课程回顾：

1. 泛型

2. 反射

3. 注解

4. 注解优化BaseDao

目标：

1. 写一个mystruts框架

2. Strtus框架

# 1. 写一个mystruts框架

MVC模式：

Model 模型

View 视图

Control 控制器

Control， 控制器

Servlet起到控制器作用！

----》 获取请求数据封装 【BeanUtils可以优化，(调用方法？)】

----》 调用Service处理业务逻辑

----》 跳转(转发/重定向) 【跳转代码写死】

传统mvc开发总结：

1. 跳转代码写死，不灵活

2. 每次都去写servlet，web.xml中配置servlet！

(配置目的： 请求, Servlet处理类)

需求：

登陆、注册

登陆成功 --🡪 首页

注册成功

--🡪 登陆

--🡪 首页

# 2. 基于mvc模式的应用框架之struts

Struts就是基于mvc模式的框架！

(struts其实也是servlet封装，提高开发效率！)

Struts开发步骤：

1. web项目，引入struts - jar包

2. web.xml中，引入struts的核心功能

配置过滤器

3. 开发action

4. 配置action

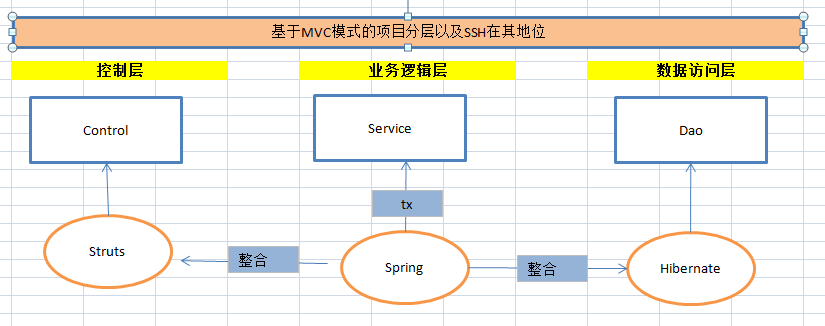
src/struts.xml

|  |
| --- |
| 1. 引入8个jar文件 |
| 2. web.xml |
| <!-- 引入struts核心过滤器 -->  <filter>  <filter-name>struts2</filter-name>  <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>struts2</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |
| 3. 开发Action |
| // 开发action： 处理请求  **public** **class** HelloAction **extends** ActionSupport {    // 处理请求  **public** String execute() **throws** Exception {  System.*out*.println("访问到了action，正在处理请求");  System.*out*.println("调用service");  **return** "success";  }  } |
|  |
| 4. 配置action : src/struts.xml |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"  "http://struts.apache.org/dtds/struts-2.0.dtd">  <struts>  <package name=*"xxxx"* extends=*"struts-default"*>  <action name=*"hello"* class=*"cn.itcast.action.HelloAction"* method=*"execute"*>  <result name=*"success"*>/success.jsp</result>  </action>  </package>  </struts> |

# 3. Struts框架学习

## a. 框架学习概述

SSH框架在mvc模式的的位置作用：



框架：

软件中的框架，是一种半成品； 我们项目开发需要在框架的基础上进行！

因为框架已经实现了一些功能，这样就可以提高开发效率！

## b. Struts2框架

Struts1最早的一种基于mvc模式的框架；

Struts2 是在Struts1的基础上，融合了xwork的功能;

也可以说，Struts2 = struts1 + xwork

Struts2框架预先实现了一些功能：

1. 请求数据自动封装

2. 文件上传的功能

3. 对国际化功能的简化

4. 数据效验功能

……………….

## c.Struts2开发流程：

版本： 2.3

* 引入jar文件

commons-fileupload-1.2.2.jar 【文件上传相关包】

commons-io-2.0.1.jar

struts2-core-2.3.4.1.jar 【struts2核心功能包】

xwork-core-2.3.4.1.jar 【Xwork核心包】

ognl-3.0.5.jar 【Ognl表达式功能支持表】

commons-lang3-3.1.jar 【struts对java.lang包的扩展】

freemarker-2.3.19.jar 【struts的标签模板库jar文件】

javassist-3.11.0.GA.jar 【struts对字节码的处理相关jar】

* 配置web.xml

Tomcat启动-🡪 加载自身web.xml---🡪加载所有项目的web.xml

通过在项目的web.xml中引入过滤器，

-🡪Struts的核心功能的初始化，通过过滤器完成

🡪 filter 【

init/ 启动执行

doFilter/ 访问执行

destroy

】

|  |
| --- |
| <!-- 引入struts核心过滤器 -->  <filter>  <filter-name>struts2</filter-name>  <filter-class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>struts2</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping> |
|  |

struts2-core-2.3.4.1.jar

StrutsPrepareAndExecuteFilter 即为核心过滤器

注意：

使用的struts的版本不同，核心过滤器类是不一样的！

* 开发Action

注意：

1. action类，也叫做动作类; 一般继承ActionSupport类

即处理请求的类 (struts中的action类取代之前的servlet)

2. action中的业务方法，处理具体的请求

-🡪 必须返回String

🡪 方法不能有参数

|  |
| --- |
| **public** **class** HelloAction **extends** ActionSupport {  // 处理请求  **public** String execute() **throws** Exception {}  } |

* 配置struts.xml

## c.Struts2执行流程

服务器启动：

1. 加载项目web.xml

2. 创建Struts核心过滤器对象， 执行filter 🡪 init()

struts-default.xml, 核心功能的初始化

struts-plugin.xml, struts相关插件

struts.xml 用户编写的配置文件

访问：

3. 用户访问Action, 服务器根据访问路径名称，找对应的aciton配置, **创建action对象**

4. 执行默认拦截器栈中定义的18个**拦截器**

5. 执行action的**业务处理方法**

struts-default.xml, 详解

目录：struts2-core-2.3.4.1.jar/ struts-default.xml

内容：

1. bean节点指定struts在运行的时候创建的对象类型

2.指定struts-default包 【用户写的package(struts.xml)一样要继承此包 】

package struts-default 包中定义了：

a. 跳转的结果类型

dispatcher 转发，不指定默认为转发

redirect 重定向

redirectAction 重定向到action资源

stream (文件下载的时候用)

b. 定义了所有的拦截器

定义了32个拦截器！

为了拦截器引用方便，可以通过定义栈的方式引用拦截器，

此时如果引用了栈，栈中的拦截器都会被引用!

defaultStack

默认的栈，其中定义默认要执行的18个拦截器！

c. 默认执行的拦截器栈、默认执行的action

<default-interceptor-ref name="defaultStack"/>

<default-class-ref class="com.opensymphony.xwork2.ActionSupport" />

<interceptor

name="prepare" class="com.opensymphony.xwork2.interceptor.PrepareInterceptor"/>

<interceptor

name="params" class="com.opensymphony.xwork2.interceptor.ParametersInterceptor"/>

拦截器（先睹为快）：

拦截器功能与过滤器功能类似。

区别：

共同点： 都拦截资源！

区别：

过滤器，拦截器所有资源都可以； (/index.jsp/servlet/img/css/js)

拦截器，只拦截action请求。

拦截器是struts的概念，只能在struts中用。

过滤器是servlet的概念，可以在struts项目、servlet项目用。

// 面试题： 拦截器什么时候执行？ (访问/启动) 先执行action类创建，先执行拦截器？

// --》 1. 用户访问时候按顺序执行18个拦截器；

//---》 2. 先执行Action类的创建，再执行拦截器； 最后拦截器执行完，再执行业务方法

## d.共性问题

问题1：Struts.xml配置文件没有提示

解决a：

找到struts-2.0.dtd文件, 拷贝到某个目录：d:/dtd /.. (不要用中文目录)

让MyEclipse关联到上面dtd文件，

🡪 windows 🡪preferences -🡪 搜索xml catalog

配置：

Location: 上面配置的dtd目录

Key: -//Apache Software Foundation//DTD Struts Configuration 2.0//EN

解决b:

或者，

让机器连接互联网，工具会自动下载dtd文件，缓存到MyEclipse中！

配置文件解释

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"  "http://struts.apache.org/dtds/struts-2.0.dtd">  <struts>  <!--  package 定义一个包。 包作用，管理action。  (通常,一个业务模板用一个包)  name 包的名字； 包名不能重复；  extends 当前包继承自哪个包  在struts中，包一定要继承struts-default  struts-default在struts-default.xml中定的包  abstract  表示当前包为抽象包； 抽象包中不能有action的定义，否则运行时期报错  abstract=true 只有当当前的包被其他包继承时候才用！  如：  <package name="basePackage" extends="struts-default" abstract="true"></package>  <package name="user" extends="basePackage">  namespace 名称空间，默认为"/"  作为路径的一部分  访问路径= http://localhost:8080/项目/名称空间/ActionName        action 配置请求路径与Action类的映射关系  name 请求路径名称  class 请求处理的aciton类的全名  method 请求处理方法  result  name action处理方法返回值  type 跳转的结果类型  标签体中指定跳转的页面    -->  <package name="user" extends="struts-default" namespace="/">  <action name="login" class="cn.itcast.b\_execute.UserAction" method="login">  <result name="login">/index.jsp</result>  </action>  </package>    </struts> |

总配置文件可以实现包含

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8" ?>  <!DOCTYPE struts PUBLIC  "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"  "http://struts.apache.org/dtds/struts-2.0.dtd">  <struts>  <!-- struts在运行时候会加载这个总配置文件： src/struts.xml -->    <!-- 总配置文件中引入其他所有的配置文件 -->  <include file="cn/itcast/a\_action/hello.xml"></include>  <include file="cn/itcast/b\_execute/config.xml"></include>      </struts> |