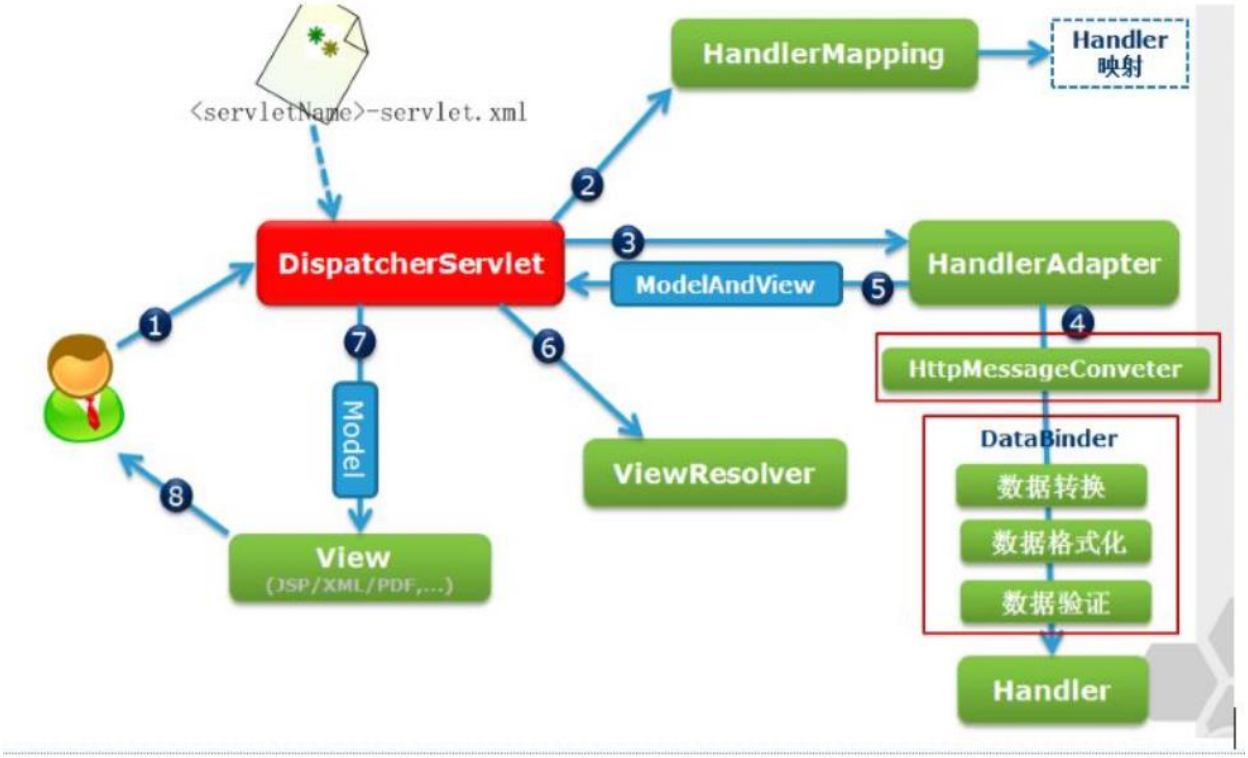
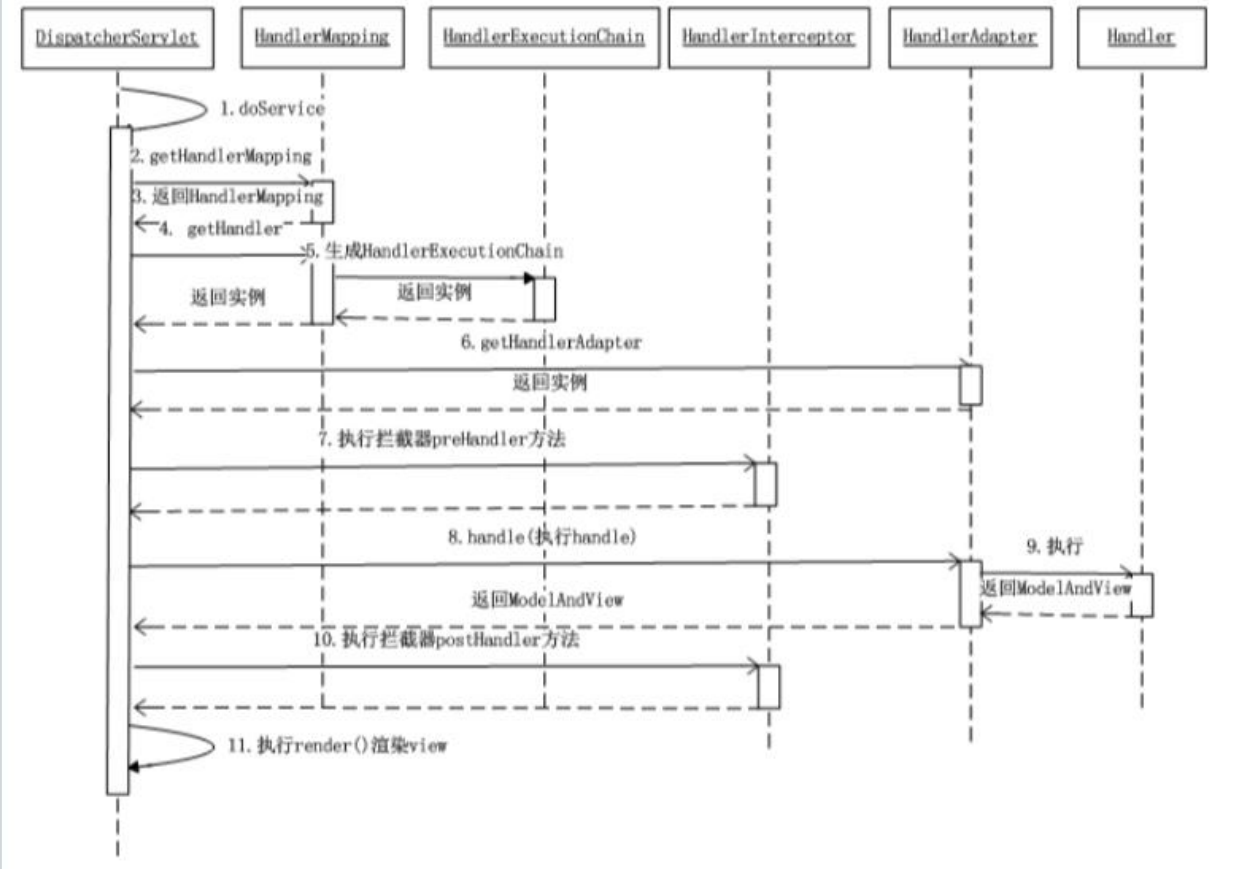
1、SpringMVC的工作机制



springMVC的工作流程？



（1）用户发送请求，经过前端控制器DispatcherServlet，DispatcherServlet将URL交给处理器映射器HandlerMapping处理HandlerMapping 有点类似于Struts2中的ActionMapper.

（2）HandlerMapping接收到URL，处理完URL，返回了HandlerExecutionChain，HandlerExecutionChain里面包含了当前URL对应的Handler的所有信息，包括Handler本身以及当前Handler对象对应的拦截器对象

（3）HandlerMapping处理返回HandlerExecutionChain之后，DispatcherServlet会根据当前的Handler请求对应的HandlerAdapter。如果当前的Handler有拦截器的话，会先调用拦截器的preHandler方法，然后会调用Handle方法，而HandlerAdapter会调用我们的Handler对象中的执行方法（类似于Action里面的Execute方法）

（4）Handler对象执行方法结束后，返回ModelAndView

（5）ModelAndView会被视图解析器（ViewResolver）解析，然后返回到DispatcherServlet，最后DispatcherServlet将对应的视图返回给客户端扩展

注意：SpringMVC中为什么要用两个对象HandlerMapping和HandlerAdapter,不能把这两个对象合并吗?

答：可以合并，只是我们在设计框架的时候，分成两个对象更加遵循单一职责原则

web.xml配置文件中：

```
id="WebApp_ID" version="3.1">
<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>
<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:applicationContext_springmvc.xml</param-value>
    </init-param>
</servlet>
<servlet-mapping>
    <servlet-name>springmvc</servlet-name>
    <url-pattern>*.action</url-pattern>
</servlet-mapping>

/web-app>
```

applicationContex配置文件中：

```xml
<?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.xsd">

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

    <!-- 1.自动组件扫描 -->
    <context:component-scan base-package="com.gem.springmvc.controllers"/>
    <!-- HandlerMapping和HandlerAdapter -->
    <!--
    <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"/>
    <bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"/>
     -->
    <mvc:annotation-driven conversion-service="conversionService" />

    <!-- 配置convensionService -->
    <bean id="conversionService" class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
        <property name="converters">
            <list>
                <bean class="com.gem.springmvc.converters.UtilDateConverter"/>
            </list>
        </property>
    </bean>
```

StudentHandler.java文件中：

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

@Controller
public class StudentHandler2 {

    @RequestMapping("/addStu2.action")
    public String addStu(int sid , String sname , Date birth ) throws UnsupportedEncodingException{
        System.out.println("sid:"+sid);
        System.out.println("sname:"+sname);
        System.out.println("birth:"+birth);
//      System.out.println("cname:"+car.getCname());
        return "index";
    }
}
```

或者IndexHandler.java文件中：

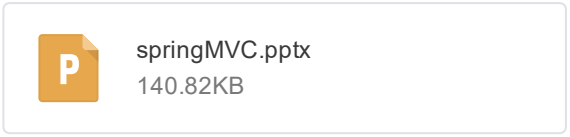
```java
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

@Controller
@RequestMapping("/usr")
public class IndexHandler{
    @RequestMapping(value="/index1.action",method=RequestMethod.GET)
    public void index1(HttpServletRequest request , HttpServletResponse response , HttpSession session ) throws ServletException, IOException{
        System.out.println("hello springmvc2");
        request.getRequestDispatcher("/index.jsp").forward(request, response);
    }
}
```

如果Jsp页面中要上传日期类型的：

```java
import org.springframework.core.convert.converter.Converter;

public class UtilDateConverter implements Converter<String, Date> {

    @Override
    public Date convert(String str) {
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
        try {
            return sdf.parse(str);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        return null ;
    }

}
```

```xml
<!-- 配置convensionService -->
<bean id="conversionService" class="org.springframework.format.support.FormattingConversionServiceFactoryBean">
    <property name="converters">
        <list>
            <bean class="com.gem.springmvc.converters.UtilDateConverter"/>
        </list>
    </property>
</bean>
```

---------->



54. struts2 和 springMVC 的区别？

1、Struts2 是类级别的拦截， 一个类对应一个 request 上下文，SpringMVC 是方法级别的拦截，一个方法对应一个 request 上下文，而方法同时又跟一个 url 对应,所以说从架构本身上 SpringMVC 就容易实现 restful url,而 struts2 的架构实现起来要费劲，因为 Struts2 中 Action 的一个方法可以对应一个 url，而其类属性却被所有方法共享，这也就无法用注解或其他方式标识其所属方法了。

2、由上边原因，SpringMVC 的方法之间基本上独立的，独享 request response 数据，请求数据通过参数获取,处理结果通过 ModelMap 交回给框架,方法之间不共享变量,而 Struts2 搞的就比较乱，虽然方法之间也是独立的，但其所有 Action 变量是共享的，这不会影响程序运行，却给我们编码 读程序时带来麻烦，每次来了请求就创建一个 Action，一个 Action 对象对应一个 request 上下文。

3、由于 Struts2 需要针对每个 request 进行封装，把 request，session 等 servlet 生命周期的变量封装成一个一个 Map，供给每个 Action 使用，并保证线程安全，所以在原则上，是比较耗费内存的。

4、 拦截器实现机制上，Struts2 有以自己的 interceptor 机制，SpringMVC 用的是独立的 AOP 方式，这样导致 Struts2 的配置文件量还是比 SpringMVC 大。

5、SpringMVC 的入口是 servlet，而 Struts2 是 filter（这里要指出，filter 和 servlet 是不同的。以前认为 filter 是 servlet 的一种特殊），这就导致了二者的机制不同，这里就牵涉到 servlet 和 filter 的区别了。

6、SpringMVC 集成了 Ajax，使用非常方便，只需一个注解@ResponseBody 就可以实现，然后直接返回响应文本即可，而 Struts2 拦截器集成了 Ajax，在 Action 中处理时一般必须安装插件或者自己写代码集成进去，使用起来也相对不方便。

7、SpringMVC 验证支持 JSR303，处理起来相对更加灵活方便，而 Struts2 验证比较繁琐，感觉太烦乱。

8、Spring MVC 和 Spring 是无缝的。从这个项目的管理和安全上也比 Struts2 高(当然 Struts2 也可以通过不同的目录结构和相关配置做到 SpringMVC 一样的效果，但是需要 xml 配置的地方不少)。

9、 设计思想上，Struts2 更加符合 OOP 的编程思想， SpringMVC 就比较谨慎，在 servlet 上扩展。

10、SpringMVC 开发效率和性能高于 Struts2。

11、SpringMVC 可以认为已经 100%零配置。