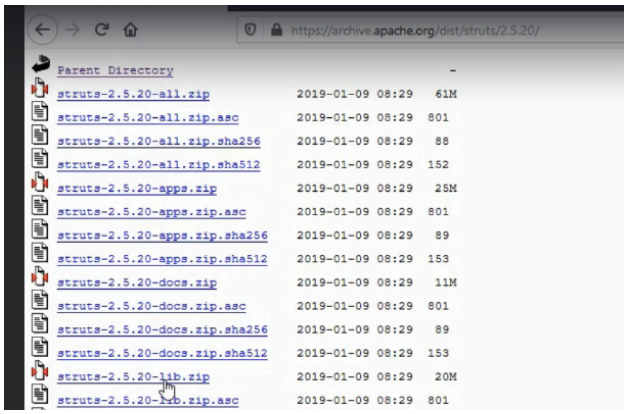
Struts 2 Annotations:

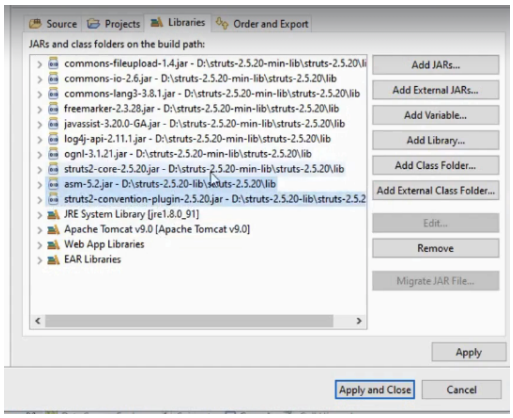
> Two ways of declaring Struts 2 architecture:-
> 1) XML-based configurations (struts.xml)
> 2) Annotations

> Benefits of annotations:-
> 1) Reduce our code to some extent
> 2) Provides support for some tools
> 3) Organizes code in a better way

We need additional jar files

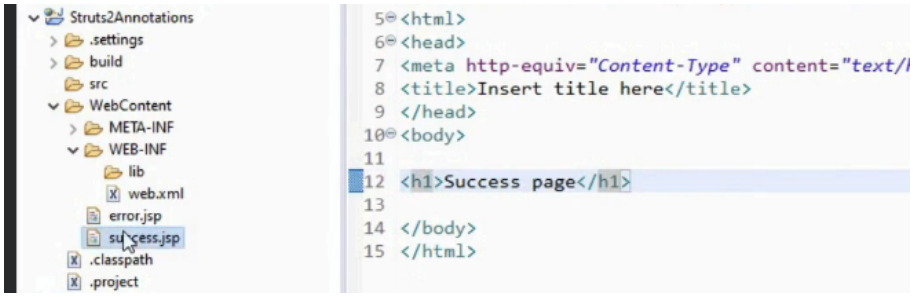


We need these two additional jars

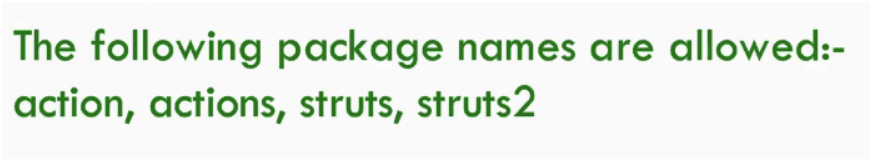


And also deployment assembly -> add -> select Java Build Path Entities -> select jars and finish.

Add success and error jsp pages

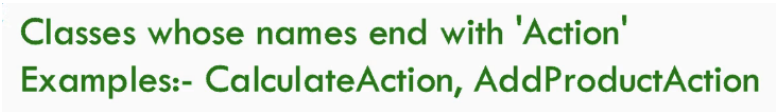


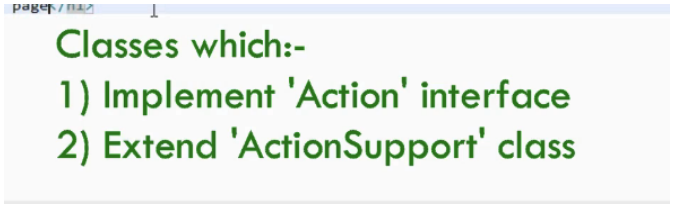
Our action classes must be any one of this package. The below are the conventions we have to follow for annotation configuration.



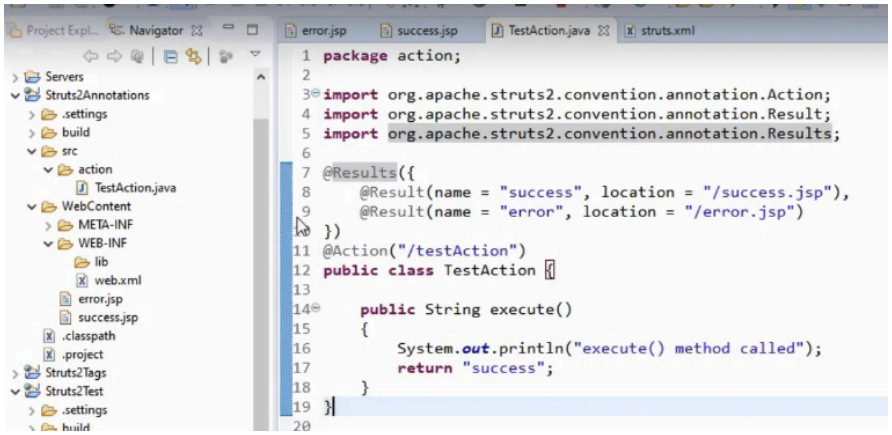
Annotations won't work for the last column in the above pic

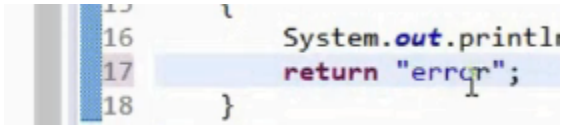
Framework locates action classes with above package name and below classes convention and their implementation/extend.

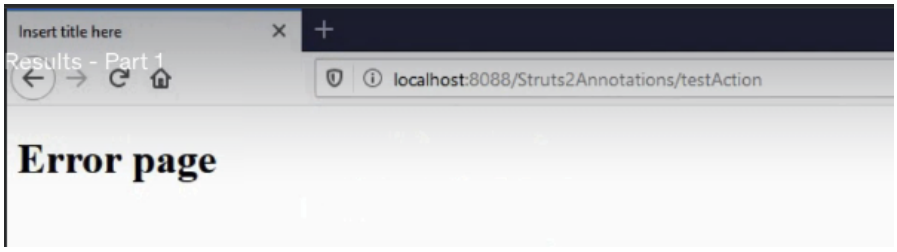




In line number 8 name refers to the endpoint and class refers to the action class which has an execute method. In that place we use this annotation. We can use this at either class level or at execute method as shown in below screenshot. The above one xml the below one java config.

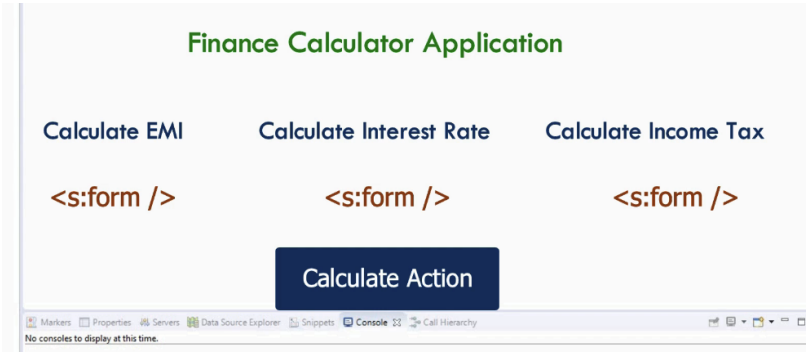






Combine Action and result annotations as follow (another approach)

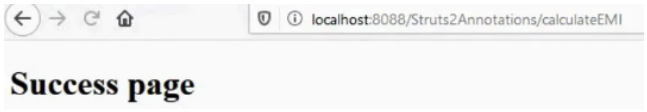
```java
package actions;

import org.apache.struts2.convention.annotation.Action;

@Action(value = "/testAction", results = { @Result(name = "success", location = "/success.jsp"),
        @Result(name = "error", location = "/error.jsp") })
public class TestAction {

    public String execute()
    {
        System.out.println("execute() method called");
        return "error";
    }
}
```

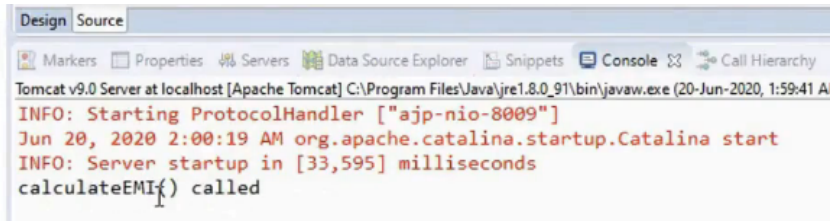


Single action class is okay for multiple similar actions

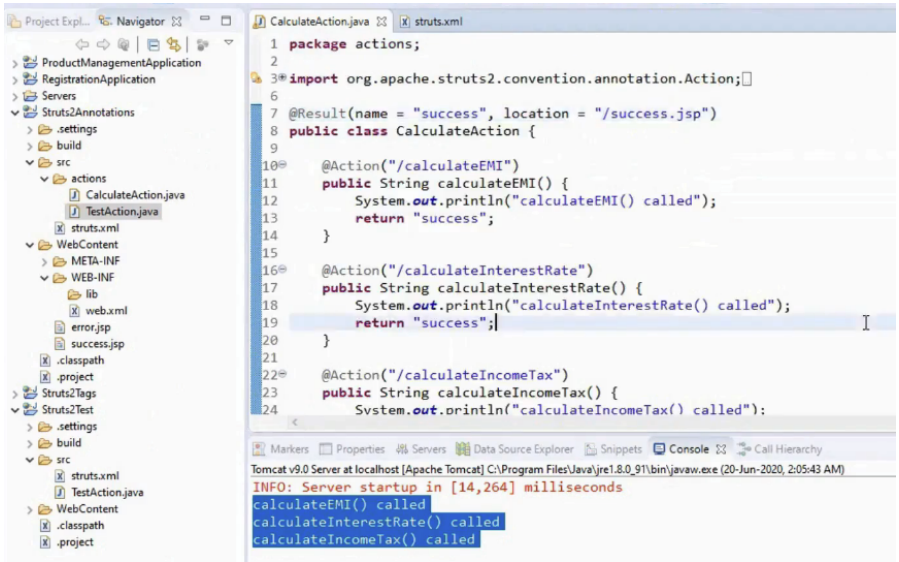
```java
package actions;

import org.apache.struts2.convention.annotation.Action;


public class CalculateAction {

    public String calculateEMI() {
        System.out.println("calculateEMI() called");
        return "success";
    }

    public String calculateInteres sessions {
        System.out.println("calculateInterestRate() called");
        return "success";
    }

    public String calculateIncomeTax() {
        System.out.println("calculateIncomeTax() called");
        return "success";
    }

}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
    "http://struts.apache.org/dtds/struts-2.5.dtd">

<struts>
    <package name="test" extends="struts-default">
        <action name="calculateEMI" class="actions.CalculateAction" method="calculateEMI">
            <result name="success">/success.jsp</result>
        </action>
        <action name="calculateInterestRate" class="actions.CalculateAction" method="calculateInterestRate">
            <result name="success">/success.jsp</result>
        </action>
        <action name="calculateIncomeTax" class="actions.CalculateAction" method="calculateIncomeTax">
            <result name="success">/success.jsp</result>
        </action>
    </package>
</struts>
```



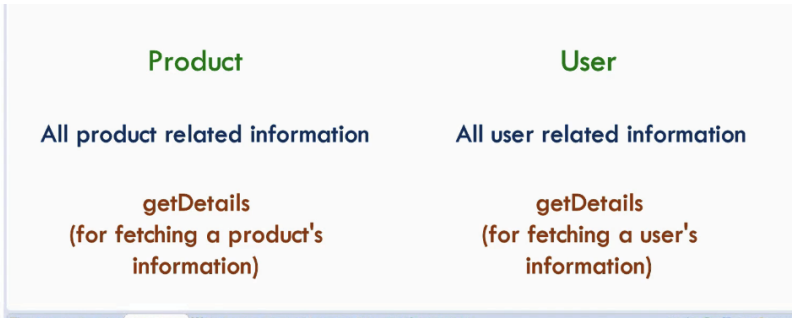


The above one is xml configuration and the below one is java configuration

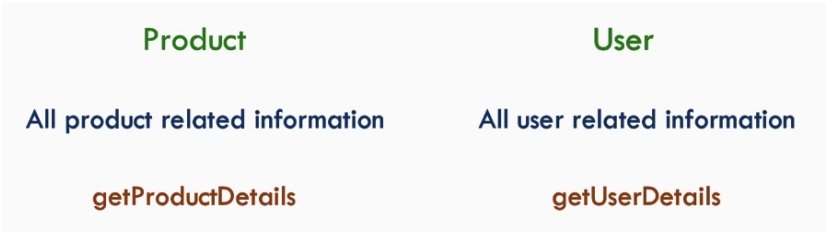


Now let us map different jsp pages for different methods

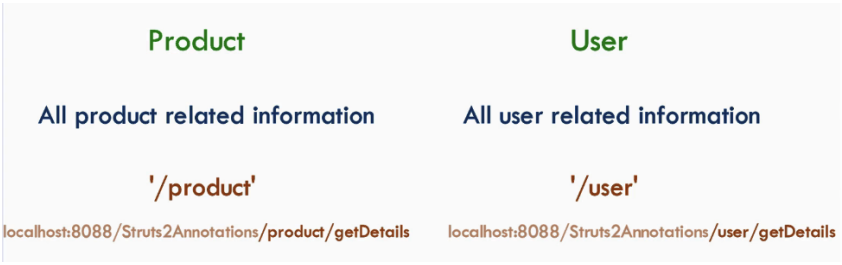
Naming space:



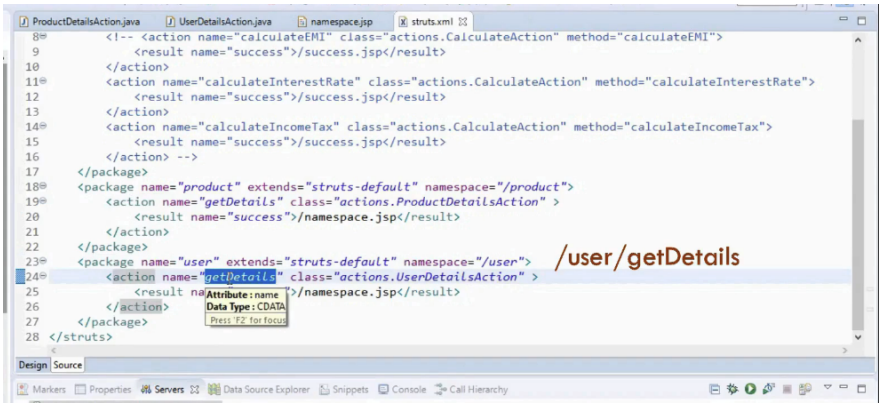
If we try like below. Imagine if we have 100 of such actions (difficult to remember)

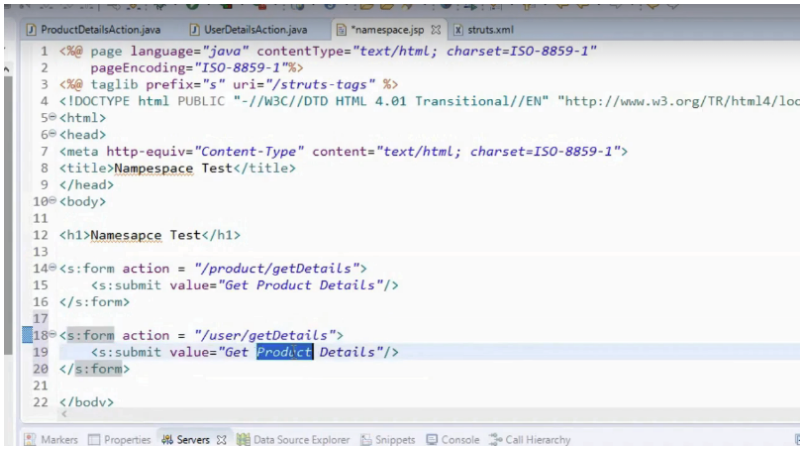


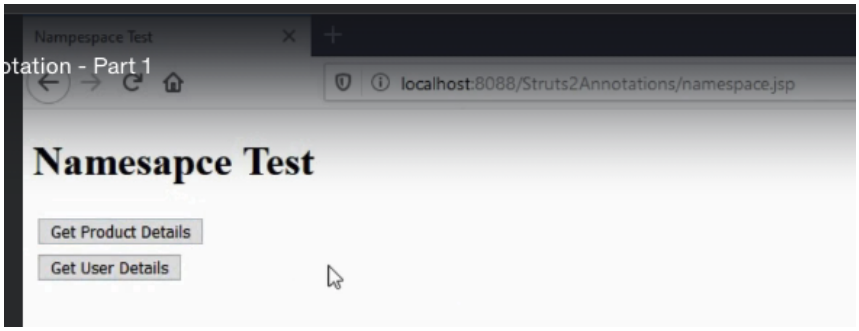
The below one is called name space.



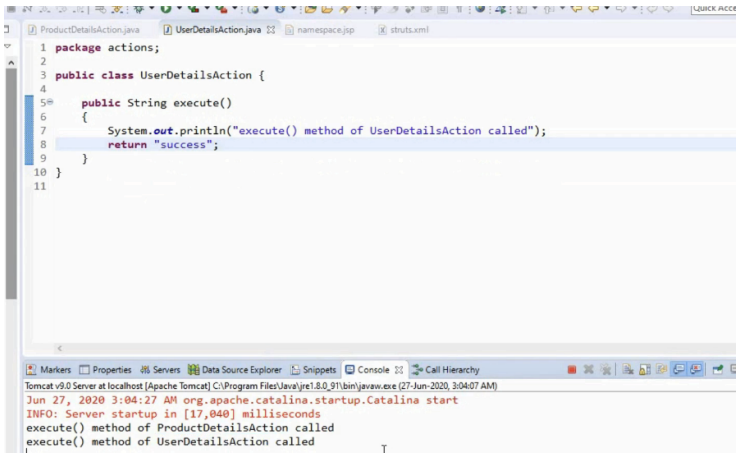
Packages are used to group actions into logical units as shown below where we have same action line number 19 and 24 .



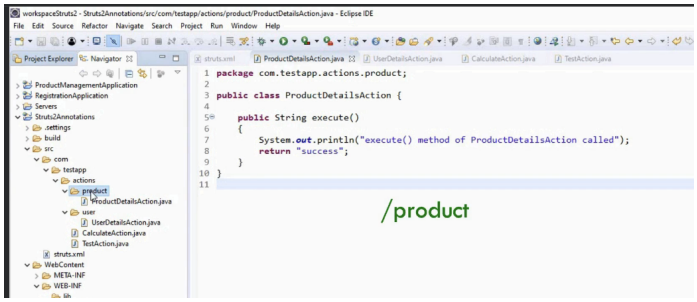




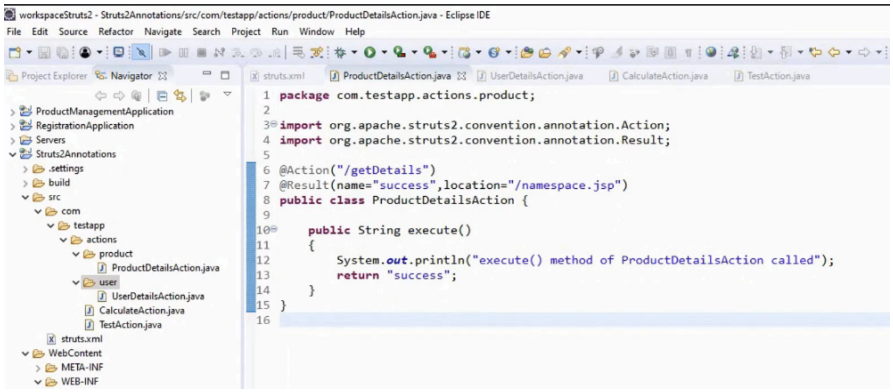
Click the above link and not the messages in console.

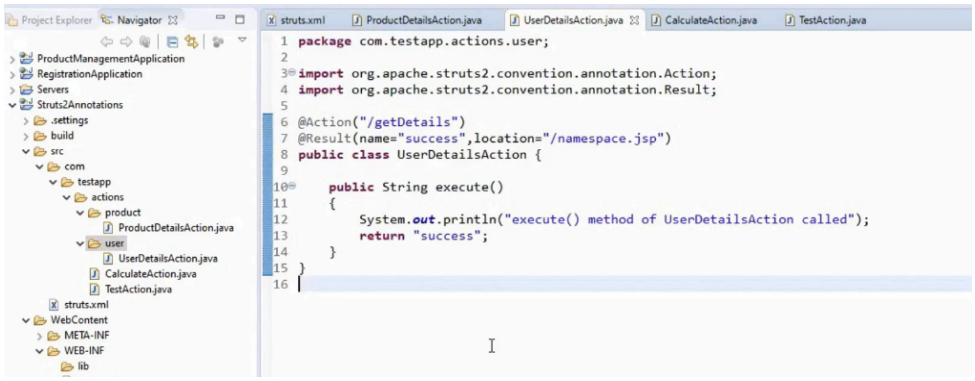


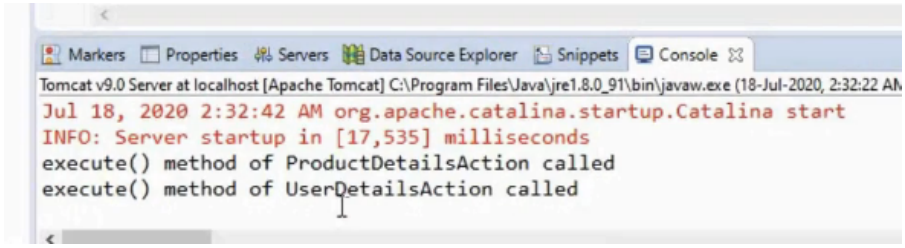
Lets us do the same thing with annotations. Because of the project structure namespaces will be created as shown below.


workspaceStruts2 - Struts2Annotations/src/com/testapp/actions/product/ProductDetailsAction.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer Navigator
ProductManagementApplication
RegistrationApplication
Servers
Struts2Annotations
.settings
build
src
com
testapp
actions
product
ProductDetailsAction.java
user
UserDetailsAction.java
CalculateAction.java
TestAction.java
struts.xml
WebContent
META-INF
WEB-INF
struts.xml ProductDetailsAction.java UserDetailsAction.java CalculateAction.java TestAction.java
1 package com.testapp.actions.product;
2
3 public class ProductDetailsAction {
4
5 public String execute()
6 {
7 System.out.println("execute() method of ProductDetailsAction called");
8 return "success";
9 }
10 }
11
/product

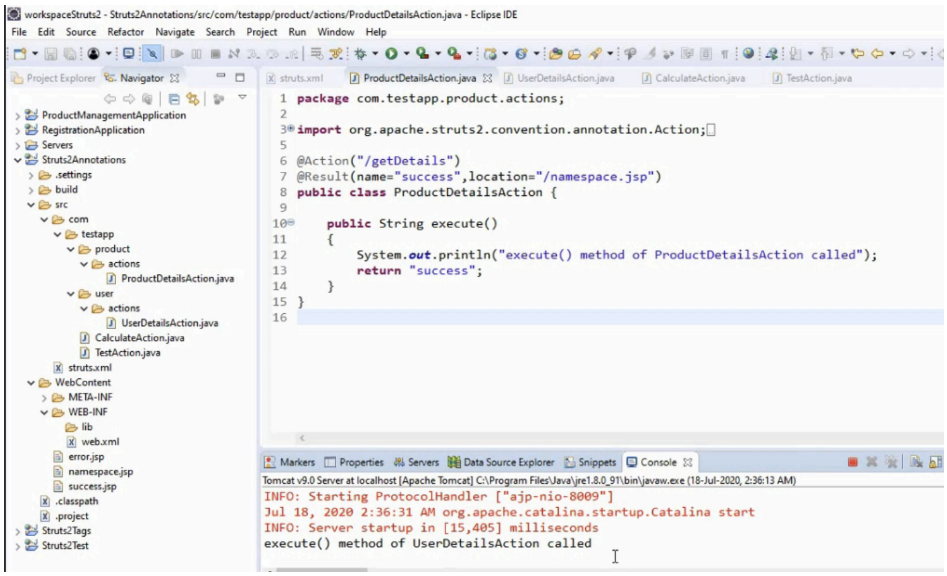

actions
product
ProductDetailsAction.java
user
UserDetailsAction.java
CalculateAction.java
TestAction.java
struts.xml
10 }
11
/user


workspaceStruts2 - Struts2Annotations/src/com/testapp/actions/product/ProductDetailsAction.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer Navigator
ProductManagementApplication
RegistrationApplication
Servers
Struts2Annotations
.settings
build
src
com
testapp
actions
product
ProductDetailsAction.java
user
UserDetailsAction.java
CalculateAction.java
TestAction.java
struts.xml
WebContent
META-INF
WEB-INF
struts.xml ProductDetailsAction.java UserDetailsAction.java CalculateAction.java TestAction.java
1 package com.testapp.actions.product;
2
3 import org.apache.struts2.convention.annotation.Action;
4 import org.apache.struts2.convention.annotation.Result;
5
6 @Action("/getDetails")
7 @Result(name="success",location="/namespace.jsp")
8 public class ProductDetailsAction {
9
10 public String execute()
11 {
12 System.out.println("execute() method of ProductDetailsAction called");
13 return "success";
14 }
15 }
16


Project Explorer Navigator
ProductManagementApplication
RegistrationApplication
Servers
Struts2Annotations
.settings
build
src
com
testapp
actions
product
ProductDetailsAction.java
user
UserDetailsAction.java
CalculateAction.java
TestAction.java
struts.xml
WebContent
META-INF
WEB-INF
lib
struts.xml ProductDetailsAction.java UserDetailsAction.java CalculateAction.java TestAction.java
1 package com.testapp.actions.user;
2
3 import org.apache.struts2.convention.annotation.Action;
4 import org.apache.struts2.convention.annotation.Result;
5
6 @Action("/getDetails")
7 @Result(name="success",location="/namespace.jsp")
8 public class UserDetailsAction {
9
10 public String execute()
11 {
12 System.out.println("execute() method of UserDetailsAction called");
13 return "success";
14 }
15 }
16

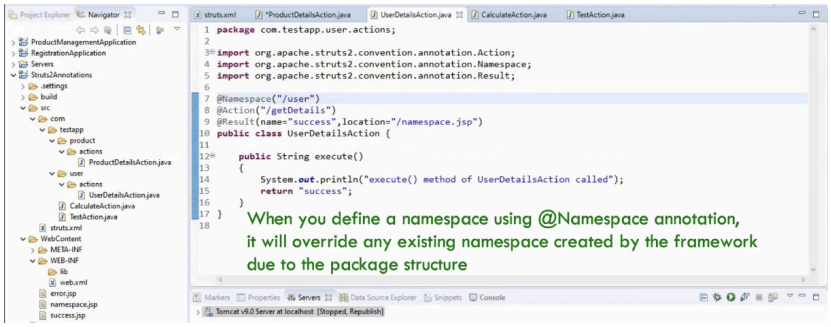


If namespace does not happen automatically follow the below folder structure.
Its developer interest to creating a namespace automatically or creating a custom name space.

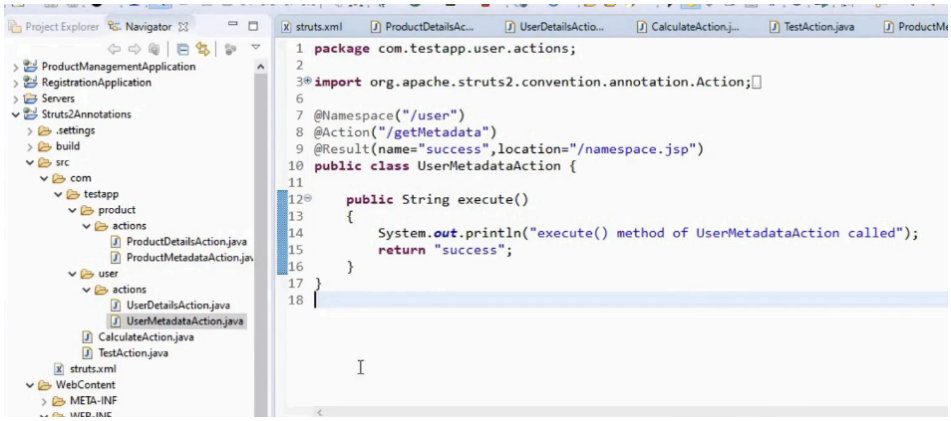


With the above folder structure namespace wont work

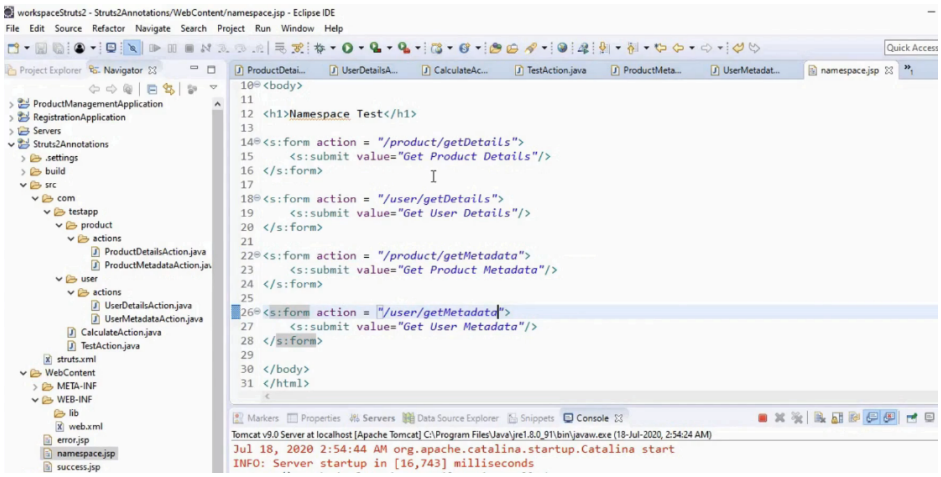
Use namespace annotation as shown below.



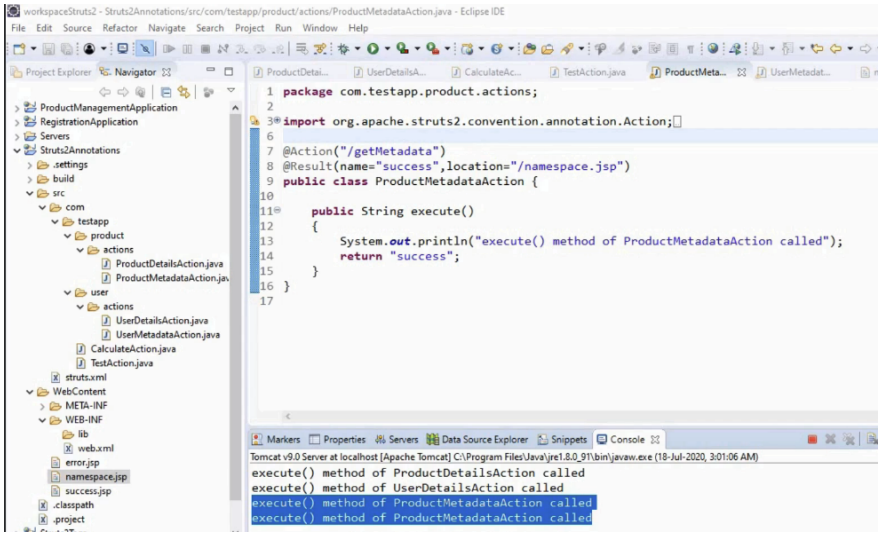
Create user and product metadata classes as shown below

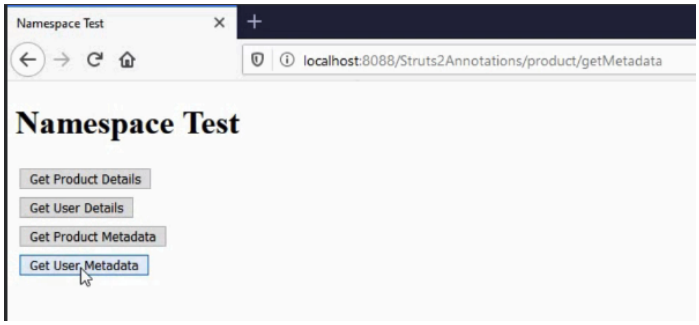


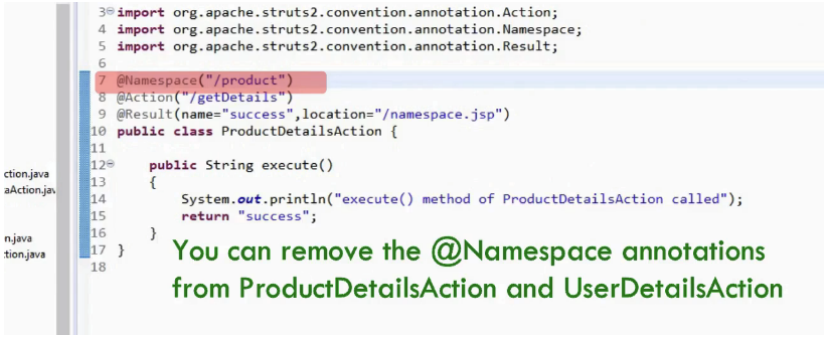
Include something in the namespace



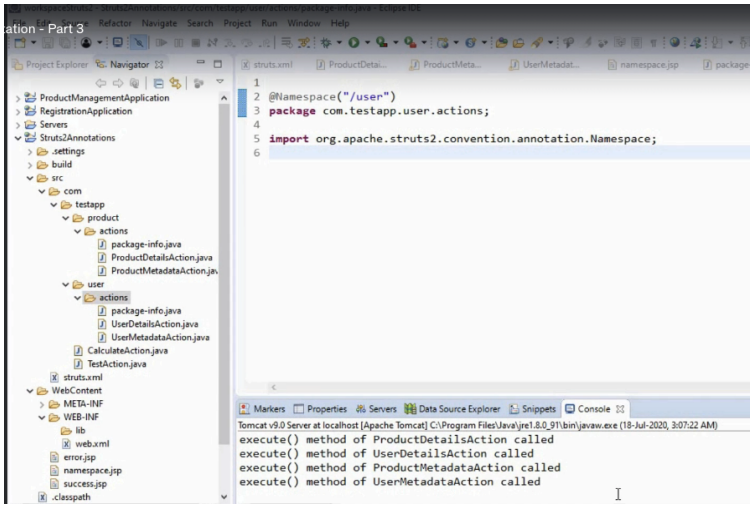
Using Action annotation we can override namesapce annotations as shown below



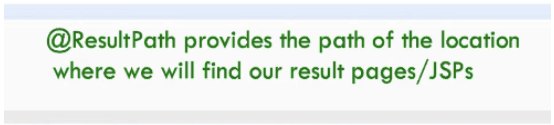




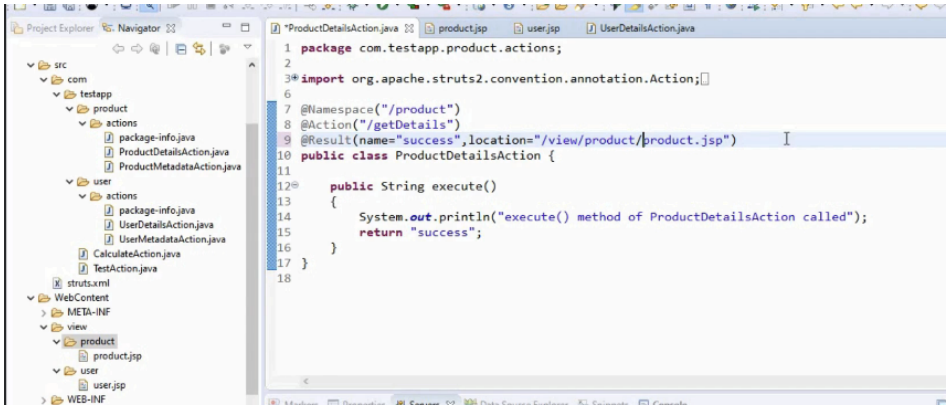
Add it in the info package-info.java classes as shown below. This is applicable at package level for action classes



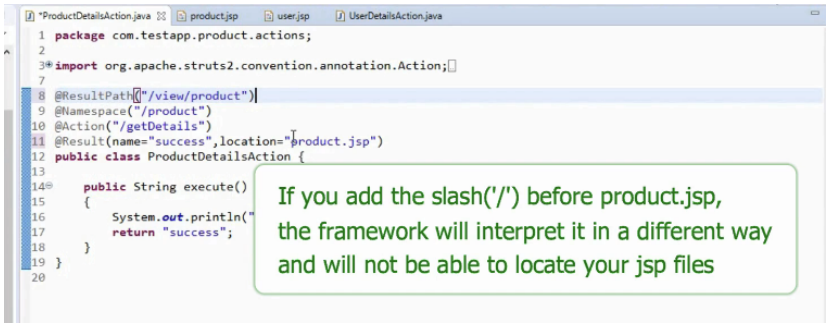
Result path annotation:



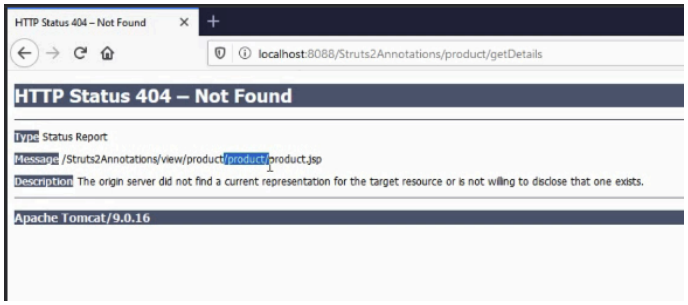
Create view folder and subfolder and place corresponding jsp pages and specify complete path at result annotation.



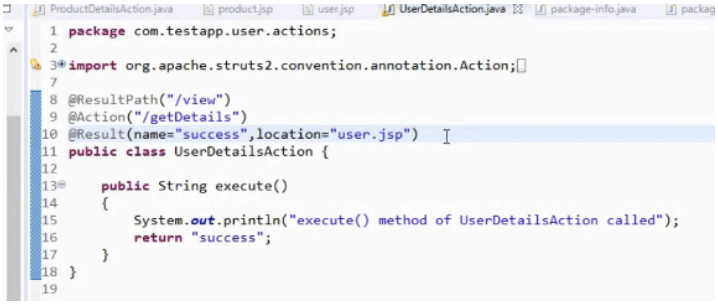
Another approach is use @ResultPath



If you up and running the application we get 404 errors because of the path it forms. Remove '/product' either from @ResultPath or @Namespace annotation. Refer below screenshot.

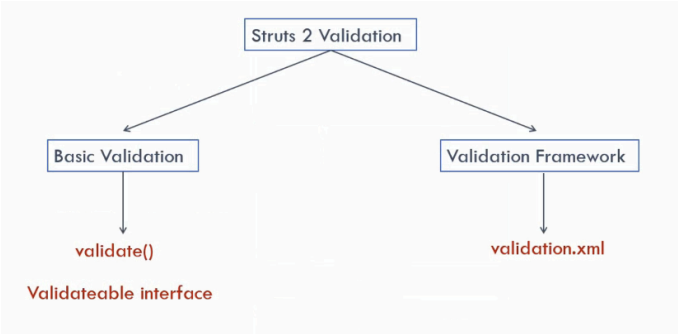


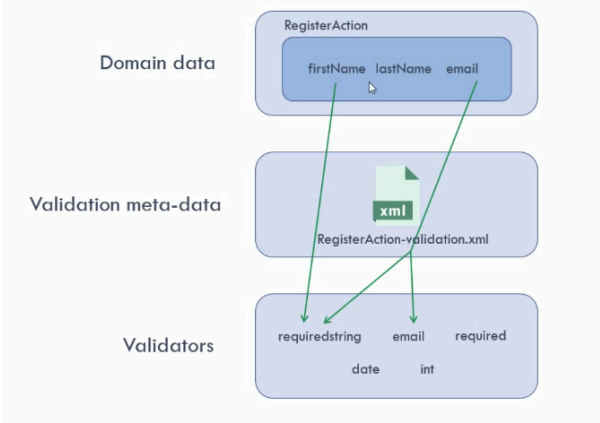
So follow the below approach. It will work for us.

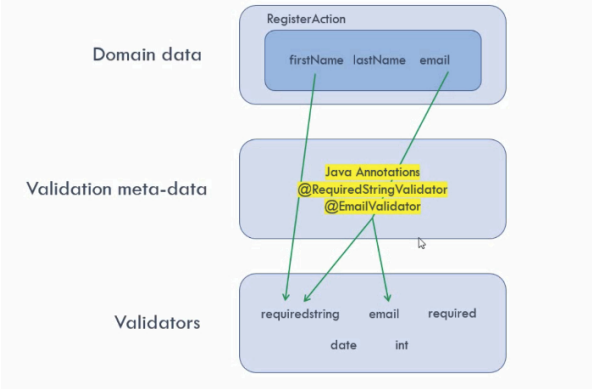


```java
package com.testapp.user.actions;

import org.apache.struts2.convention.annotation.Action;

@ResultPath("/view")
@Action("/getDetails")
@Result(name="success",location="user.jsp")
public class UserDetailsAction {

    public String execute()
    {
        System.out.println("execute() method of UserDetailsAction called");
        return "success";
    }
}
```

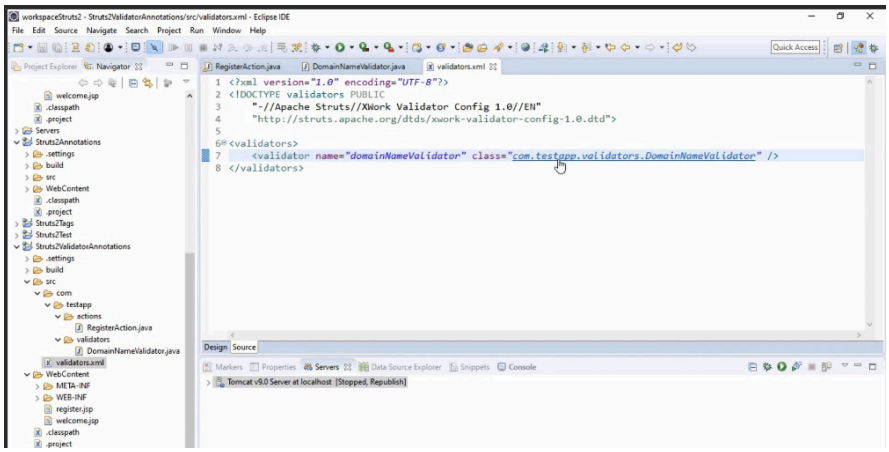
Validation annotations:



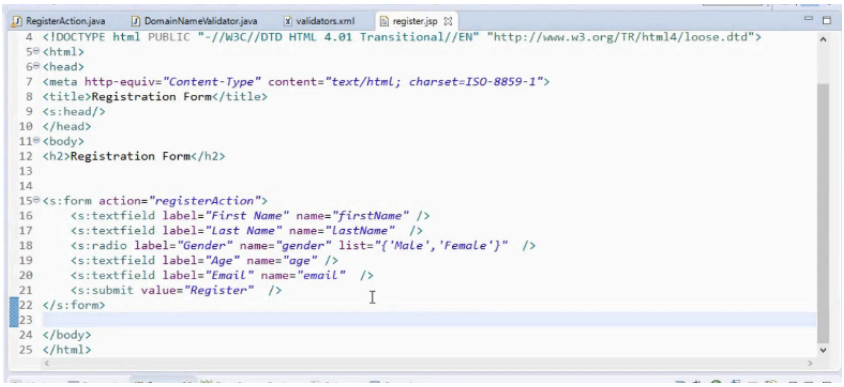




Create new project and maintain folder structure like below, copy required path and update validators.xml



And register.jsp page.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Registration Form</title>
<s:head/>
</head>
<body>
<h2>Registration Form</h2>


<s:form action="registerAction">
    <s:textfield label="First Name" name="firstName" />
    <s:textfield label="Last Name" name="lastName" />
    <s:radio label="Gender" name="gender" list="{'Male','Female'}" />
    <s:textfield label="Age" name="age" />
    <s:textfield label="Email" name="email" />
    <s:submit value="Register" />
</s:form>

</body>
</html>
```

Let us implement below validations using annotations

RegisterAction.java | DomainNameValidator.java | validators.xml | register.jsp | RegisterAction-validation.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE validators PUBLIC "-//Apache Struts//XWork Validator 1.0.3//EN"
 "http://struts.apache.org/dtds/xwork-validator-1.0.3.dtd">

<validators>
    <validator type="requiredstring">
        <param name="fieldName">firstName</param>
        <message key="error.firstName.required" />
    </validator>
    <validator type="requiredstring">
        <param name="fieldName">lastName</param>
        <message key="error.lastName.required" />
    </validator>
    <validator type="required">
        <param name="fieldName">gender</param>
        <message key="error.gender.required" />
    </validator>
    <validator type="required">
        <param name="fieldName">age</param>
        <message key="error.age.required" />
    </validator>
```

RegisterAction.java | DomainNameValidator.java | validators.xml | register.jsp | RegisterAction-validation.xml | global.properties

```
        <param name="fieldName">age</param>
        <message key="error.age.required" />
    </validator>
    <validator type="int">
        <param name="fieldName">age</param>
        <param name="min">18</param>
        <message key="error.age.range" />
    </validator>
    <field name="email">
        <field-validator type="requiredstring">
            <message key="error.email.required" />
        </field-validator>
        <field-validator type="email">
            <message key="error.email.valid" />
        </field-validator>
        <field-validator type="domainNameValidator">
            <param name="domainName">gmail.com</param>
            <message key="error.email.validdomain" />
        </field-validator>
    </field>
</validators>
```

Design | Source

RegisterAction.java | DomainNameValidator.java | validators.xml | register.jsp | RegisterAction-validation.xml | global.properties

```
@RequiredStringValidator(type=ValidatorType.SIMPLE, fieldName="firstName", message="First name is required")
public String getFirstName() {
    return firstName;


public void setFirstName(String firstName) {
    this.firstName = firstName;


@RequiredStringValidator(type=ValidatorType.SIMPLE, fieldName="lastName", message="Last name is required")
public String getLastName() {
    return lastName;


public void setLastName(String lastName) {
    this.lastName = lastName;


@RequiredFieldValidator(type=ValidatorType.SIMPLE, fieldName="gender", message="Gender is required")
public String getGender() {
    return gender;
```

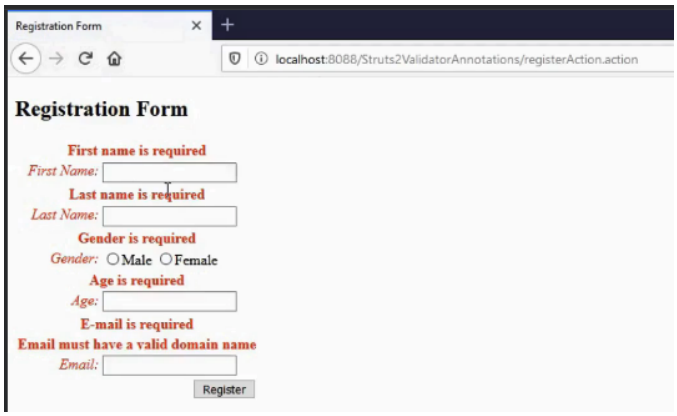
Markers | Properties | Servers | Data Source Explorer | Snippets | Console

Tomcat v9.0 Server at localhost [Stopped, Republish]

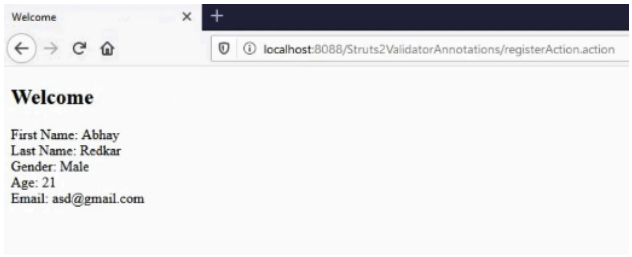
```java
public void setLastName(String lastName) {
    this.lastName = lastName;
}

@RequiredFieldValidator(type=ValidatorType.SIMPLE, fieldName="gender", message="Gender is required")
public String getGender() {
    return gender;
}

public void setGender(String gender) {
    this.gender = gender;
}

@RequiredFieldValidator(type=ValidatorType.SIMPLE, fieldName="age", message="Age is required")
@IntRangeFieldValidator(type=ValidatorType.SIMPLE, fieldName="age", min="18", message="Age should be above
public Integer getAge() {
    return age;
}

public void setAge(Integer age) {
    this.age = age;
}
```

```java
@RequiredStringValidator(type=ValidatorType.SIMPLE, fieldName="email", message="E-mail is required")
@EmailValidator(type=ValidatorType.SIMPLE, fieldName="email", message="Must provide a valid email")
@CustomValidator(type="domainNameValidator", fieldName="email", parameters = { @ValidationParameter(name="
public String getEmail() {
    return email;
}
```

```java
ge="E-mail is required")
provide a valid email")
= { @ValidationParameter(name="domainName",value="gmail.com")}, message="Email must have a valid domain name")
```



Try with valid data you get below page.

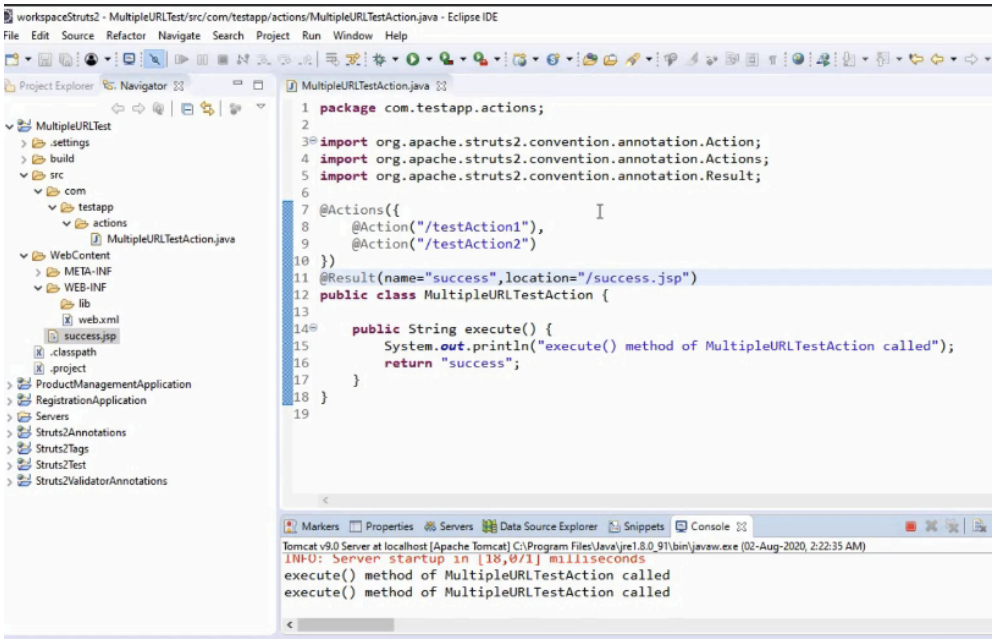


Another way of doing validators is moving them above class name as shown below.

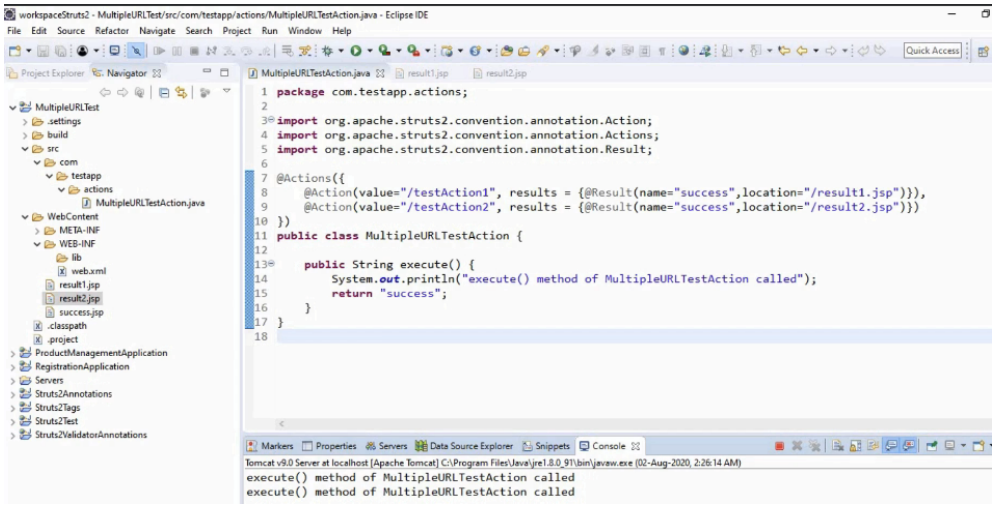
```
RegisterAction.java | DomainNameValidator.java | validators.xml | register.jsp | RegisterAction-validation.xml | global.properties
@Results({
    @Result(name="success",location="/welcome.jsp"),
    @Result(name="input",location="/register.jsp")
})
@Validations(
        requiredStrings = {
                @RequiredStringValidator(type=ValidatorType.SIMPLE, fieldName="firstName", message="First name is r
                @RequiredStringValidator(type=ValidatorType.SIMPLE, fieldName="lastName", message="Last name is rec
                @RequiredStringValidator(type=ValidatorType.SIMPLE, fieldName="email", message="E-mail is required"
        },
        requiredFields = {
                @RequiredFieldValidator(type=ValidatorType.SIMPLE, fieldName="gender", message="Gender is required"
                @RequiredFieldValidator(type=ValidatorType.SIMPLE, fieldName="age", message="Age is required")
        },
        emails = {
                @EmailValidator(type=ValidatorType.SIMPLE, fieldName="email", message="Must provide a valid email")
        },
        intRangeFields = {
                @IntRangeFieldValidator(type=ValidatorType.SIMPLE, fieldName="age", min="18", message="Age should b
        },
        customValidators = {
                @CustomValidator(type="domainNameValidator", fieldName="email", parameters = { @ValidationParameter
```

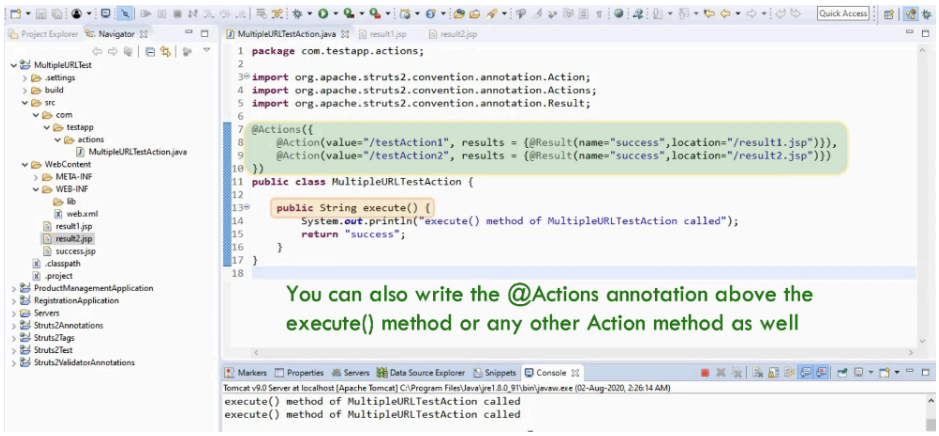
Actions Annotation:

For both urls we get success page



Same action executed above and below. Where above we returned same page below we returned two pages


workspaceStruts2 - MultipleURLTest/src/com/testapp/actions/MultipleURLTestAction.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explorer Navigator
MultipleURLTest
.settings
build
src
com
testapp
actions
MultipleURLTestAction.java
WebContent
META-INF
WEB-INF
lib
web.xml
result1.jsp
result2.jsp
success.jsp
.classpath
.project
ProductManagementApplication
RegistrationApplication
Servers
Struts2Annotations
Struts2Tags
Struts2Test
Struts2ValidatorAnnotations
MultipleURLTestAction.java
result1.jsp
result2.jsp
1 package com.testapp.actions;
2
3 import org.apache.struts2.convention.annotation.Action;
4 import org.apache.struts2.convention.annotation.Actions;
5 import org.apache.struts2.convention.annotation.Result;
6
7 @Actions({
8 @Action(value="/testAction1", results = {@Result(name="success",location="/result1.jsp")}),
9 @Action(value="/testAction2", results = {@Result(name="success",location="/result2.jsp")})
10 })
11 public class MultipleURLTestAction {
12
13 public String execute() {
14 System.out.println("execute() method of MultipleURLTestAction called");
15 return "success";
16 }
17 }
18
Markers Properties Servers Data Source Explorer Snippets Console
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (02-Aug-2020, 2:26:14 AM)
execute() method of MultipleURLTestAction called
execute() method of MultipleURLTestAction called


Quick Access
Project Explorer Navigator
MultipleURLTest
.settings
build
src
com
testapp
actions
MultipleURLTestAction.java
WebContent
META-INF
WEB-INF
lib
web.xml
result1.jsp
result2.jsp
success.jsp
.classpath
.project
ProductManagementApplication
RegistrationApplication
Servers
Struts2Annotations
Struts2Tags
Struts2Test
Struts2ValidatorAnnotations
MultipleURLTestAction.java
result1.jsp
result2.jsp
1 package com.testapp.actions;
2
3 import org.apache.struts2.convention.annotation.Action;
4 import org.apache.struts2.convention.annotation.Actions;
5 import org.apache.struts2.convention.annotation.Result;
6
7 @Actions({
8 @Action(value="/testAction1", results = {@Result(name="success",location="/result1.jsp")}),
9 @Action(value="/testAction2", results = {@Result(name="success",location="/result2.jsp")})
10 })
11 public class MultipleURLTestAction {
12
13 public String execute() {
14 System.out.println("execute() method of MultipleURLTestAction called");
15 return "success";
16 }
17 }
18
You can also write the @Actions annotation above the execute() method or any other Action method as well
Markers Properties Servers Data Source Explorer Snippets Console
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_91\bin\javaw.exe (02-Aug-2020, 2:26:14 AM)
execute() method of MultipleURLTestAction called
execute() method of MultipleURLTestAction called