入门实现：

1、写两个HTML或JSP页面（一个用于用户访问 一个用于访问交互的跳转）

第一个JSP

```
<body>
    <%--入门案例:通过超链接访问struts2中类的方法
        struts2框架的核心控制器（Filter）默认会接受 以.action为后缀的请求，或者是没有任何后缀的请求.
        默认情况下用户访问看到的地址是：
        http://localhost:8080/Struts2Test/hello
        http://localhost:8080/Struts2Test/hello.action
    --%>
    <a href="/Struts2Test/hello">访问Struts2的第一个案例</a>
    <hr/>
    <a href="/Struts2Test/hello.action">访问Struts2的第一个案例.action</a>
    <hr/>
    <%--
    通过对struts.xml配置文件的修改  <constant name="struts.action.extension" value="do"></constant>
    改变默认接受值之后可以使用 任意的后缀 指定某种后缀之后其它后缀将无法访问--%>
    <a href="/Struts2Test/hello.do">访问Struts2的第一个案例.do</a>
    <hr/>
    <a href="${pageContext.request.contextPath}/hello.abc">访问Struts2的第一个案例.abc</a>
    <hr/>
    <a href="${pageContext.request.contextPath}/hello.html">访问Struts2的第一个案例.html</a>
</body>
```

第二个JSP

success.jsp

```
1  <%@ page language="java" contentType="text/html; char
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transiti
4  <html>
5  <head>
6  <meta http-equiv="Content-Type" content="text/html; c
7  <title>Insert title here</title>
8  </head>
9  <body>
10 执行成功
11 </body>
```

2、写一个类定义一个带返回值为String类型的方法

3、配置struts.xml

代码见 入门案例

在配置文件中有配置拦截后缀的情况下又保证请求中不带后缀能使用的方法

改变struts2默认拦截的url后缀为.html 如果页面是html那么拦截后缀就不能是html

value的值为： ,action 请求地址不用加后缀名action

```
<constant name="struts.action.extension" value=",action">
</constant>
```

请求就可以不要后缀来使用

```
<form action="/SSH_Inte/api/customer/apiins" method="post">
```

HTML或JSP中链接的相关问题

入门案例:通过超链接访问struts2中类的方法

struts2框架的核心控制器（Filter）默认只会接受　以.action为后缀的请求，或者是没有任何后缀的请求.

默认情况下用户访问看到的地址是:

http://localhost:8080/Struts2Test/**hello**

http://localhost:8080/Struts2Test/**hello.action**

```
<a href="/Struts2Test/hello">访问Struts2的第一个案例</a>
<a href="/Struts2Test/hello.action">访问Struts2的第一个案例.action</a>
```

通过对struts.xml配置文件的修改

```
<constant name="struts.action.extension" value="do">
</constant>
```

改变默认接受值之后可以使用　任意的后缀　指定某种后缀之后其它后缀将无法访问

```
<a href="/Struts2Test/hello.do">访问Struts2的第一个案例.do</a>
```

修改配置为:

```
<constant name="struts.action.extension" value="abc">
</constant>
```

```
<a href="${pageContext.request.contextPath}/hello.abc">访问Struts2的.abc</a>
```

修改配置为:

```
<constant name="struts.action.extension" value="html">
```

```
</constant>
```

```
<a href="${pageContext.request.contextPath}/hello.html">访问Struts2的.html</a>
```

总结：你配置文件里配置的是什么后缀，页面里就用什么后缀，地址就显示什么后缀，不配置默认就是空或**.action**后缀

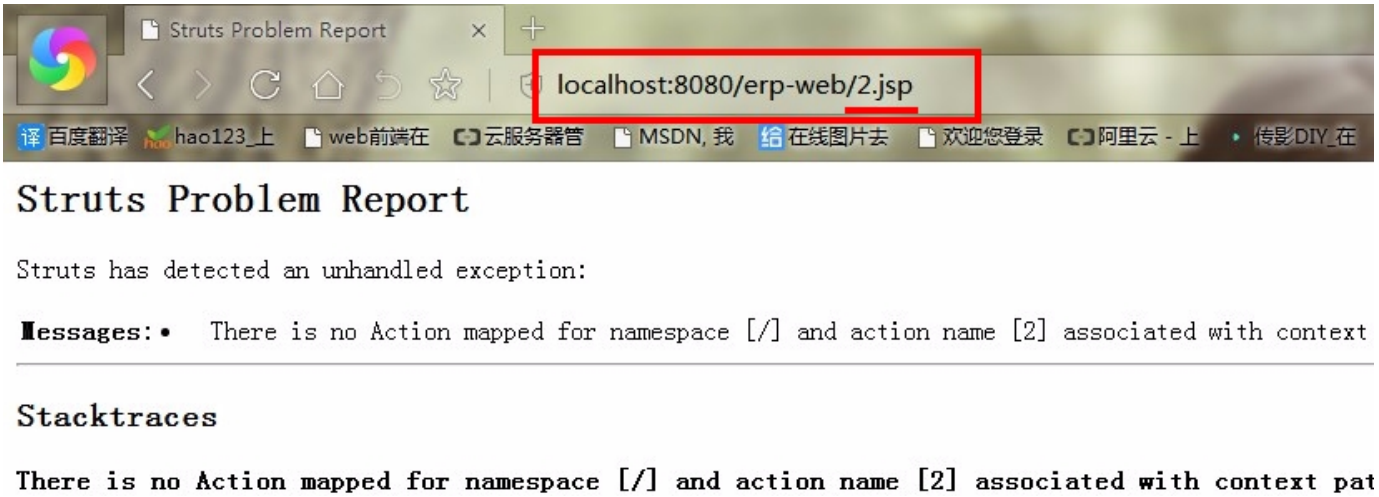
struts.xml里的的常用配置：

```
<!-- 开启开发者模式： -->
<constant name="struts.devMode" value="true"></constant>
```

```
<!-- 修改struts2默认接受的url后缀为.do  这个配置可以改客户访问所看到的地址后坠
    如果改.do  那客户看到的就是http://xxxx/xxx.do  -->
```

后缀设置非常重要：后缀的设置不能与任何静态页面后扩展名一模一样 可以是**action do** 等等，一句话不能把后缀设置成**html,jsp**等等，可以是别的

如果把后缀设置成**html**，当你启动服务去访问项目中的**html**的静态页面就会报一个非常严重的错误，错误见下：




```
<constant name="struts.action.extension" value="html">
</constant>
```

struts.xml里的的相关标签说明：

package标签具体说明：

<package name="p1" extends="struts-default" namespace="/n1">

package标签：把配置文件按照面向对象的思想来管理。使我们可以按照模块化来开发。

属性：

name：指定包的名称。必须写。并且包的名称必须唯一。

extends：指定当前包的父包。子包自动具备父包的配置。它其实就是面向对象思想管理配置文件的体现。我们的包一般都继承struts-default。

该包在struts-default.xml中定义着。如果我们不继承struts-default，则不能使用struts2的核心功能。

abstract:把当前包声明为抽象包。抽象包就是用来被继承的，它里面都定义是一些公共的配置。

只有没有action标签的包才能声明为抽象包。

namespace：指定当前包的名称空间。是让我们把访问动作的URL按照模块化来管理。

当我们指定了名称空间之后：

访问的路径URL就变成了：名称空间+action标签的name属性取值。

例如：/customer/hello.action

名称空间的写法：

必须以 /开头。不能有中文。并且必须/后面紧跟的是字母

名称空间可以有多级

例如：

系统管理——用户管理 /system/user/addUser.action

系统管理——角色管理 /system/role/updateRole.action

namespace有默认值：

默认值是："" 当我们不写的时候有默认的名称空间

action标签具体说明：

<action **name="findUser"**

class="com.itheima.web.action.UserAction"

method="findUser"> </action>

action标签：指定动作名称和动作类以及动作方法之间的对应关系

属性：

name：指定动作名称。该值必须唯一。和浏览器中的必须保持一致。此处不要写也不能写后缀。

class:指定动作类的全限定类名。

method：指定的是动作方法名称。

创建动作类的三种方式：

第一种：无侵入式的 一般不用

例如：HelloAction

```
public class HelloAction {

    public String sayHello(){
        System.out.println(this);
        System.out.println("HelloAction的sayHello方法执行了。。。。。");
        return "success";
    }
```

第二种：实现接口的方式 一般也不用

需要实现一个Action的接口

```java
/**
 * 创建动作类的第二种方式：
 *      实现Action接口
 */
public class Hello2Action implements Action{

    @Override
    public String execute() throws Exception {
        System.out.println("Hello2Action的execute方法执行了。。。。。1");
        return SUCCESS;
    }
```

接口中定义着5个常量

SUCCESS：一般用于执行成功。

LOGIN：一般用于去登录页面

ERROR：一般用于动作方法执行产生异常，前往错误页面error.jsp

NONE：一般用于不返回任何结果视图。例如：文件下载或者ajax异步请求

INPUT：一般用于数据回显。在**struts2**中，框架当执行出现问题时默认会返回**input**视图 当成重点来看

回显要注意的事情：

1、从哪来回哪去 register.jsp——>register.jsp

2、必须把错误的信息带回去

3、必须把原来的数据带回去

默认动作方法：

execute。当执行此方法时，可以不写method属性。

例如：Hello2Action

第三种：继承ActionSupport类 实际开发中采用的方式

需要继承ActionSupport类

例如：Hell3Action

```java
/**
 * 创建动作类的第三种方式
 *  继承ActionSupport
 */
public class Hello3Action extends ActionSupport{

}
```

默认动作类型：

ActionSupport类

result标签：

作用：用于定义结果视图

属性：

name：逻辑结果视图。它是一个字符串，需要和动作方法的返回值进行比较

type：采用何种方式前往。

常用的取值：

dispatcher：请求转发

Dispatcher dispatcher =request.getRequestDispatcher("jsp");

dispatcher.forward(request,response);

redirect：重定向到另外一个jsp页面

redirectAction：重定向到另外一个Action

stream：它是用于文件下载的