

原创 推荐

Spring MVC 传值方式总结

 [cnn237111](#) [关注](#)

2017-02-01 16:01:5018321人阅读1人评论

前端传到Controller:

方法1

通过HttpServletRequest 。写法如下:

```
@Controller
public class MyTestController {
    @RequestMapping("/print")
    public String PrintInfo(HttpServletRequest request) {
        System.out.println("name:" +request.getParameter("name"));
        System.out.println("age:" + request.getParameter("age"));
        return "testpage";
    }
}
```

HttpServletRequest类是Servlet中的类型，代表了一个Servlet请求。无论Post还是Get请求，都能通过这种方式获取到。



比如上面的代码，通过Get方法，如下地址

<http://127.0.0.1:8080/WebApp/print?name=zhangsan&age=30>

也可以通过Post方法，使用Postman工具模拟一个post请求，都可以将值传到Controller。

这招可以获得Cookie以及Session数据。

还可以通过注解@Autowired，将HttpServletRequest 自动的注入进来，不必担心多线程下的并发问题，因为这里HttpServletRequest注入的是一个AOP proxy，而不是一个普通bean。每次请求过来，都会检查线程本地属性，来获取真正的Request对象。这些都是Spring自动配置的默认场景。可以参阅<https://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html#beans-factory-scopes-other-injection>

但是不推荐使用这个方法，因为这种方法破坏了对一个注入对象的常规理解，造成混乱。

代码如下:

```
@Controller
public class MyTestController {
    @Autowired
    private HttpServletRequest request;
    @RequestMapping(value="/print")
    public String PrintInfo() {
        System.out.println("name:" +request.getParameter("name"));
        System.out.println("age:" + request.getParameter("age"));
        return "testpage";
    }
}
```

在线客服

使用路径变量。写法如下：

```
@Controller
public class MyTestController {
    @RequestMapping("/print/{name}/{age}")
    public String PrintInfo(@PathVariable String name, @PathVariable int age) {
        System.out.println("name:" + name);
        System.out.println("age:" + age);
        return "testpage";
    }
}
```

@RequestMapping中的{}中即为路径变量，该变量还需要在方法的参数值出现，并且标记@PathVariable。

通过URL匹配的方式既可以实现传值，这是REST风格的一种传值方式。

上面的例子，只需输入URL：

<http://127.0.0.1:8080/WebApp/print/ZhangSan/30>

controller接收到传值，输出：

name:ZhangSan

age:30

@RequestMapping("/print/{name}/{age}")是@RequestMapping(Value="/print/{name}/{age}")的缩写形式，本质上是一样的。

方法3

参数名匹配的方式：

```
@Controller
public class MyTestController {
    @RequestMapping(value="/print")
    public String PrintInfo(String name, int age) {
        System.out.println("name:" +name);
        System.out.println("age:" + age);
        return "testpage";
    }
}
```



或者：

```
@Controller
public class MyTestController {
    @RequestMapping(value="/print")
    public String PrintInfo(@RequestParam("name") String name,@RequestParam("age") int age) {
        System.out.println("name:" +name);
        System.out.println("age:" + age);
        return "testpage";
    }
}
```

当请求传入的参数名字和controller

中代码的名字一样的时候，两种方式都可以，区别在于使用了注解@RequestParam，可以设置一个默认值来处理到null值。

```
@RequestParam(value="name", defaultValue="John")
```

在线
客服



http://localhost:8080/WebApp/print?user_name=somename&user_age=30

Controller代码只能如下的写法

```
@RequestMapping(value="/print")
public String PrintInfo(@RequestParam("user_name") String name, @RequestParam("user_age")int a
...
}
```

尽量使用@RequestParam注解，因为这样可以清晰的知道该参数来自Request，可读性高。

方法4

传递请求头中的参数，需要用到@RequestHeader注解，该注解将Header中的值绑定到参数上，可以获取一个，多个或者所有的参数。例如

```
@Controller
public class MyTestController {
    @RequestMapping(value="/print")
    public String PrintInfo(@RequestHeader Map<String, String> headers) {
        for (String elem: headers.keySet()) {
            System.out.println(elem + ":" + headers.get(elem));
        }
        return "testpage";
    }
}
```

或者

```
@Controller
public class MyTestController {
    @RequestMapping(value="/print")
    public String PrintInfo(@RequestHeader("User-Agent") String userAgent) {
        System.out.println("12");
        System.out.println("name:" +userAgent);
        //System.out.println("age:" + age);
        return "testpage";
    }
}
```



方法5

使用到@RequestBody注解，得到整个RequestBody的信息

```
@Controller
public class MyTestController {
    @RequestMapping(value="/print")
    public String PrintInfo(@RequestBody String body) {
        System.out.println("body:" +body);
        return "testpage";
    }
}
```

@RequestBody可以将Json数据直接映射程Java对象。例如：

方法6

采用@ModelAttribute注解,命名匹配，Post中的参数值和Model中的参数值一致的话，会自动绑定到该值。

@Controller

3

4

1

分享



cnn237111

关注

在线客服

```
public String PrintInfo(@ModelAttribute User user) {
    System.out.println("6");
    System.out.println("Name:" +user.getName());
    System.out.println("Age:" +user.getAge());
    return "testpage";
}

public class User {
    private String name;
    private int age;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

然后当Post的值中有name和age时，Controller中的user对象会自动附上值。

Controller传递到JSP

方法1

使用ModelAndView类，代码如下：

```
@RequestMapping("/hello")
public ModelAndView showMessage() {
    ModelAndView mv = new ModelAndView("helloworld");
    mv.addObject("userList", GetUserList());
    return mv;
}

public List<User> GetUserList()
{
    List<User> lst=new ArrayList<User>();
    User user1=new User();
    user1.setName("zhangsan");
    user1.setAge(20);
    lst.add(user1);
    User user2=new User();
    user2.setName("lisi");
    user2.setAge(30);
    lst.add(user2);
    return lst;
}
```



JSP页面中：

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
```

在线
客服



```
</head>
<body>
<c:forEach items="${userList}" var="user">
${user.name} ${user.age}
<br />
</c:forEach>
</body>
</html>
```

ModelAndView 初始化的时候，设置了view的名字，同时也把对象存起来，直接传给view。简单实用。

方法2

使用Model或者ModelMap

(Model是一个接口，ModelMap实现了Model接口)

该方法和ModelAndView方法相似，只是Model和View分开来了，通过返回一个String来找到View，Model是注入到Controller的一个参数，通过对它添加属性，在jsp端读取值。代码如下：

```
@Controller
public class HelloWorldController {
    String message = "Welcome to Spring MVC!";
    @RequestMapping("/hello")
    public String showMessage(Model model) {
        model.addAttribute("userList", GetUserList());
        return "helloworld";
    }
    public List<User> GetUserList()
    {
        List<User> lst=new ArrayList<User>();
        User user1=new User();
        user1.setName("zhangsang");
        user1.setAge(10);
        lst.add(user1);
        User user2=new User();
        user2.setName("lisi");
        user2.setAge(33);
        lst.add(user2);
        return lst;
    }
}
```



JSP页面中：

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Spring 4 MVC -HelloWorld</title>
</head>
<body>
<c:forEach items="${userList}" var="user">
${user.name} ${user.age}
<br />
```

在线
客服



©著作权归作者所有：来自51CTO博客作者cnn237111的原创作品，如需转载，请注明出处，否则将追究法律责任

spring

mvc

Java

3

收藏

分享

上一篇：浮点数

下一篇：更新一下



cnn237111

554篇文章，260W+人气，9粉丝

关注



提问和评论都可以，用心的回复会被更多人看到和认可

Ctrl+Enter 发布


取消

发布



1条评论

按时间正序 | 按时间倒序



luminousman

1楼 2017-10-22 23:51:44

很详细，学到了不少。

推荐专栏

更多



基于Python的DevOps实战

自动化运维开发新概念

共20章 | 抚琴煮酒

¥ 51.00 378人订阅

订 阅



微服务技术架构和大数据治理实战

大数据时代的微服务之路

共18章 | 纯洁微笑

¥ 51.00 645人订阅

订 阅

在线客服

- 搬家到新站点，<http://www.javathings.top>
- spring boot + mybatis + layui + shiro后台权限管理系统
- 响应式Spring的道法术器（Spring WebFlux 快速上手 + ...
- （4）Reactor 3快速上手——响应式Spring的道法术器
- Spring切入点表达式常用写法
- Spring MVC Controller单例陷阱
- 掌握 MySQL 这 19 个骚操作，效率至少提高3倍
- 谈谈Java引用和Threadlocal的那些事
- 关于jHipster框架在构建中的出现的error修复
- java8的时间和`Date`的对比
- 线程安全意味着不需要同步了吗？
- Angular 6集成Spring Boot 2, Spring Security, JWT和CORS
- 深入理解spring注解之@ComponentScan注解
- 给你一份Spring Boot核心知识清单
- Spring Boot 2.0(四)：使用 Docker 部署 Spring Boot
- Spring Boot 2.0(五)：感受 Docker 魅力， 排解决多应用...
- Netty 防止内存泄漏措施
- 阿里P7给你一份超详细 Spring Boot 知识清单
- 探秘MySQL InnoDB 存储引擎
- Spring Batch快速入门



在线客服