步骤

- 实现 `HandlerMethodReturnValueHandler` - `PropertiesHandlerMethodReturnValueHandler`
- ~~配置 `PropertiesHandlerMethodReturnValueHandler` 到 `WebMvcConfigurer#addReturnValueHandlers`~~
- `RequestMappingHandlerAdapter#setReturnValueHandlers`

跨域访问

当前页面渲染请求为 `http://localhost:8080/index.html`

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title>CORS 示例</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <script src="https://code.jquery.com/jquery-3.3.1.min.js" ></script>
</head>
<body>
    <div id="message"></div>
</body>

<script>
$(document).ready(function(){
     // Ajax 跨域 GET 请求
     $.get( "http://api.rest.org:8080/hello", function( data ) {
                alert( data );
                $( "#message" ).html( data );
        });
});
</script>

</html>
```

注解驱动 `@CrossOrigin`

代码驱动 `WebMvcConfigurer#addCorsMappings`