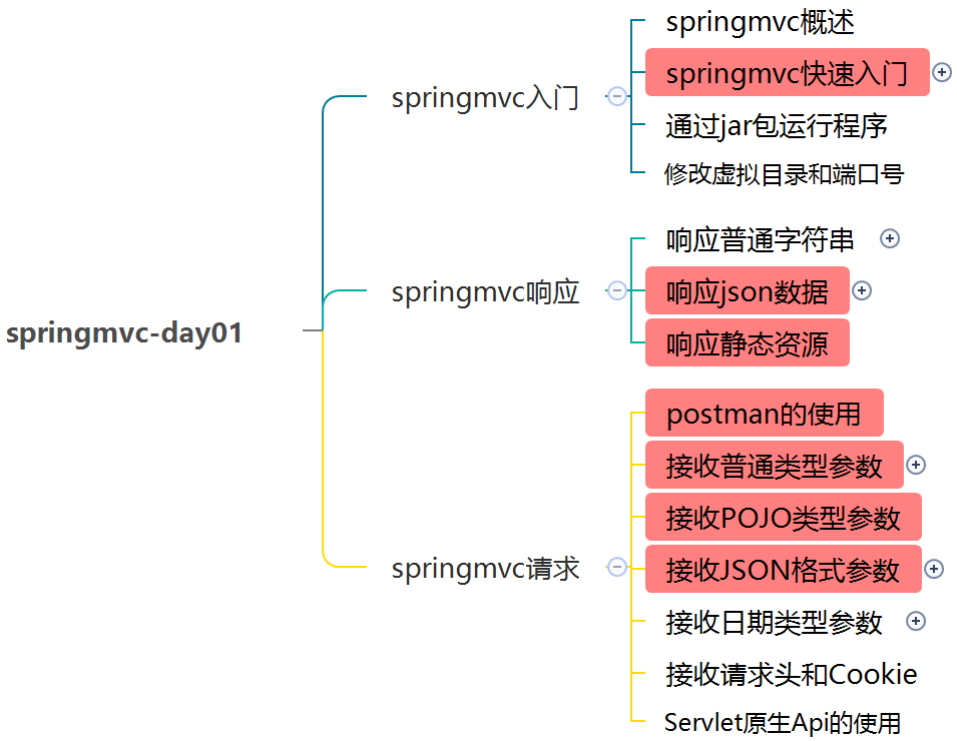
SpringMVC day01

目录

Contents


springmvc-day01
springmvc入门
springmvc概述
springmvc快速入门
通过jar包运行程序
修改虚拟目录和端口号
springmvc响应
响应普通字符串
响应json数据
响应静态资源
springmvc请求
postman的使用
接收普通类型参数
接收POJO类型参数
接收JSON格式参数
接收日期类型参数
接收请求头和Cookie
Servlet原生Api的使用

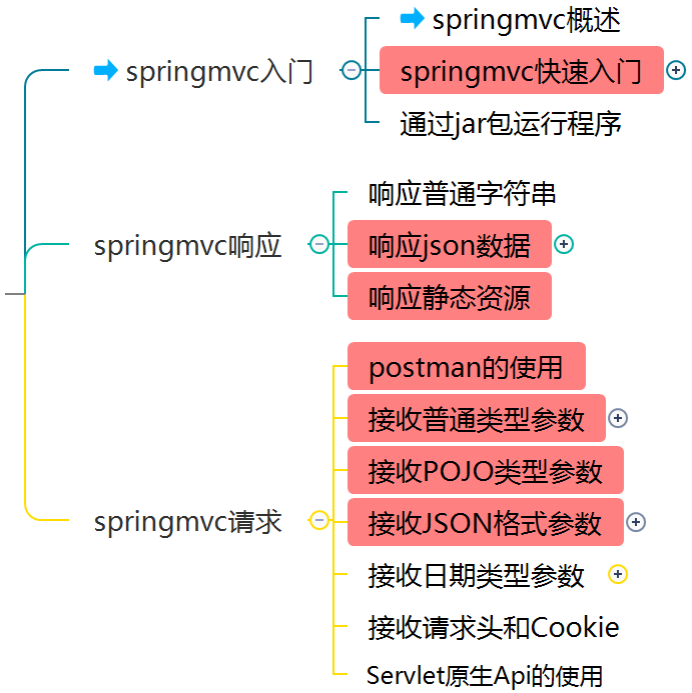
SpringMVC简介

- springmvc入门
  - springmvc概述
  - springmvc快速入门
  - 通过jar包运行程序
- springmvc响应
  - 响应普通字符串
  - 响应json数据
  - 响应静态资源
- springmvc请求
  - postman的使用
  - 接收普通类型参数
  - 接收POJO类型参数
  - 接收JSON格式参数
  - 接收日期类型参数
  - 接收请求头和Cookie
  - Servlet原生Api的使用

三层架构

- 表现层：负责请求处理和数据展示
- 业务层：负责业务处理
- 数据层：负责数据操作

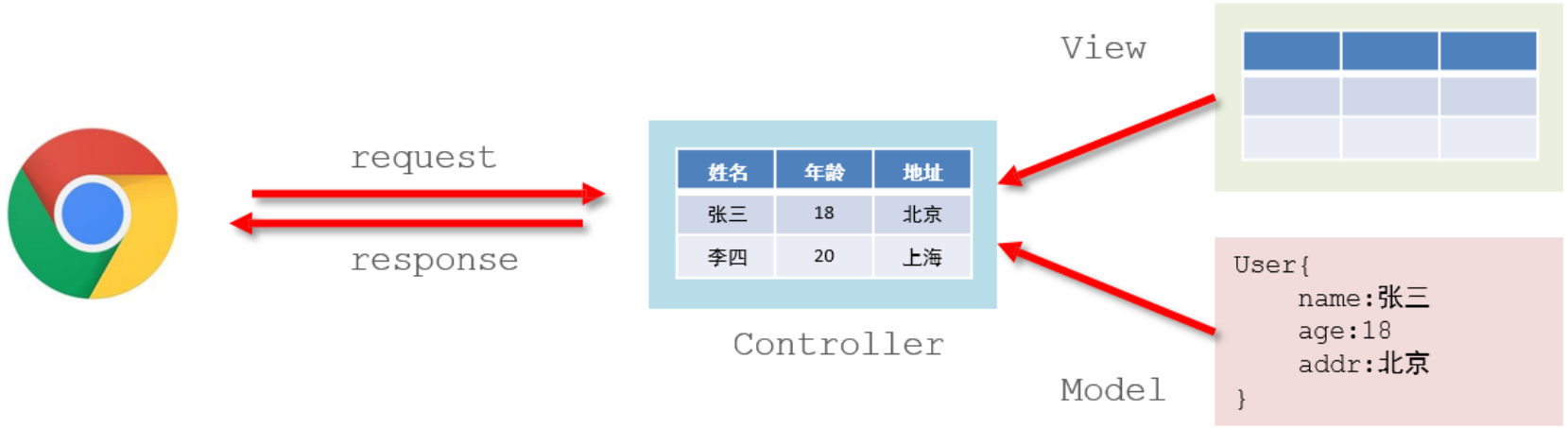
SpringMVC简介



MVC

MVC（Model View Controller），一种用于设计创建Web应用程序的开发模式

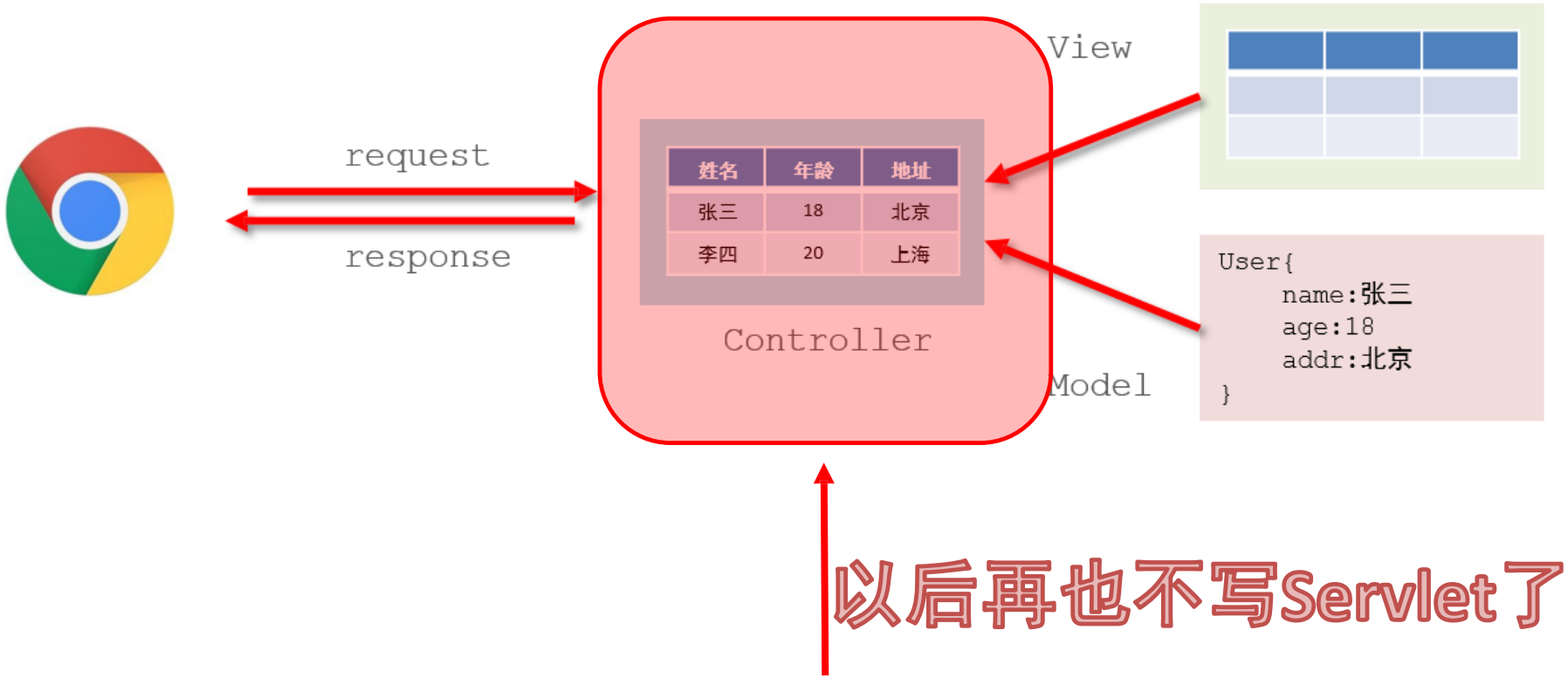
- Model（模型）：数据模型，用于数据处理，包括(service和dao)
- View（视图）：页面视图，用于展示数据
  - jsp
  - html
- Controller（控制器）：处理用户发送的请求，调用model完成数据处理，并对view视图进行响应
  - Servlet



SpringMVC简介

- springmvc入门
  - springmvc概述
  - springmvc快速入门
  - 通过jar包运行程序
- springmvc响应
  - 响应普通字符串
  - 响应json数据
  - 响应静态资源
- springmvc请求
  - postman的使用
  - 接收普通类型参数
  - 接收POJO类型参数
  - 接收JSON格式参数
  - 接收日期类型参数
  - 接收请求头和Cookie
  - Servlet原生Api的使用

Spring MVC 是 Spring 提供的一个**基于 MVC 设计模式**的轻量级 Web 开发框架，**本质上相当于 Servlet**。



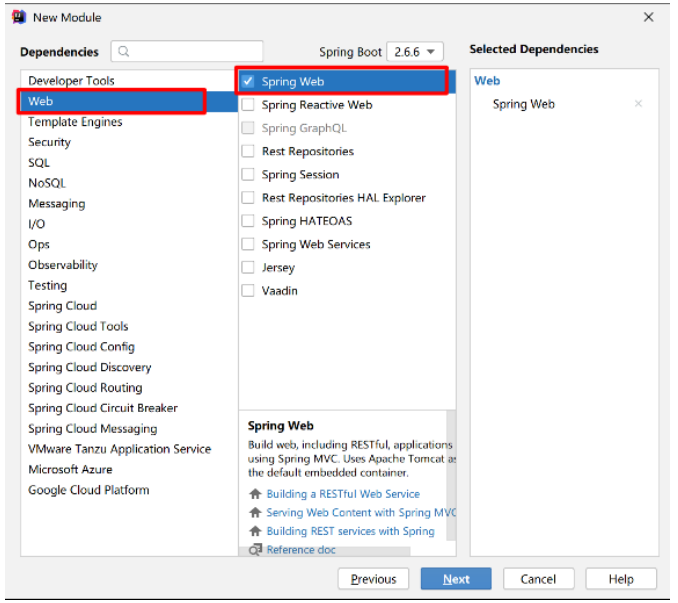
以后再也不写Servlet了

Spring MVC

快速入门

- springmvc入门
  - springmvc概述
  - springmvc快速入门
    - @Controller
    - @RequestMapping
  - 通过jar包运行程序
- springmvc响应
  - 响应普通字符串
  - 响应json数据
  - 响应静态资源
- springmvc请求
  - postman的使用
  - 接收普通类型参数
  - 接收POJO类型参数
  - 接收JSON格式参数
  - 接收日期类型参数
  - 接收请求头和Cookie
  - Servlet原生Api的使用

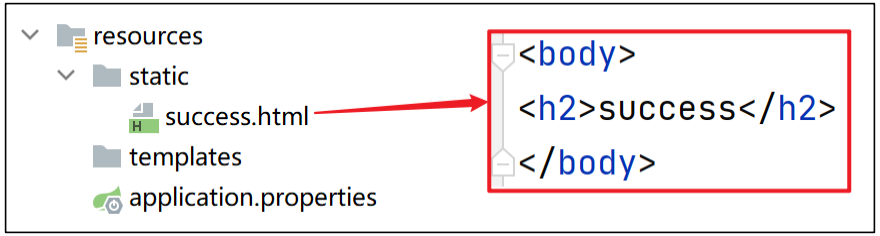
步骤1：创建模块，打包方式选择 jar，勾选 Spring Web 支持



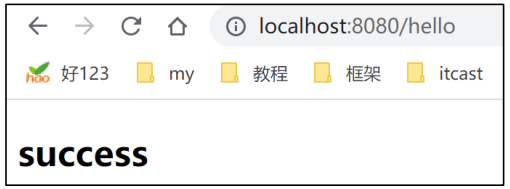
步骤2：编写控制器，注意位置要在扫描范围内

```java
@Controller
public class UserController {
    /**
     * springmvc快速入门方法
     * @return 默认表示要跳转的页面
     */
    @RequestMapping("/hello")//定义访问路径
    public String hello(){
        System.out.println("hello springmvc...");
        return "/success.html";
    }
}
```

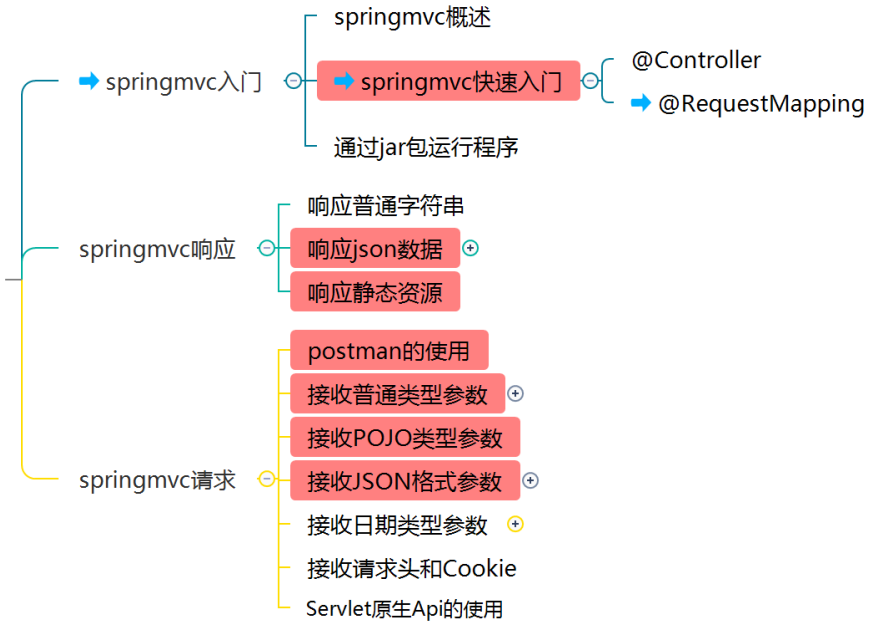
步骤3：在static目录中创建success.html页面



步骤4：运行引导类，打开浏览器，输入如下地址访问控制器方法

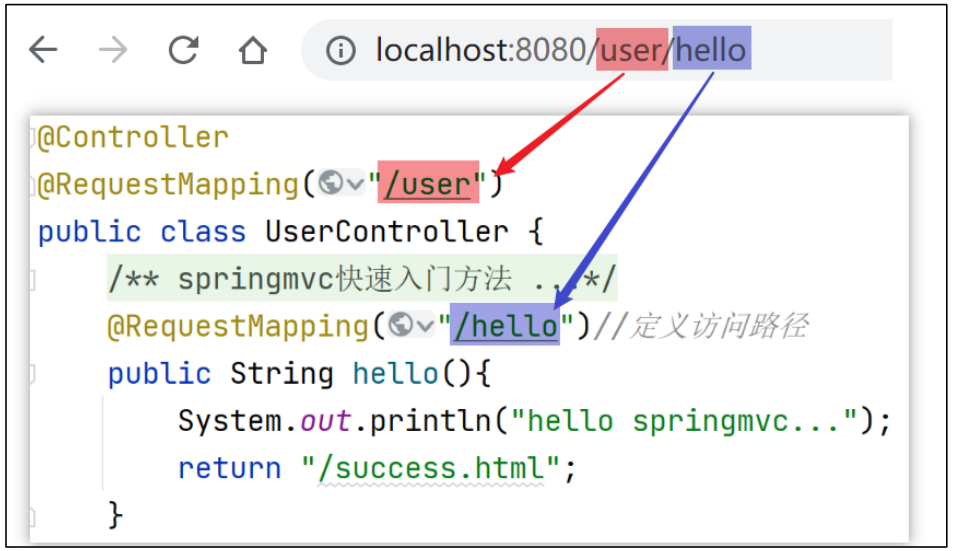


快速入门-@RequestMapping



@RequestMapping 用来定义请求路径

- 除了可以加在方法上以外，还可以同时加在类上
- 如果类和方法都加了 @RequestMapping，那么最终的请求路径由二者共同决定



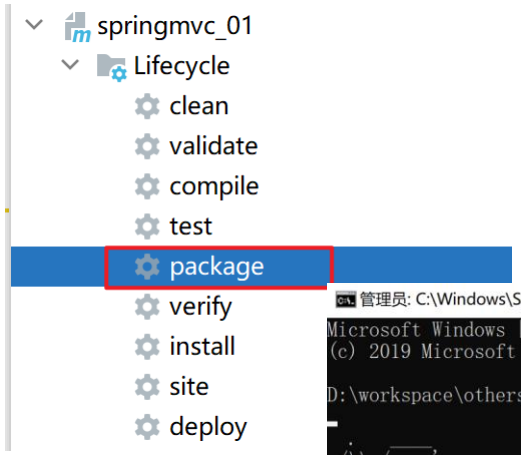
@RequestMapper注解可以定义如下属性：

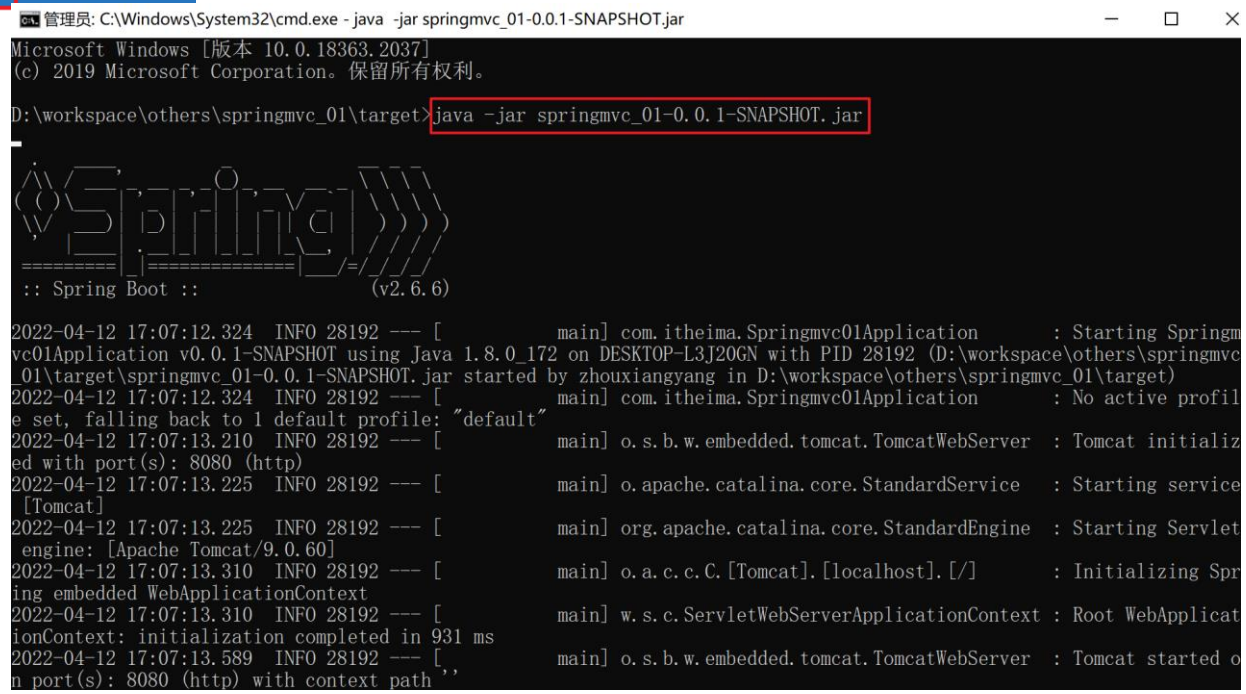
```
@RequestMapping(
    value="/hello", //设定请求路径，与path属性、 value属性相同
    method = RequestMethod.GET, //设定请求方式
    params = "name", //设定请求参数条件
    headers = "content-type=text/*", //设定请求消息头条件
    consumes = "text/*", //用于指定可以接收的请求正文类型（MIME类型）
    produces = "text/*" //用于指定可以生成的响应正文类型（MIME类型）
)
```

快速入门-通过jar包运行程序

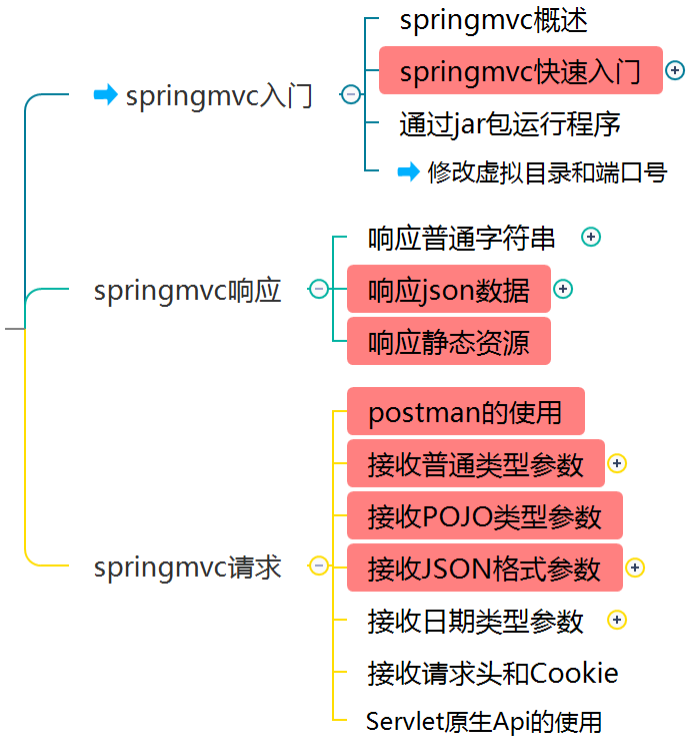
- springmvc入门
  - springmvc概述
  - springmvc快速入门
  - 通过jar包运行程序
  - 修改虚拟目录和端口号
- springmvc响应
  - 响应普通字符串
  - 响应json数据
  - 响应静态资源
- springmvc请求
  - postman的使用
  - 接收普通类型参数
  - 接收POJO类型参数
  - 接收JSON格式参数
  - 接收日期类型参数
  - 接收请求头和Cookie
  - Servlet原生Api的使用

- SpringBoot程序默认打jar包，当然也可以打war包，官方推荐打jar包。SpringBoot程序jar包中内置了web服务器，可以直接运行jar包并访问程序。
- 命令：java -jar jar包的名称





快速入门-修改虚拟目录和端口号



- 可以修改web程序虚拟目录和端口号

application.properties

```
# 设置虚拟目录
server.servlet.context-path=/springmvc_01
# 设置端口号
server.port=8080
```

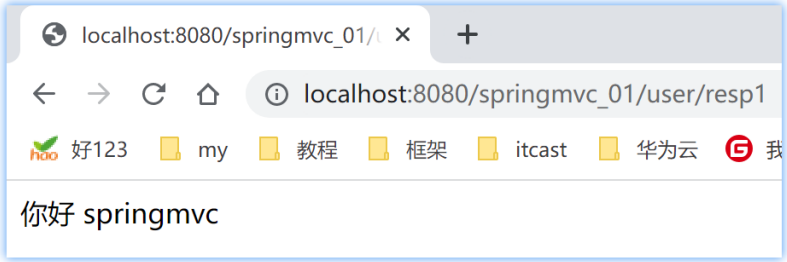
application.yml

```
server:
  servlet:
    context-path: springmvc_01 # 设置虚拟目录
  port: 8080 # 设置端口号
```

响应普通字符串-@ResponseBody

- 问题：如果Controller中方法不想进行页面跳转，需要响应数据该怎么办？
- 解决：在方法上使用@ResponseBody注解，表示该方法不进行页面跳转，将返回值通过响应体响应给客户端

```
/**
 * 演示响应普通字符串
 * @return
 */
@RequestMapping("/resp1")
@ResponseBody
public String resp1(){
    return "你好 springmvc";
}
```

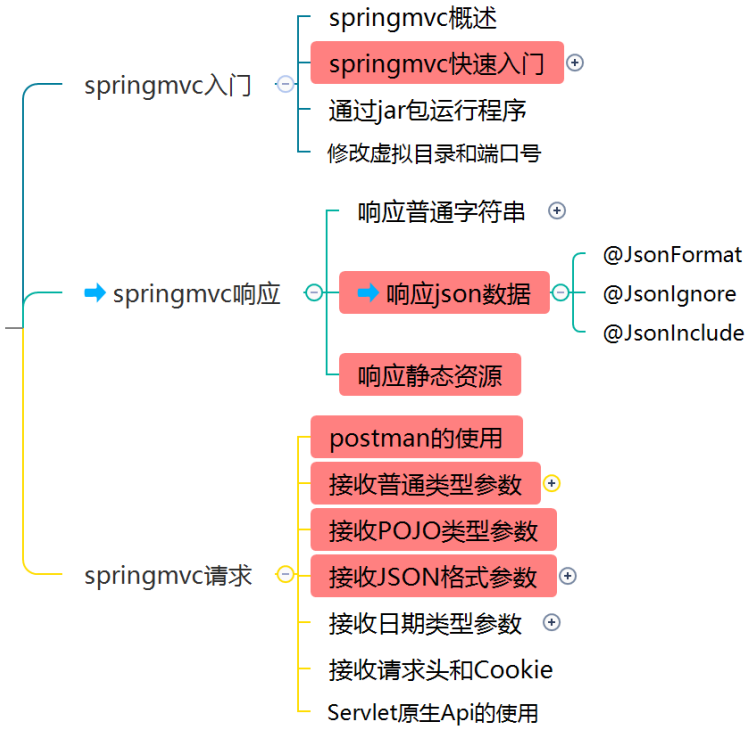


springmvc_01是虚拟目录，user是Controller类上定义的一级路径

- @ResponseBody注解还可以写在Controller类上，表示该类的所有方法都不进行页面跳转，而是响应数据

说明：@RestController 是一个组合注解，同时含有 @Controller 与 @ResponseBody功能

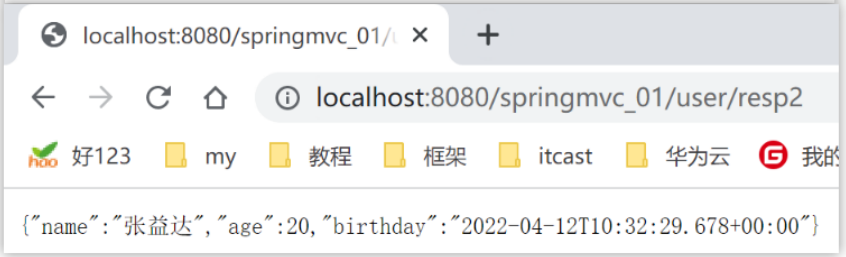
响应json数据



- Springmvc底层集成了jackson工具，能够自动将java对象转换成json响应给客户端

```
@Data
@NoArgsConstructor
@AllArgsConstructor
public class User {
    private String name;
    private Integer age;
    private Date birthday;
}
```

```
/** 演示响应json数据 ...*/
@RequestMapping("/resp2")
@ResponseBody
public User resp2(){
    //说明：将来是调用service层方法，查询到一个User对象
    User user=new User( name: "张益达", age: 20,new Date());
    return user;
}
```



- 转换json的细节：日期格式问题？日期时区问题？如果不想要该属性被转换到json中？

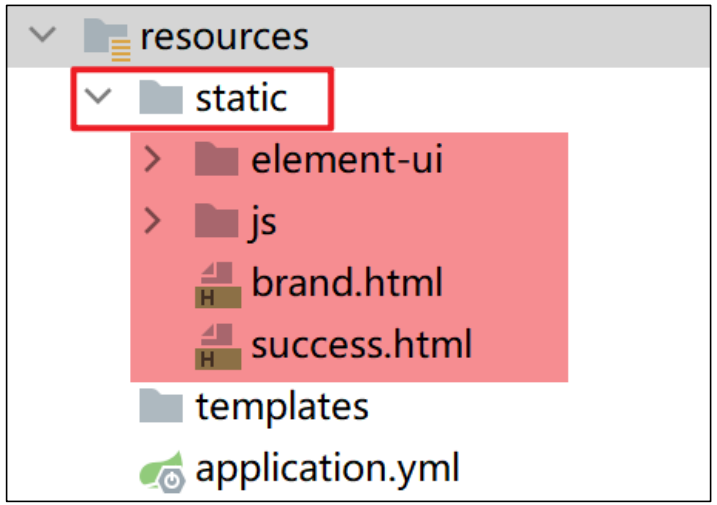
@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss",timezone="Asia/Shanghai")
作用：格式化，pattern指定格式化模式，timezone指定日期时区。

@JsonIgnore：忽略该属性，不转换到json中

@JsonInclude(JsonInclude.Include.NON_NULL)
作用：该属性值非空才会被转换到json中

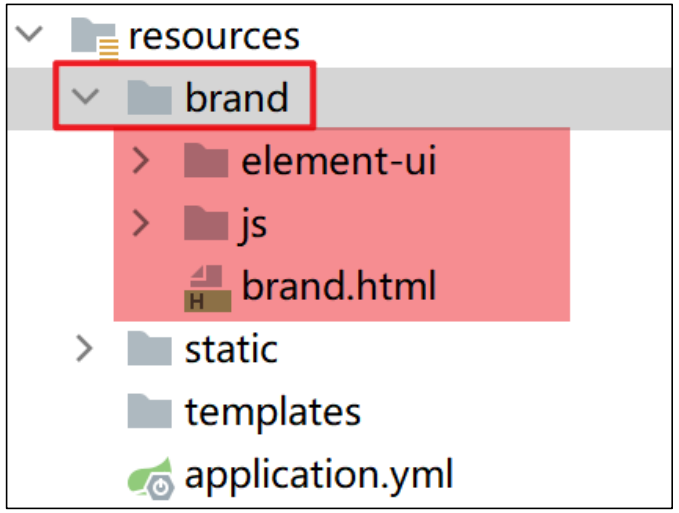
响应静态资源

➢ **静态资源**放在如下目录，能够被Spring Boot所自动找到




```
src
    |- main
        |- java
        |- resources
            |- static (此处)
            |- public (此处)
            |- resources (此处)
            |- META-INF
                |- resources (此处)
```

如果不想采用默认位置，SpringBoot是否能自动找到静态资源？ **不能**



响应静态资源

解决：定义配置类，实现WebMvcConfigurer接口配置静态资源访问路径

```
@SpringBootApplication
public class Demo2Application implements WebMvcConfigurer {

    public static void main(String[] args) {
        SpringApplication.run(Demo2Application.class, args);
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler("【url路径】").addResourceLocations("【程序路径】");
    }
}
```

示例：

```
@SpringBootApplication
public class Springmvc01Application implements WebMvcConfigurer {
    public static void main(String[] args) {
        SpringApplication.run(Springmvc01Application.class, args);
    }
    //重写addResourceHandlers方法，配置静态资源访问路径
    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry) {
        registry.addResourceHandler( ...pathPatterns: "/brand/**").addResourceLocations("classpath:/brand/");
    }
}
```

请求-postman的使用

- springmvc入门
  - springmvc概述
  - springmvc快速入门
  - 通过jar包运行程序
- springmvc响应
  - 响应普通字符串
  - 响应json数据
  - 响应静态资源
- springmvc请求
  - postman的使用
  - 接收普通类型参数
  - 接收POJO类型参数
  - 接收JSON格式参数
  - 接收日期类型参数
  - 接收请求头和Cookie
  - Servlet原生Api的使用

安装资料中提供的Postman-win64-7.14.0-Setup.exe，使用参考资料中的“postman的基本使用.zip”

Sign Up with Google

OR

email

username

password

Sign Up (it's free)

By signing up you agree to the EULA

Already have an account? Sign In

Take me straight to the app. I'll create an account another time.

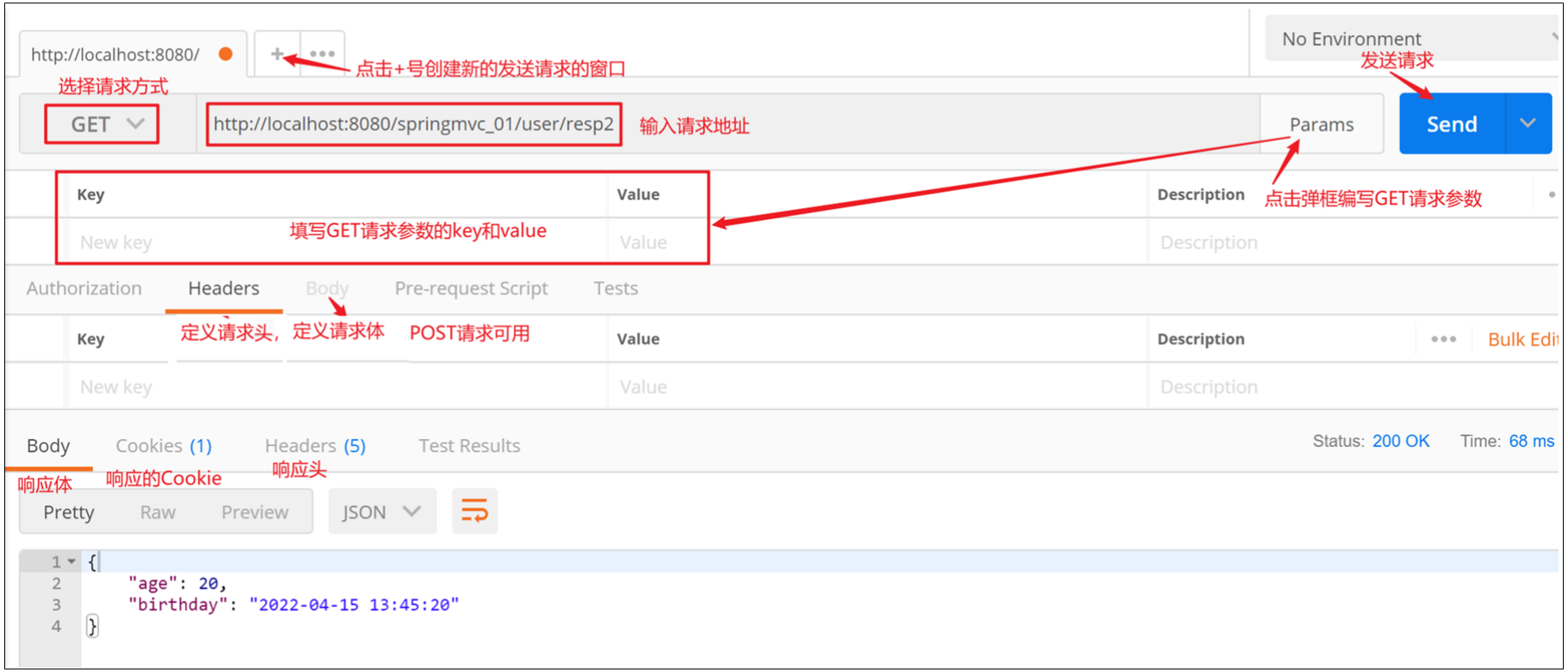
点击蓝色链接，跳过注册登录

Why Sign Up

- Organize all your API development within Postman Workspaces
- Sync your Postman data across devices
- Backup your Postman data
- Create Documentation pages, Monitors and Mock Servers

请求-postman的使用

Postman功能介绍：



请求-接收普通类型参数

- postman发送请求:

- controller接收请求参数

```
@RequestMapping("/req1")
public String req1(String name,Integer age){
    log.info("name={}",name);
    log.info("age={}",age);
    return "req1接收请求参数成功!";
}
```

- 注意：controller方法中的形参名要和请求参数名称一样才可以自动赋值

请求-接收普通类型参数

- controller接收请求参数（形参名和请求参数名称不一样）

```
@RequestMapping("/req1")
public String req1(@RequestParam(value = "name") String username, Integer age){
    log.info("username={}",username);
    log.info("age={}",age);
    return "req1接收请求参数成功!";
}
```

value的属性值要和请求参数名称一样，表示将请求参数name的值赋值给username变量

@RequestParam注解介绍：

```
public @interface RequestParam {
    @AliasFor("name")
    String value() default ""; 请求参数名称

    @AliasFor("value")
    String name() default ""; 和value作用一样

    boolean required() default true; 请求是否必须携带该参数

                                    请求中没有该参数时赋默认值
    String defaultValue() default "\n\t\t\n\t\t\n\ue000
}
```

请求-接收POJO类型参数

- postman发送请求:

- controller接收请求参数

```java
@Data
public class Person {
    private String name;
    private Integer age;
    private String[] hobby;
}
```

```java
@RequestMapping("/req2")
public String req2(Person person){
    log.info("person={}",person);
    return "req2接收请求参数成功!";
}
```

- 注意：javabean属性的名称要和请求参数的名称一样

请求-接收JSON格式参数

- springmvc入门
  - springmvc概述
  - springmvc快速入门
  - 通过jar包运行程序
- springmvc响应
  - 响应普通字符串
  - 响应json数据
  - 响应静态资源
- springmvc请求
  - postman的使用
  - 接收普通类型参数
  - 接收POJO类型参数
  - 接收JSON格式参数 — @RequestBody
  - 接收日期类型参数
  - 接收请求头和Cookie
  - Servlet原生Api的使用

➢ postman发送请求:

➢ controller接收请求参数

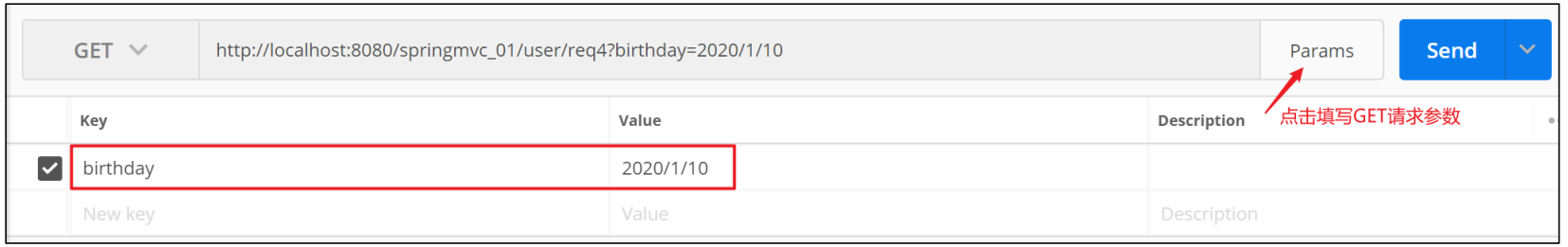
```java
@Data
public class Person {
    private String name;
    private Integer age;
    private String[] hobby;
}
```

```java
@RequestMapping("/req3")
public String req3(@RequestBody Person person){
    log.info("person={}",person);
    return "req3接收请求参数成功!";
}
```

➢ @RequestBody：将请求体中的JSON数据封装成java对象

请求-接收日期类型参数

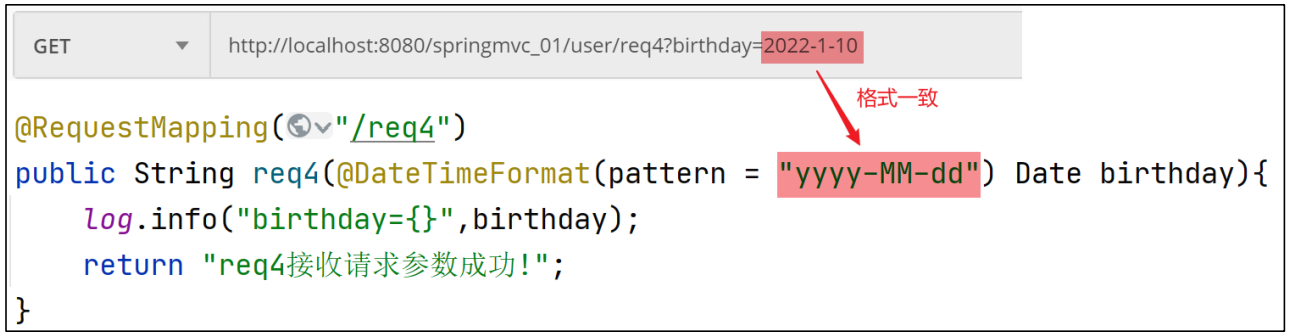
- postman发送请求:



- controller接收请求参数

```
@RequestMapping("/req4")
public String req4(Date birthday){
    log.info("birthday={}",birthday);
    return "req4接收请求参数成功!";
}
```

- 注意：springmvc默认支持转换的日期格式是yyyy/MM/dd。如果发送的日期格式是yyyy-MM-dd，那么后台方法参数需要使用@DateTimeFormat注解指定格式化模式。




```
@RequestMapping("/req4")
public String req4(@DateTimeFormat(pattern = "yyyy-MM-dd") Date birthday){
    log.info("birthday={}",birthday);
    return "req4接收请求参数成功!";
}
```

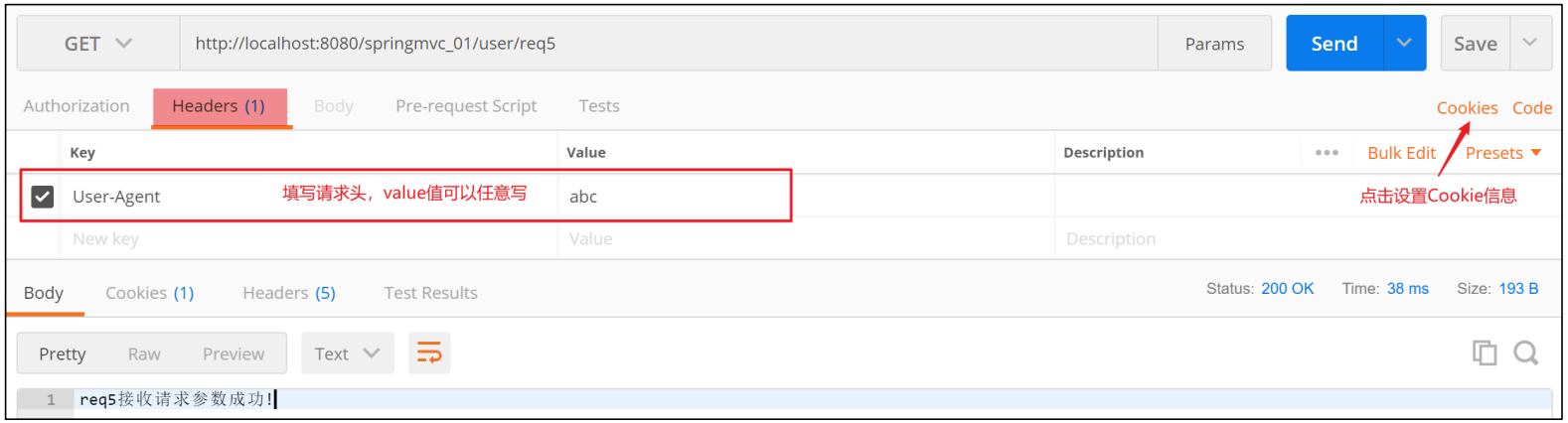
- 如果封装到POJO中，需要在属性上加@DateTimeFormat注解

```
public class User {
    private String name;
    private Integer age;
    @DateTimeFormat(pattern = "yyyy-MM-dd")
    private Date birthday;
}
```

请求-接收请求头和Cookie

- springmvc入门
  - springmvc概述
  - springmvc快速入门
  - 通过jar包运行程序
- springmvc响应
  - 响应普通字符串
  - 响应json数据
  - 响应静态资源
- springmvc请求
  - postman的使用
  - 接收普通类型参数
  - 接收POJO类型参数
  - 接收JSON格式参数
  - 接收日期类型参数
  - 接收请求头和Cookie
  - Servlet原生Api的使用

➢ postman发送请求:

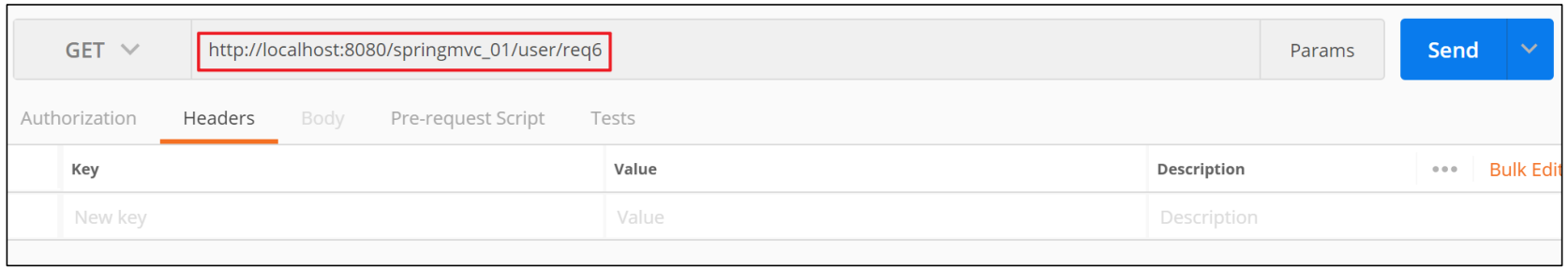


➢ controller接收请求参数

```
@RequestMapping("/req5")                    注解属性值为请求头名称
public String req5(@RequestHeader("user-agent") String userAgent,
                   @CookieValue("username") String username){
    log.info("userAgent={}",userAgent);     注解属性值为cookie名称
    log.info("username={}",username);
    return "req5接收请求参数成功!";
}
```

请求-Servlet原生Api的使用

- postman发送请求:



- controller接收请求参数

```
@RequestMapping("/req6")
public String req6(HttpServletRequest request, HttpSession session){
    request.setAttribute( s: "msg", o: "hello request");
    session.setAttribute( s: "msg", o: "hello session");
    return "req6接收请求参数成功!";
}
```

需要使用到Servlet原生API，直接声明就行了。

```
@RequestMapping("/req7")
public String req7(HttpServletRequest request, HttpSession session){
    log.info("request域中的msg数据：{}",request.getAttribute( s: "msg"));
    log.info("session域中的msg数据：{}",session.getAttribute( s: "msg"));
    return "req7接收请求参数成功!";
}
```

传智教育旗下高端IT教育品牌