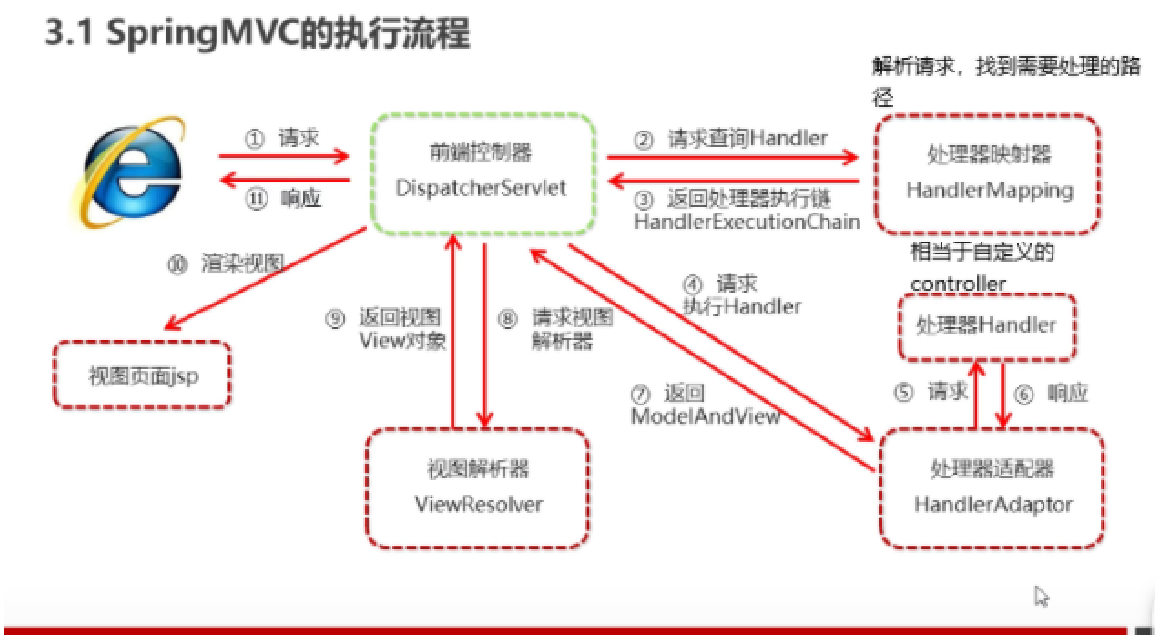
SpringMVC

Spring Framework 提供的 Web 组件，专门用来处理 Web 开发的组件

Servlet + JSP

Spring MVC 取代传统的基于 Servlet 的开发模式，为开发者提供更加便利的 Web 开发机制。

Spring MVC 流程



springmvc流程:

1.pom文件中引入依赖

```xml
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-webmvc</artifactId>
  <version>5.3.15</version>
</dependency>
```

2.在web.xml中配置前端控制器

```xml
<servlet>
  <servlet-name>springmvc</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:springmvc.xml</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>springmvc</servlet-name>
  <url-pattern>/</url-pattern>
</servlet-mapping>
```

3.创建springmvc.xml（视图解析器，自动扫描组件）

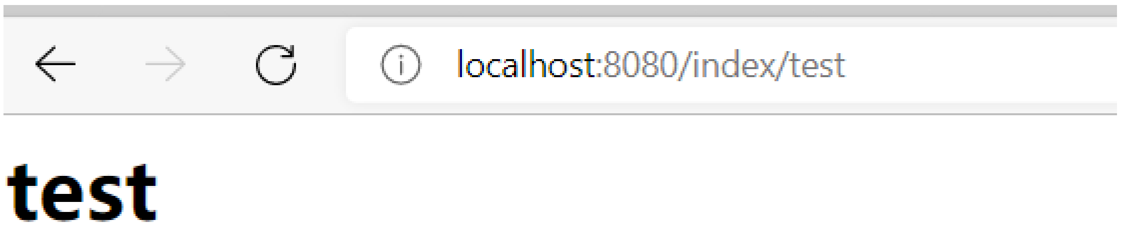
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/context
       http://www.springframework.org/schema/context/spring-context.xsd
       http://www.springframework.org/schema/mvc
       http://www.springframework.org/schema/mvc/spring-mvc-3.2.xsd">

    <context:component-scan base-package="com.ishang.controller">
</context:component-scan>

    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/"></property>
        <property name="suffix" value=".jsp"></property>
    </bean>
</beans>
```

3.在controller层写Handler

```
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.ResponseBody;

@Controller
@RequestMapping("/index")

public class Handle {
    @RequestMapping("/test")
    public String test(){
        System.out.println("执行了test方法");
        return "test";
    }
}
```

4.创建test.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page isELIgnored="false" %>
<html>
<head>
    <title>test2</title>
</head>
<body>
    <h1>test</h1>

</body>
</html>
```



test

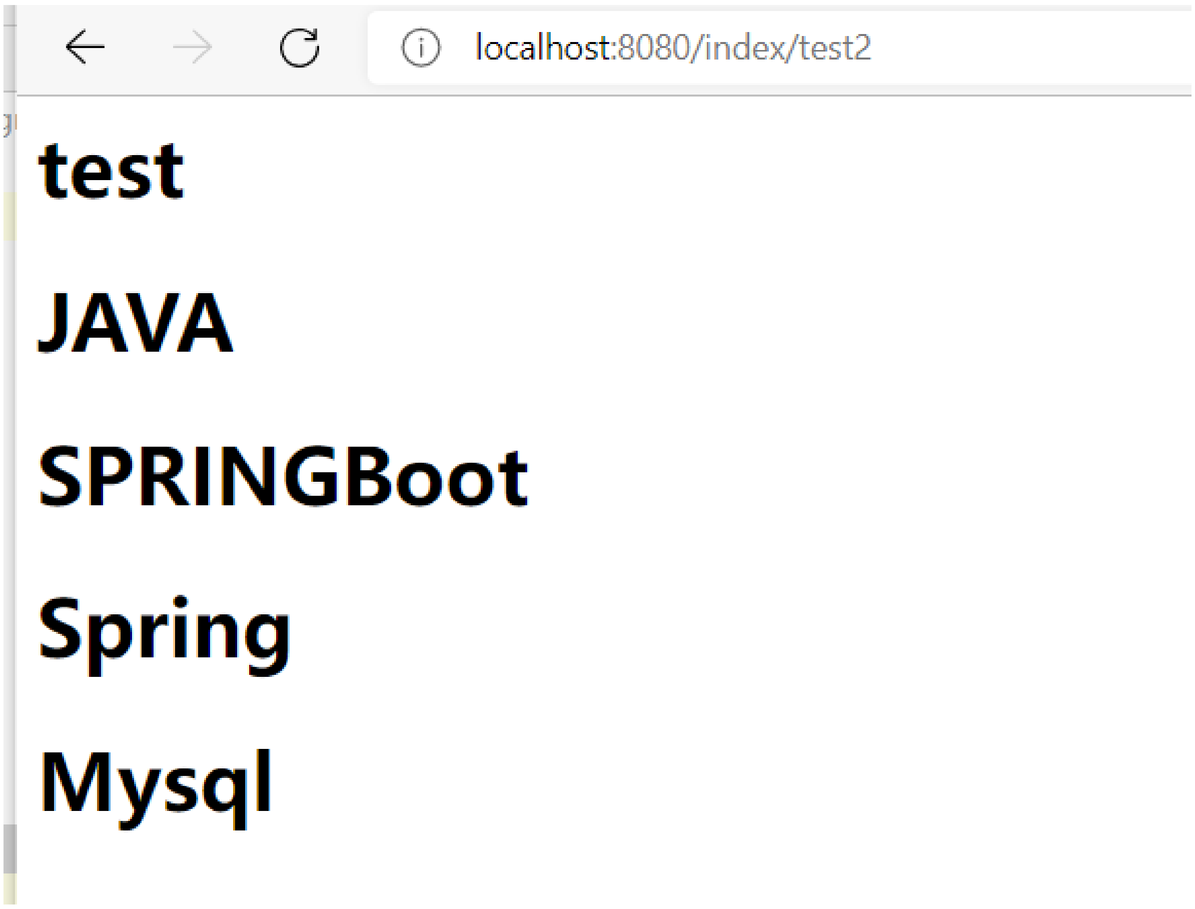
给方法添加 @ResponseBody 注解，直接将方法的返回值返回到页面，而不通过视图解析器

视图数据返回，ModelAndView

```
@RequestMapping("/test2")
public ModelAndView test2(){
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.setViewName("test");
    List<String> list = Arrays.asList("JAVA", "SPRINGBoot", "Spring", "Mysql");
    modelAndView.addObject("list",list);
    return  modelAndView;
}
```

test2.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page isELIgnored="false" %>
<html>
<head>
    <title>test</title>
</head>
<body>
<h1>test</h1>
        <c:forEach  items="${list}" var="str">
        <h1>${str}</h1>
        </c:forEach>

</body>
</html>
```


localhost:8080/index/test2

test

JAVA

SPRINGBoot

Spring

Mysql

springmvc数据校验

1.导入依赖

```
<!--         数据参数校验-->
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate-validator</artifactId>
            <version>5.1.3.Final</version>
        </dependency>
        <dependency>
            <groupId>javax.validation</groupId>
            <artifactId>validation-api</artifactId>
            <version>1.1.0.Final</version>
        </dependency>
        <dependency>
            <groupId>org.jboss.logging</groupId>
            <artifactId>jboss-logging</artifactId>
            <version>3.1.3.GA</version>
        </dependency>
```

2.创建实体类，通过注解在实体类 User 上进行标注

```
import lombok.Data;
import org.hibernate.validator.constraints.NotEmpty;

@Data
public class User {

    @NotEmpty(message = "姓名不能为空")
    private String name;
    @NotEmpty(message = "年龄不能为空")
    private String age;
    @NotEmpty(message = "id不能为空")
    private  String id;
}
```

3.

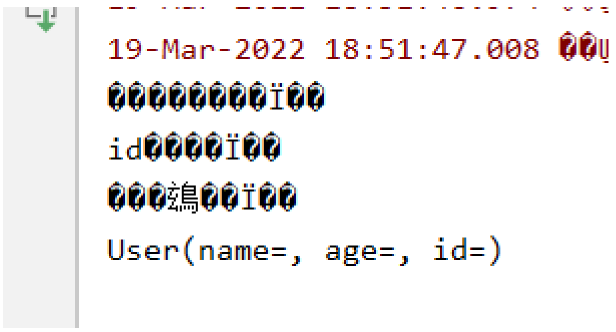
```
 @RequestMapping("/add")
    public String add(Model model){
        model.addAttribute(new User());
        return "add";
    }

@RequestMapping("/adjuge")

public ModelAndView user(@Valid User user, BindingResult bindingResult){
    if (bindingResult.hasErrors()){
        List<ObjectError> allErrors = bindingResult.getAllErrors();
        for (ObjectError allError : allErrors) {
            System.out.println(allError.getDefaultMessage());
        }
    }
}
```

4.add.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>
<%--<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form" %>--
%>
<html>
<head>
    <title>Title</title>
</head>
<body>

<%--    以form表单的方式提交--%>
<form:form action="/index/adjuge" method="post" modelAttribute="user">
    <form:input path="name"></form:input>
    <form:input path="age"></form:input>
    <form:input path="id"></form:input>
    <input type="submit" value="提交">
</form:form>

</body>
</html>
```

5.添加注解驱动

```xml
<mvc:annotation-driven></mvc:annotation-driven>
```



```
19-Mar-2022 18:51:47.008 ��\u
��������Ï��
id����Ï��
���鴳��Ï��
User(name=, age=, id=)
```