本章学习目标

- 默认参数类型
- @PathVariable 路径参数接收
- 简单数据类型（*）
- JavaBean 类型（*）
- 包装 JavaBean 类型（*）
- 绑定数组类型（*）
- 自定义参数类型转换

1. 默认参数类型

1.1. 页面

```
<h3>1.默认参数类型</h3>
   <form action="" method="post">
    用户名：<input type="text" name="userName"/><br/>
    密码：<input type="password" name="userPass"/><br/>
    <input type="submit" value="提交"/>
   </form>
```

1.2. Controller

```
/**
     * 1.默认参数类型
     */
    @RequestMapping(value="/test1",method=RequestMethod.POST)
    public String test1(HttpServletRequest request){
        String userName = request.getParameter("userName");
```

```
        String userPass = request.getParameter("userPass");
        System.out.println(userName);
        System.out.println(userPass);
        return "success";
    }
```

1.3. 解决 POST 请求中文乱码问题

在 web.xml 加上 Spring 的编码过滤器：

```
<!-- 配置 Spring 的 POST 请求编码过滤器 -->
  <filter>
    <filter-name>CharacterEncodingFilter</filter-name>

    <filter-class>org.springframework.web.filter.CharacterEncodingFilter
</filter-class>
    <init-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </init-param>
  </filter>
  <filter-mapping>
    <filter-name>CharacterEncodingFilter</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
```

2. @PathVariable 路径参数接收

2.1. 页面

```
  <h3>2.@PathVariable 路径参数接收</h3>
   <form
action="${pageContext.request.contextPath}/param/test2/zhangsan/123456.
action" method="post">
    <input type="submit" value="提交"/>
   </form>
```

2.2. Controller

```
/**
     * 2.@PathVariable 路径参数接收
     */
    @RequestMapping(value="/test2/{userName}/{userPass}",method=RequestM
ethod.POST)
    public String test2(@PathVariable(value="userName")String
userName,@PathVariable(value="userPass")String userPass){
        System.out.println(userName);
        System.out.println(userPass);
        return "success";
    }
```

3. 简单数据类型

3.1. 页面

```
    <h3>3.简单数据类型</h3>
   <form action="${pageContext.request.contextPath}/param/test3.action"
method="post">
    用户名：<input type="text" name="userName"/><br/>
    密码：<input type="password" name="userPass"/><br/>
    <input type="submit" value="提交"/>
   </form>
```

3.2. Controller

```
/**
     * 3.简单数据类型
     * @RequestParam:指定参数名称
     *    value: 指定参数名
     *    required: 该参数是否为必须
     *    defaultValue: 默认值
     */
    @RequestMapping(value="/test3",method=RequestMethod.POST)
    public String
test3(@RequestParam(value="name",required=false,defaultValue="zhangsan"
)String userName,String userPass){
        System.out.println(userName);
        System.out.println(userPass);
        return "success";
```

```
    }
```

4. JavaBean 类型（*）

4.1. 页面

```
<h3>4.JavaBean 类型</h3>
   <form action="${pageContext.request.contextPath}/param/test4.action"
method="post">
    用户名：<input type="text" name="userName"/><br/>
    密码：<input type="password" name="userPass"/><br/>
    手机：<input type="text" name="userTelephone"/><br/>
    <input type="submit" value="提交"/>
   </form>
```

4.2. User 类

```
public class User implements Serializable{
    private String userName;
    private String userPass;
    private String userTelephone;
```

4.3. Controller

```
/**
     * 4.JavaBean 类型
     */
```

```
    @RequestMapping(value="/test4",method=RequestMethod.POST)
    public String test4(User user){
        System.out.println(user.getUserName());
        System.out.println(user.getUserPass());
        System.out.println(user.getUserTelephone());
        return "success";
    }
```

5. 包装 JavaBean 类型（*）

5.1. 页面

```
<h3>5.包装 JavaBean 类型</h3>
   <form action="${pageContext.request.contextPath}/param/test5.action"
method="post">
    用户名：<input type="text" name="user.userName"/><br/>
    密码：<input type="password" name="user.userPass"/><br/>
    手机：<input type="text" name="user.userTelephone"/><br/>
    性别：
        <input type="radio" name="gender" value="男"/>男
        <input type="radio" name="gender" value="女"/>女
    <input type="submit" value="提交"/>
   </form>
```

5.2. Controller

```
/**
     * 5.包装 JavaBean 类型
     */
```

```
    @RequestMapping(value="/test5",method=RequestMethod.POST)
    public String test5(UserVo userVo){
        System.out.println(userVo.getUser().getUserName());
        System.out.println(userVo.getUser().getUserPass());
        System.out.println(userVo.getUser().getUserTelephone());
        System.out.println(userVo.getGender());
        return "success";
    }
```

5.3. UserVo 类

```
public class UserVo {


    private User user;
    private String gender;
```

6. 绑定数组类型

6.1. 页面

```
 <h3>6.绑定数组类型</h3>
   <form action="${pageContext.request.contextPath}/param/test6.action"
method="post">
   <input type="checkbox" name="custId" value="1"/>数据 1<br/>
   <input type="checkbox" name="custId" value="2"/>数据 2<br/>
   <input type="checkbox" name="custId" value="3"/>数据 3<br/>
    <input type="submit" value="删除"/>
    </form>
```

6.2. Controller

```
/**
     * 6.绑定数组类型
     */
    @RequestMapping(value="/test6",method=RequestMethod.POST)
    public String test6(@RequestParam("custId")Integer[] id){
        System.out.println(Arrays.asList(id));
        return "success";
    }
```

7. 自定义参数类型转换

通常用于日期类型的转换。

字符串类型使用自定义类型转换器转换为 Date 类。

7.1. 自定义类型转换器

```
/**
 * 自定义类型转换器
 * 字符串类型 -> Date 类型
 * @author lenovo
 *
 */
public class DateConverter implements Converter<String,Date>{
```

```
    @Override
    public Date convert(String birth) {
        //字符串->Date 对象
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        try {
            return sdf.parse(birth);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        return null;
    }

}
```

7.2. 配置 spring-mvc.xml

```
<!-- 配置新的 HandlerMapping 和 HandlerAdapter -->
    <mvc:annotation-driven
conversion-service="conversionServiceFactoryBean"></mvc:annotation-driv
en>

    <!-- 自定义类型转换器 -->
    <bean id="conversionServiceFactoryBean"
class="org.springframework.format.support.FormattingConversionServiceFa
ctoryBean">
        <!-- 转换器集合 -->
        <property name="converters">
            <set>
                <ref bean="dateConverter"/>
```

```
            </set>
        </property>
    </bean>

    <!-- 创建自定义类型转换器 -->
    <bean id="dateConverter" class="cn.sm1234.converter.DateConverter"/>
```