

Java后端开发

1、Java包管理Maven

maven仓库的依赖管理，

1 | <https://mvnrepository.com>

对应的黑马视频：[Day04-01. maven-课程介绍哔哩哔哩bilibili](#)

2、如何创建JavaSpring项目？

3、Java Web服务器后端接收参数：

1. 原始方式获取请求参数

- Controller方法形参中声明HttpServletRequest对象
- 调用对象的getParameter(参数名)

2. SpringBoot中接收简单参数

- 请求参数名与方法形参变量名相同
- 会自动进行类型转换

3. @RequestParam注解

- 方法形参名称与请求参数名称不匹配，通过该注解完成映射
- 该注解的required属性默认是true，代表请求参数必须传递

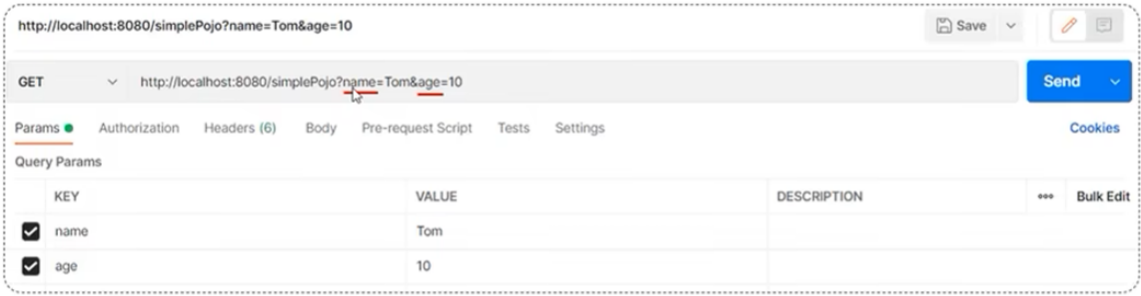
请求参数接收：

当要接收的参数很多时，创建对应的实体类：

1、简单实体类创建：

实体参数

- 简单实体对象：请求参数名与形参对象属性名相同，定义POJO接收即可



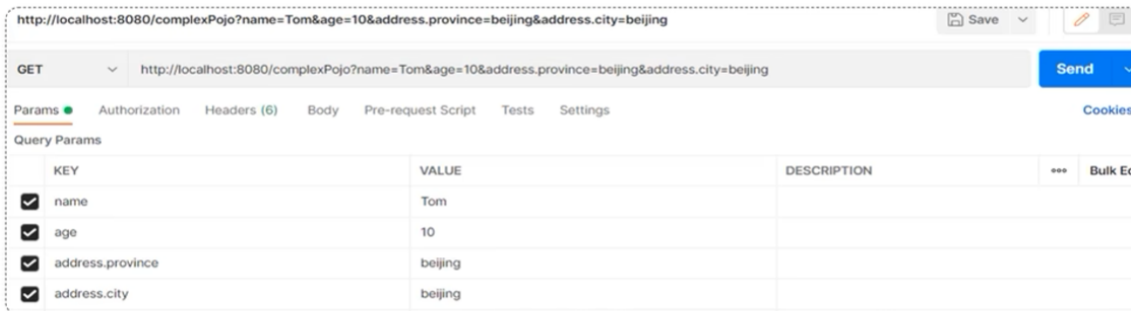
```
@RequestMapping("/simplePojo")
public String simplePojo(User user){
    System.out.println(user);
    return "OK";
}
```

```
public class User {
    private String name;
    private Integer age;
}
```

2、复杂实体类创建：

实体参数

- 复杂实体对象：请求参数名与形参对象属性名相同，按照对象层次结构关系即可接收嵌套POJO属性参数。



```
@RequestMapping("/complexPojo")
public String complexPojo(User user){
    System.out.println(user);
    return "OK";
}
```

```
public class User {
    private String name;
    private Integer age;
    private Address address;
}
```

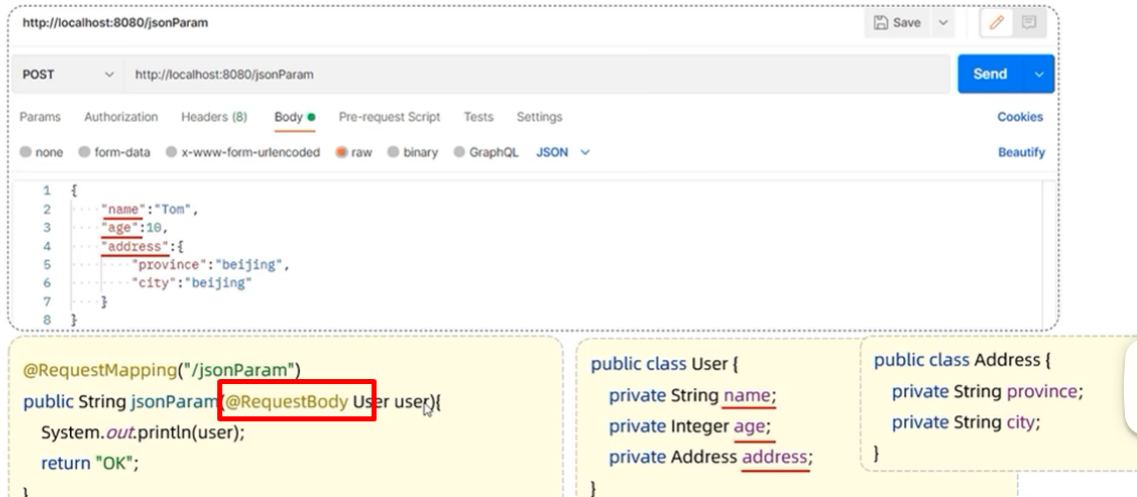
```
public class Address {
    private String province;
    private String city;
}
```

3、接收json数据；

要添加RequestBody;

JSON 参数

- JSON参数：JSON数据键名与形参对象属性名相同，定义POJO类型形参即可接收参数，需要使用 `@RequestBody`



```
POST http://localhost:8080/jsonParam

{
  "name": "Tom",
  "age": 10,
  "address": {
    "province": "beijing",
    "city": "beijing"
  }
}
```

```
@RequestMapping("/jsonParam")
public String jsonParam(@RequestBody User user){
    System.out.println(user);
    return "OK";
}
```

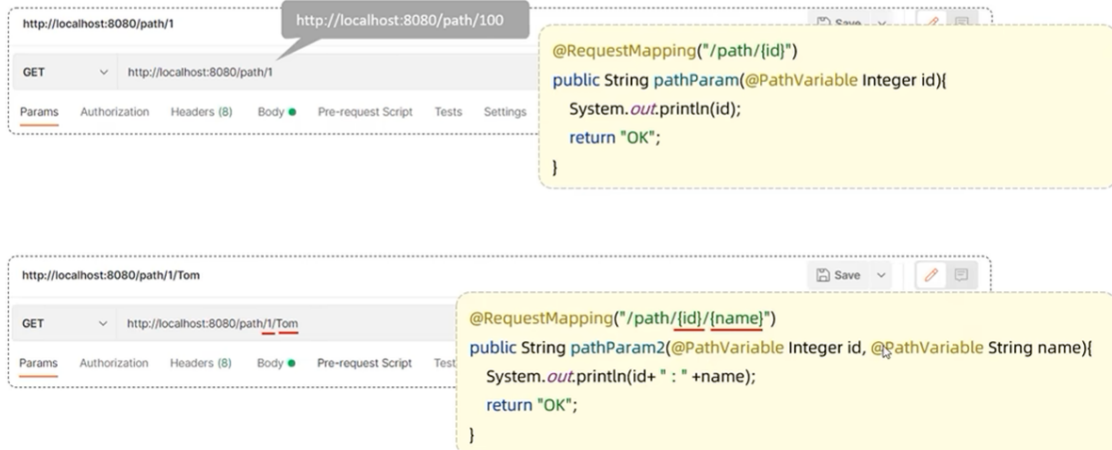
```
public class User {
    private String name;
    private Integer age;
    private Address address;
}
```

```
public class Address {
    private String province;
    private String city;
}
```

4、接收路径参数：

路径参数

- 路径参数：通过请求URL直接传递参数，使用[...]来标识该路径参数，需要使用 `@PathVariable` 获取路径参数



```
GET http://localhost:8080/path/1

@RequestMapping("/path/{id}")
public String pathParam(@PathVariable Integer id){
    System.out.println(id);
    return "OK";
}
```

```
GET http://localhost:8080/path/1/Tom

@RequestMapping("/path/{id}/{name}")
public String pathParam2(@PathVariable Integer id, @PathVariable String name){
    System.out.println(id + " : " + name);
    return "OK";
}
```

RestController注释包括：

- Controller;
- ResponseBody;

统一响应结果：

Result(code, message, data);

三层架构

控制类中不能同时包括：数据访问，逻辑处理，响应数据。

[Day05-09. 分层解耦-三层架构哔哩哔哩bilibili](#)

三层架构



- controller: 控制层, 接收前端发送的请求, 对请求进行处理, 并响应数据。
- service: 业务逻辑层, 处理具体的业务逻辑。
- dao: 数据访问层(Data Access Object) (持久层), 负责数据访问操作, 包括数据的增、删、改、查。

为什么要增强DAO的灵活性和可扩展性? 具体怎么做?

分层这块代码, 需要好好理解下。

这个代码需要好好实践下。

什么是控制反转?

对象的创建交由容器负责, 不需要在类内部处理。具体代码上, 只需要在class类声明前, 加上如下的关键词:

```
1 @Component
2 public class EmpService implements EmpService{
3
4     @Autowired
5     private EmpDao empDao;
6     ...
7 }
```

什么是依赖注入?

在对象的类成员变量前, 加上autowired关键词:

```
1 @Componentpublic
2 class EmpService implements EmpService{
3     @Autowired
4     private EmpDao empDao;
5     ...
6 }
```

4、MyBatis持久层框架

