JavaEE平台技术
SpringMVC和RestFul API

邱明 博士

厦门大学信息学院

mingqiu@xmu.edu.cn

提纲

- RestFul API
- RestFul API请求
- JSON
- SpringMVC
- 参数的合法性检查
- 跨域访问问题
- Servlet并发机制

1. RestFul API

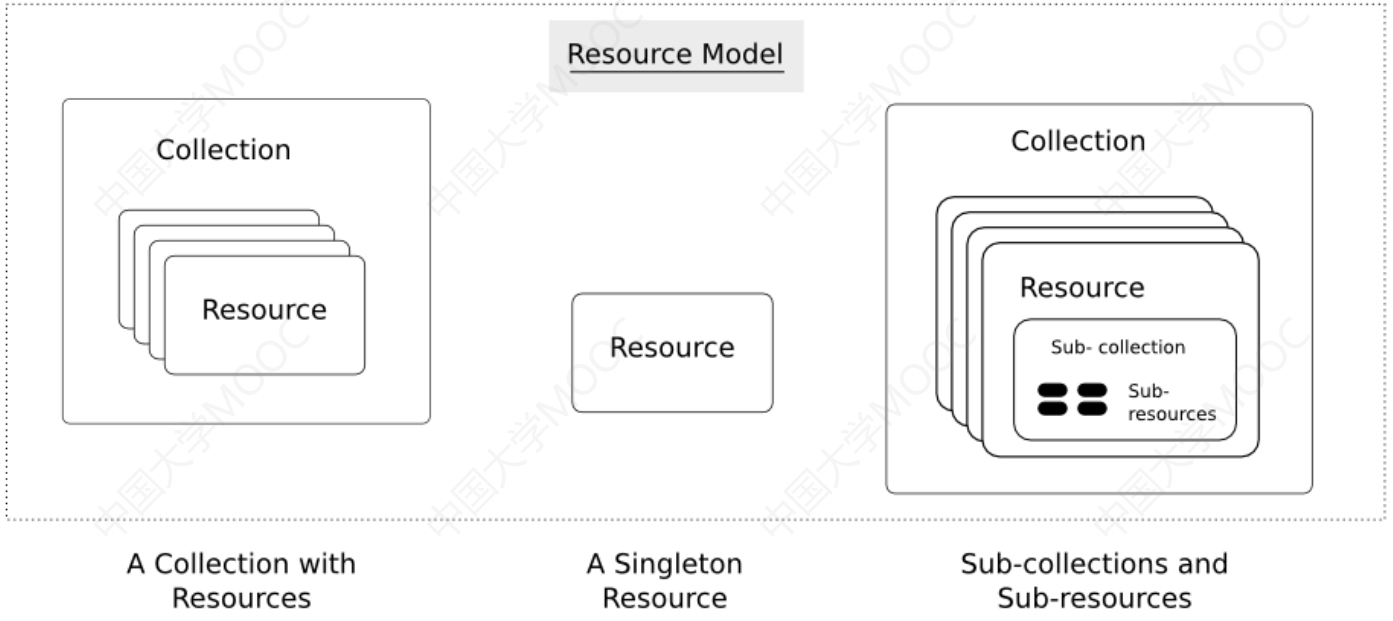
- **Restful API**

1. RestFul API

- REST的最基本特征是面向资源，
  - Resource – 资源
    - 资源是构成应用系统的所有的基本概念
  - Representational – 表述
    - REST的资源可以用XML，JSON，纯文本等多种方式表述
  - State Transfer – 状态转移
    - 客户端通过HTTP的GET/POST/PUT/DELETE/PATCH对资源进行操作，实现资源的状态转移

1. RestFul API

- 资源



1. RestFul API

- Restful API用URL表示特定的资源或资源集合

  https://demo.xmu.edu.cn/goods

  https://demo.xmu.edu.cn/brands

  https://demo.xmu.edu.cn/categories

1. RestFul API

- URL构成

URL	描述
/categories	商品的所有一级分类categories
/categories/1	id为1的一级分类
/categories/1/subcategories	id为1的一级分类下的所有二级分类
/subcategories/2	id为2的二级分类
/subcategories/2/goods	id为2的二级分类下所有商品
/categories/1/goods	id为1的一级分类下所有商品
/goods/5	id为5的商品

2. RestFul API 操作

- 状态转移-利用HTTP的五种请求方法

请求方法	用途
POST	创建一个新的资源并返回新资源
GET	查询特定资源
PUT	修改资源的全部属性并返回修改后的资源
PATCH	修改资源的特定属性并返回修改后的资源
DELETE	删除资源

2. RestFul API 操作

- HTTP状态码

HTTP状态码	描述
2**	操作成功接收并处理
3**	重定向
4**	客户端错误，请求包含语法错误
5**	服务器错误，服务器在处理请求的过程中发生了错误

2. RestFul API 操作

- HTTP状态码

HTTP状态码	描述
2**	操作成功接收并处理
3**	重定向
4**	客户端错误，请求包含语法错误
5**	服务器错误，服务器在处理请求的过程中发生了错误

2. RestFul API 操作

- 所有请求共用的HTTP状态码

共用HTTP状态码	描述
400	参数格式错误（缺少或不符合要求）
401	用户需登录访问
403	用户无权限访问
409	资源冲突或资源被锁定（由于事务回滚造成的）
500	服务器通用错误（服务器程序错误）
503	服务当前无法处理请求（服务器忙）

2. RestFul API 操作

- GET请求

URL	描述
GET /categories	获取所有的一级商品分类
GET /categories/12	获取id为12的一级商品分类对象
GET /categories/12/goods	获取id为12的一级商品分类下所有商品
GET /categories/12/subcategories	获取id为12的分类下的所有子分类
GET /goods?name=电视机&sort=price&order=acend	获取商品名称为电视机的商品，按照价格升序排序
GET /hot-goods?page=2&pagesize=10	以分页方式获取所有秒杀的商品中的第二页，每页10个商品

2. RestFul API 操作

- Get请求的HTTP状态码

HTTP状态码	描述
200	请求成功，数据在Response中返回

2. RestFul API 操作

- POST请求

URL	描述
POST /categories	创建一个新的一级商品分类
POST /categories/12/subcategories	在id为12的一级商品分类下创建一个二级分类
POST /subcategories/19/goods/1	在id为19的二级商品分类下增加id为1的商品
POST /orders/1000/payments	为id为1000的订单付款
POST /login	用户登录

2. RestFul API 操作

- Post请求的HTTP状态码

HTTP状态码	描述
201	新资源成功，数据在Response中返回
202	已经接受处理请求但尚未完成（异步处理）

2. RestFul API 操作

- PUT请求

URL	描述
PUT /categories/12	修改id为12的一级商品分类
PUT /subcategories/13	修改id为13的二级商品分类
PUT /goods/5/onshelves	将id为5的商品上架
PUT /goods/5/offshelves	将id为5的商品下架

2. RestFul API 操作

- PUT请求的HTTP状态码

HTTP状态码	描述
200	修改成功
202	已经接受处理请求但尚未完成（异步处理）

2. RestFul API 操作

- DELETE请求

URL	描述
DELETE /categories/12	删除id为12的一级商品分类
DELETE /categories/12/goods/5	在id为12的分类中将id为5的商品移除

2. RestFul API 操作

- Delete请求的HTTP状态码

HTTP状态码	描述
200	删除成功
202	已经接受处理请求但尚未完成（异步处理）

3. JSON

- JSON（JavaScript Object Notation）
  - 是一种轻量级的数据交换格式。它采用完全独立于编程语言的文本格式来存储和表示数据。JSON易于人阅读和编写的同时，便于机器解析和生成，可有效地提升网络传输效率，是理想的数据交换语言。

3. JSON

- JSON的构成
  - 值可以是对象、数组、数字、字符串或者三个字面值(false、null、true)中的一个。

对象:

```
{"name": "John Doe", "age": 18, "address": {"country" : "china", "zip-code": "10000"}}
```

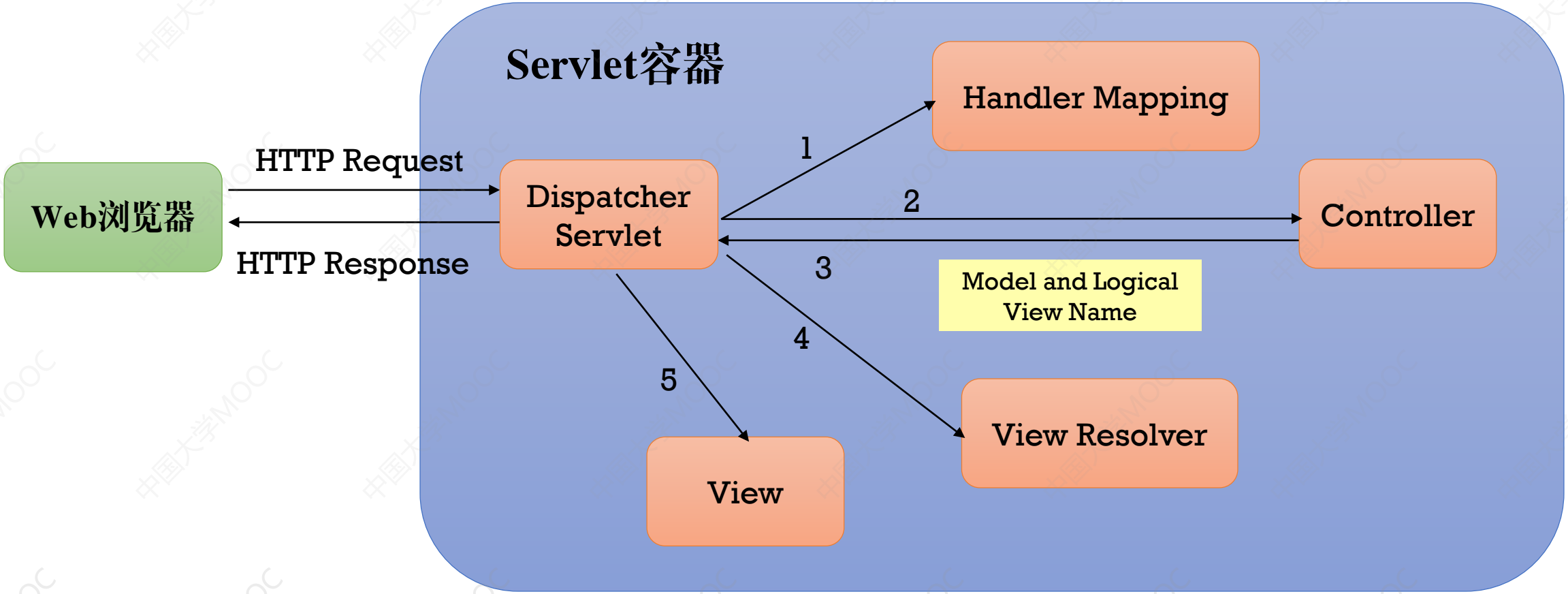
数组:

```
[3, 1, 4, 1, 5, 9, 2, 6]
```

4. Spring MVC

- Spring MVC中的Controller负责接收HTTP Request
  - 交给View生成HTML页面 （传统的方式）
  - 直接把数据写入到Http Response中（Restful）

4. Spring MVC


Servlet容器
Handler Mapping
HTTP Request
Web浏览器
Dispatcher Servlet
Controller
HTTP Response
1
2
3
Model and Logical View Name
4
5
View Resolver
View

4. Spring MVC

- RestFul Controller的注解
  - @RestController
    - 与@Controller标签相同，用于标注在类定义前面，使得类会被认定为Controller对象
    - 用于告知Spring容器，该类所有方法的返回值需要以JSON格式写到Response的Body内。

4. Spring MVC

- SpringMVC的HTTP请求方法注解

注解	HTTP请求方法
@GetMapping	POST
@PostMapping	GET
@PutMapping	PUT
@PatchMapping	PATCH
@DeleteMapping	DELETE
@RequestMapping	可用于以上五种请求，需在method属性中指定

5. 参数合法性检查

- JSR303 是一套JavaBean参数校验的标准，它定义了很多常用的校验注解，我们可以直接将这些注解加在我们JavaBean的属性上面(面向注解编程的时代)，就可以在需要校验的时候进行校验

5. 参数合法性检查

注解	说明
@AssertFalse/@AssertTrue	验证注解的元素值是false/true
@NotNull/@Null	验证注解的元素值不是空/是空
@NotBlank	应用于字符串，验证元素不为空或空格
@Max(value)/@Min(value)	验证注解的元素大于/小于值（value）
@Past/@Future	验证注解的元素比当前日期早/晚
@Email	验证注解的元素是Email

6. 跨域访问问题（CORS）

6. 跨域访问问题（CORS）

- 允许浏览器向跨源服务器，发出HTTPRequest请求。
  - 预检请求：浏览器先询问服务器，当前网页所在的域名是否在服务器的许可名单之中，以及可以使用哪些HTTP动词和头信息字段。只有得到肯定答复，浏览器才会发出正式的HTTPRequest请求，否则就报错。

```
OPTIONS /cors HTTP/1.1
Origin: http://api.bob.com
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: X-Custom-Header
Host: api.alice.com
Accept-Language: en-US
Connection: keep-alive
User-Agent: Mozilla/5.0...
```

6. 跨域访问问题（CORS）

- 服务器收到"预检"请求以后，检查了Origin、Access-Control-Request-Method和Access-Control-Request-Headers字段以后，确认允许跨源请求，就可以做出回应。

```
HTTP/1.1 200 OK
Date: Mon, 01 Dec 2008 01:15:39 GMT
Server: Apache/2.0.61 (Unix)
Access-Control-Allow-Origin: http://api.bob.com
Access-Control-Allow-Methods: GET, POST, PUT
Access-Control-Allow-Headers: X-Custom-Header
Content-Type: text/html; charset=utf-8
Content-Encoding: gzip
Content-Length: 0
Keep-Alive: timeout=2, max=100
Connection: Keep-Alive
Content-Type: text/plain
```

6. 跨域访问问题（CORS）

- 服务器通过了"预检"请求，以后每次浏览器正常的CORS请求，会有一个Origin头信息字段。服务器的回应，也都会有一个Access-Control-Allow-Origin头信息字段。

```
PUT /cors HTTP/1.1
Origin: http://api.bob.com
Host: api.alice.com
X-Custom-Header: value
Accept-Language: en-US
Connection: keep-alive
User-Agent: Mozilla/5.0...
```

```
Access-Control-Allow-Origin: http://api.bob.com
Content-Type: text/html; charset=utf-8
```

6. 跨域访问问题（CORS）

```
public interface WebMvcConfigurer {
    void configurePathMatch(PathMatchConfigurer var1);
    void configureContentNegotiation(ContentNegotiationConfigurer var1);
    void configureAsyncSupport(AsyncSupportConfigurer var1);
    void configureDefaultServletHandling(DefaultServletHandlerConfigurer var1);
    void addFormatters(FormatterRegistry var1);
    void addInterceptors(InterceptorRegistry var1);
    void addResourceHandlers(ResourceHandlerRegistry var1);
    void addCorsMappings(CorsRegistry var1);
    void addViewControllers(ViewControllerRegistry var1);
    void configureViewResolvers(ViewResolverRegistry var1);
    void addArgumentResolvers(List<HandlerMethodArgumentResolver> var1);
    void addReturnValueHandlers(List<HandlerMethodReturnValueHandler> var1);
    void configureMessageConverters(List<HttpMessageConverter<?>> var1);
    void extendMessageConverters(List<HttpMessageConverter<?>> var1);
    void configureHandlerExceptionResolvers(List<HandlerExceptionResolver> var1);
    void extendHandlerExceptionResolvers(List<HandlerExceptionResolver> var1);
    Validator getValidator();
     MessageCodesResolver getMessageCodesResolver();
}
```

6. 跨域访问问题（CORS）

- 跨域问题的解决方案

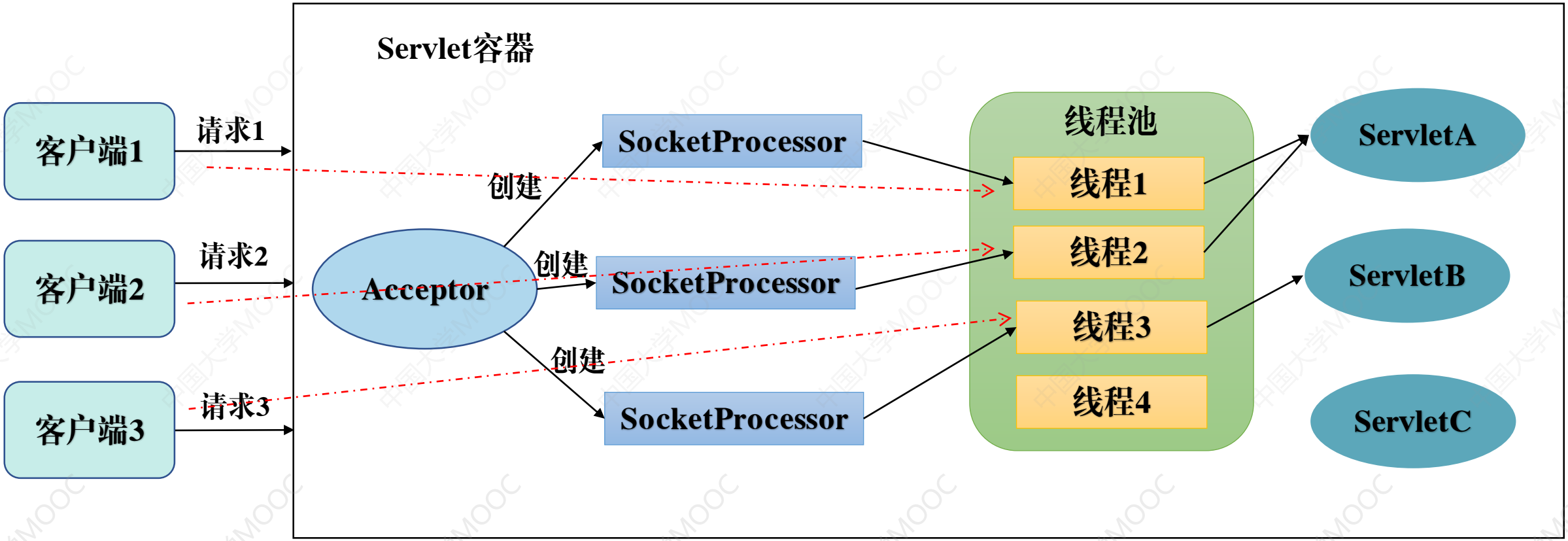
```
@Configuration
public class CorsConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {

  /**addMapping：配置可以被跨域的路径，可以任意配置，可以具体到直接请求路径。
     allowedMethods：允许所有的请求方法访问该跨域资源服务器，如：POST、GET、PUT、DELETE等。
     allowedOrigins：允许所有的请求域名访问我们的跨域资源，可以固定单条或者多条内容，如："http://www.aaa.com"，只有该域名可以访问我们的跨域资源。
     allowedHeaders：允许所有的请求header访问，可以自定义设置任意请求头信息。
  **/
        registry.addMapping("/**")
                .allowedOrigins("*")
                .allowedHeaders("*")
                .allowedMethods("GET", "POST", "DELETE", "PUT", "OPTIONS", "HEAD")
                .maxAge(3600);
    }
}
```

7. Servlet并发机制

- 线程模型 - BIO



7. Servlet并发机制

- 线程模型 - NIO