Apache Struts - 1.3

On this page:

Target audience:

Users of the extension providing **Apache Struts** support for the JEE Analyzer.

Summary: This document provides basic information about the extension providing **Apache Struts** support for the JEE Analyzer.

Extension ID

com.castsoftware.struts

What's new?

Please see Apache Struts - 1.3 - Release Notes for more information.

Description

This extension provides support for **Apache Struts** - this support is **in addition** to the basic support provided for Apache Struts in the JEE Analyzer. The extension's main role is to improve the detection of links and transaction computations where Apache Struts is implicated.

Note that:

- At the current time, analysis results will contain duplicated objects: the Struts Operation object (created by the extension) corresponds to the **Struts Action Mapping Operation object** created by the JEE Analyzer - this duplication will be fixed in a future release.
- All Transaction results involving Apache Struts code are generated using the Struts Operation object (created by the extension) and **not** the **Struts Action Mapping Operation object** created by the JEE Analyzer.

In what situation should you install this extension?

If your JEE application source code uses the **Apache Struts framework** you should install this extension to benefit from improved support for Apache Struts.

Features

Object structure and links

- Creates **Struts Operations** which represent entry-points for web services. For each operation, one Struts Operation object is created.
- The dependent Web Services Linker is responsible for detecting and creating links between the UI (JSP/HTML etc) to the Struts Operation object created by the extension.

Struts 1.1 support

Application configuration

In the "web.xml" file, the extension will detect the servlet pointing to "org.apache.struts.action.ActionServlet" class, indicating that Struts 1.1 is being used. The extension will detect the servlet mapping (i.e. "***.perform**" in the example below):

```
<web-app id="WebApp">
...
        <servlet>
            <servlet-name>action</servlet-name>
            <display-name>Struts Action</display-name>
            <servlet-class>org.apache.struts.action.ActionServlet</servlet-class>
....
        </servlet>
...
    <servlet-mapping>
            <servlet-name>action</servlet-name>
            <url-pattern>*.perform</url-pattern>
    </servlet-mapping>
....
</web-app>
```

In the struts config xml files (struts-XXX.xml), the extension will detect the correspondence between an URL and a java class (i.e. **/logout** corresponds to the "**com.company.lightspeed.struts.logon.StrutsActionLogout**" java class):

```
<struts-config>
...
  <action-mappings>
         <action
        path="/logout"
        type="com.company.lightspeed.struts.logon.StrutsActionLogout">
            <forward name="loggedOut" path="lightspeed/LoggedOut.jsp" />
        </action>
....
  </action-mappings>
</struts-config>
```

The extension will search for the **"execute"** method in the corresponding java class (here **StrutsActionLogout** class) and create a StrutsOperation named with the corresponding URL (here **/logout**). A call link between the StrutsOperation and the **"execute"** method will be created.

Note that the **"execute"** method may be defined in a parent class of **StrutsActionLogout**. This specific case which is common to both struts 1 and struts 2 is described in section Modelling when the execute method is defined in a parent Struts class.

Classes inheriting from org.apache.struts.actions.DispatchAction

The extension will search for all java classes implementing "**org.apache.struts.actions.DispatchAction**".

For example:

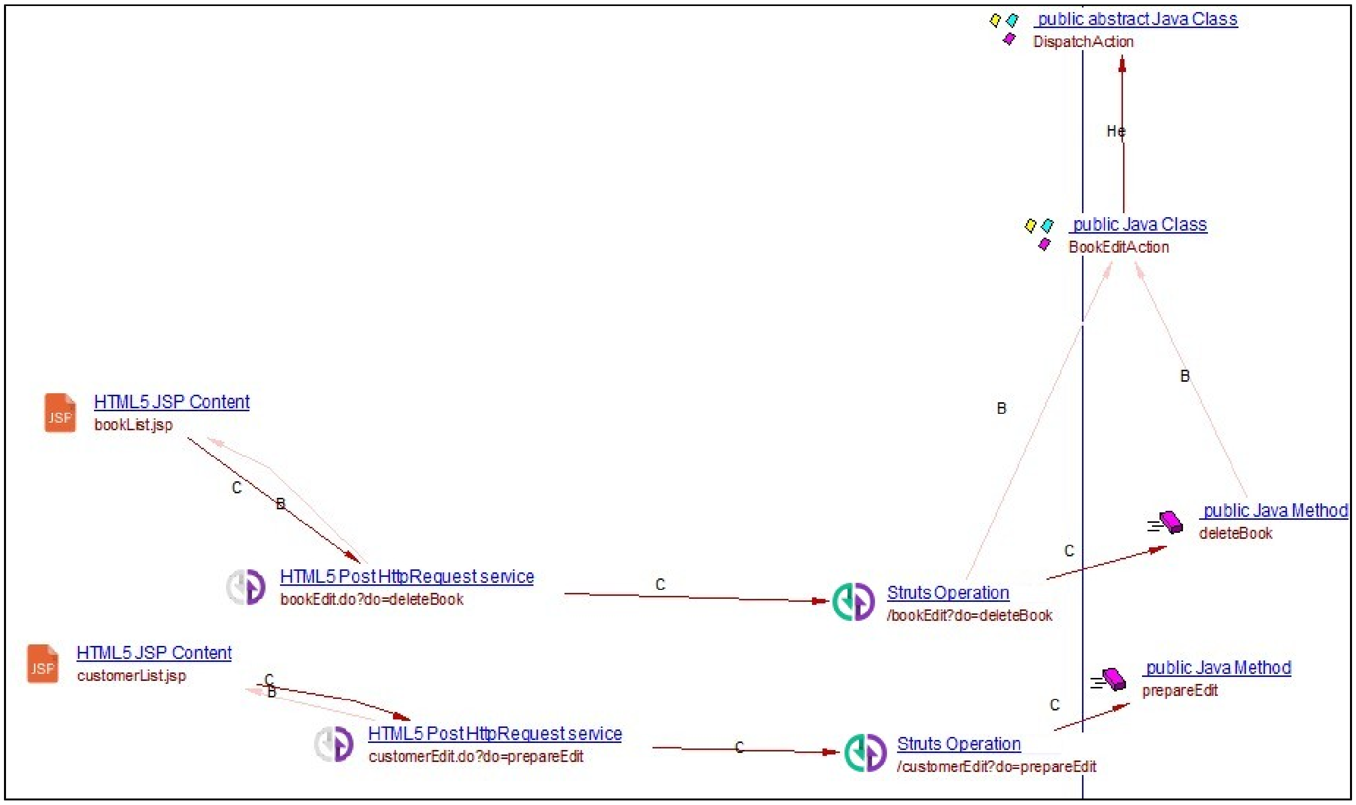
```
import org.apache.struts.actions.DispatchAction;

        public class BookEditAction extends DispatchAction {


    public ActionForward editBook(
            ActionMapping mapping,
            ActionForm form,
            HttpServletRequest request,
            HttpServletResponse response) {
            BookEditForm bookEditForm = (BookEditForm) form;

            /* lalinuna.de 04.11.2004
             * get id of the book from request
             */
            Integer id = Integer.valueOf(request.getParameter("id"));
                    // [laliluna] 28.11.2004  get business logic
            LibraryManager libraryManager = new LibraryManager();
            bookEditForm.setBook(libraryManager.getBookByPrimaryKey(id));
            return mapping.findForward("showEdit");
    }

...
}
```

We create a **Struts Operation** with the name "**/BookEdit?do=editBook**" (this name is found in the struts web xml file or, by default, is the class name without the Action part), and a call link to the "**editBook**" method. The "do" after the "?" is found in the struts web xml file in the "parameter" field of the "action" tag. The same is done for every method of the class.

Example of a struts-config.xml:

```
<!DOCTYPE struts-config PUBLIC "-//Apache Software Foundation//DTD Struts Configuration 1.2//EN" "http://struts.
apache.org/dtds/struts-config_1_2.dtd">
<struts-config>
   <action-mappings >
      <action
         attribute="bookEditForm"
         input="/jsp/bookEdit.jsp"
         name="bookEditForm"
         parameter="do"
         path="/bookEdit"
         scope="request"
         type="com.library.struts.action.BookEditAction">
         <forward name="showBorrow" path="/jsp/borrowBook.jsp" />
         <forward name="showEdit" path="/jsp/bookEdit.jsp" />
         <forward
            name="showList"
            path="/bookList.do"
            redirect="true" />
         <forward name="showAdd" path="/jsp/bookAdd.jsp" />
      </action>

   </action-mappings>
</struts-config>}
```

Results:



Classes inheriting from net.jspcontrols.dialogs.actions.SelectAction

The extension will search for all java classes implementing "net.jspcontrols.dialogs.actions.SelectAction".

For example:

```
public class MaintainMyAccountAction extends SelectAction{


         protected Map getKeyMethodMap() {
                Map map = new HashMap();
                map.put(getInitKey() + "-GET", "getfoo");
                map.put(getInitKey() + "-SAVE", "save");
                map.put(getInitKey() + "-CANCEL", "cancel");
                return map;
        }

    public ActionForward getfoo(ActionMapping mapping,
                 ActionForm form,
                HttpServletRequest request,
                HttpServletResponse response)
                throws IOException, ServletException {
                 ActionForward forward = null;
                 UserForm userForm = (UserForm)form;

                 UserVO userVO = (UserVO) UserSecurityManager.getUserContainer(request);
                 if(userVO != null) CopyUtils.copyProperties(userForm,userVO);

                 forward = mapping.findForward(Constant.SUCCESS);
                 return(forward);
         }


...
}
```

In a SelectAction class, the method getKeyMethodMap provides a mapping between the methods and a parameter name. In the previous example, the method **"getfoo"** is mapped with a parameter **DIALOG-EVENT-GET** (DIALOG-EVENT is the output of getInitKey()).

We create a **Struts Operation** with the name "**/MaintainMyAccount?DIALOG-EVENT-GET**" (this name is found in the struts web xml file or, when no reference to this class is found in the web xml file, this name is set to the class name without the Action part), and a call link to the "**getfoo**" method.

Struts 1.x + Spring

DelegatingActionProxy

Spring provides a special action class **org.springframework.web.struts.DelegatingActionProxy** that acts as a proxy for struts actions, allowing more indirections:

```
<struts-config>
...
  <action-mappings>
         <action path="/user" type="org.springframework.web.struts.DelegatingActionProxy">
....
  </action-mappings>
</struts-config>
```

The class name is then found in a bean:

```
<bean name="/user" scope="prototype" autowire="byName" class="org.example.web.UserAction"/>
```

or in a bean alias.

```
<bean name="/someBean" scope="prototype" autowire="byName" class="org.example.web.UserAction"/>
<alias alias="/user" name="/someBean"/>
```

Struts 2.x support

Application configuration

In the "web.xml" file, the extension will detect the filter pointing to "org.apache.struts2.dispatcher.FilterDispatcher" class, indicating that Struts 2.x is being used. The extension will detect the servlet mapping (i.e. "***.action;/***" in the example below). The extension will also obtain the filter "**struts2**" and "**BaseStrutsFilter**" because "**BaseStrutsFilter**" points to a class which exists in the analysis.

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:
jsp="http://java.sun.com/xml/ns/javaee/jsp" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" xsi:
schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="
WebApp_ID" version="2.5">
...
        <filter>
                <filter-name>struts2</filter-name>
                <filter-class>org.apache.struts2.dispatcher.FilterDispatcher</filter-class>
                <init-param>
                        <param-name>actionPackages</param-name>
                        <param-value>com.mapa.sipe.apresentacao.action, br.gov.mapa.arquitetura.apresentacao.
struts</param-value>
                </init-param>
        </filter>

        <filter>
                <display-name>BaseStrutsFilter</display-name>
                <filter-name>BaseStrutsFilter</filter-name>
                <filter-class>com.mapa.sipe.apresentacao.filter.BaseStrutsSipeFilter</filter-class>
                <init-param>
                      <param-name>public-url</param-name>
                      <param-value>/sipe/imprimirGruPublica.action</param-value>
            </init-param>
        </filter>

        <filter-mapping>
                <filter-name>BaseStrutsFilter</filter-name>
                <url-pattern>*.action</url-pattern>
                <dispatcher>REQUEST</dispatcher>
                <dispatcher>FORWARD</dispatcher>
        </filter-mapping>

        <filter-mapping>
                <filter-name>struts2</filter-name>
                <url-pattern>/*</url-pattern>
                <dispatcher>REQUEST</dispatcher>
        <dispatcher>ERROR</dispatcher>
        </filter-mapping>
....
</web-app>
```

If a struts config file exists as shown below, the extension can obtain a correspondence between the path and a java class: in the example below, **name="Login"** gives the name of the operation that will be created:

```
<struts>
...
    <package name="chapterSixPublic" namespace="/chapterSix" extends="struts-default">

                <action name="Login" class="manning.chapterSix.Login">
                                        <result type="redirect">/chapterSix/secure/AdminPortfolio.action<
/result>
                                        <result name="input">/chapterSix/Login.jsp</result>
        </action>

....
    </package>

</struts>
```

The extension will search for the **"execute"** method in the corresponding java class (here **manning.chapterSix.Login** class) and create a StrutsOperation named with the corresponding URL (here **/Login**). A call link between the StrutsOperation and the **"execute"** method will be created.

The **"execute"** method may be defined in a parent class of **StrutsActionLogout**. This specific case which is common to both struts 1 and struts 2 is described in section Modelling when the execute method is defined in a parent Struts class.

Classes with @Results annotation

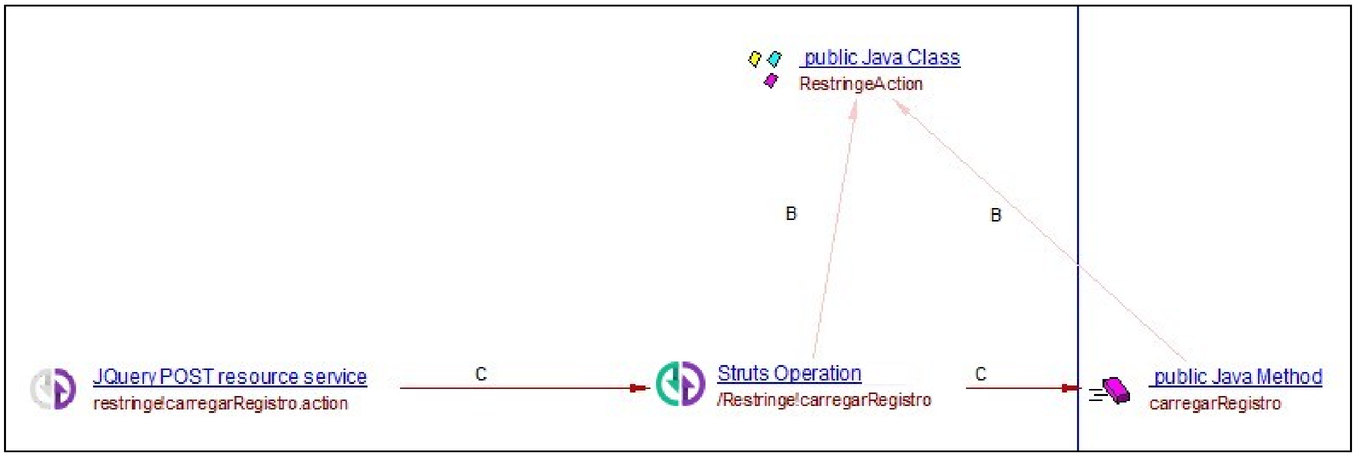
The extension will search for all java classes using the "**@Results**" annotation which is resolved to "**org.apache.struts2.config.Results**".

For example:

```
@Results({@Result(name = "popup", value = "popUpRestringe", type = TilesResult.class),
        @Result(name = "popupRegistro", value = "popUpRestringeRegistro", type = TilesResult.class),
        @Result(name = "base", value = "base", type = ActionChainResult.class, params = {"method", "execute"})})
public class RestringeAction extends SipeBaseAction {

   public String carregarRegistro() throws Exception {
        VwPessoaVinculadaEstab vw = montarVwPessoaVinculadaEstab();
        if (vw != null) {
            listaDadosRegistro =
                getSipeComumBD().consultarRegistroCadastro(vw.getIdPessoaEstabelecimento(), getJAASPrincipal().
getIdPessoaFisica(), getIdAreaSelecionada(),
                    getPreCadastroAtivo(), getStPerfil());
            for (RegistroEstabCadDTO dados : listaDadosRegistro) {
                if (dados.getCdRegistroCadastro().equals(getRegistroSelecionado())) {
                    vw.setDadosRegistro(dados);
                    recuperarInformacoesUsuarioLogado(vw);
                    break;
                }
            }
            adicionarEstabelecimentoSessao(vw);
        }
        return FW_BASE;
    }

}
```

The extension will then search for all methods which are public and have "String" as the return type. The extension will create a **Struts Operation** with the name "**/Restringe!carregarRegistro**" (Restringe is the class name without the ending "Action"), and a call link to the method.

For example:


public Java Class
RestringeAction
B
B
JQuery POST resource service
restringe!carregarRegistro.action
C
Struts Operation
/Restringe!carregarRegistro
C
public Java Method
carregarRegistro

Classes implementing SessionAware

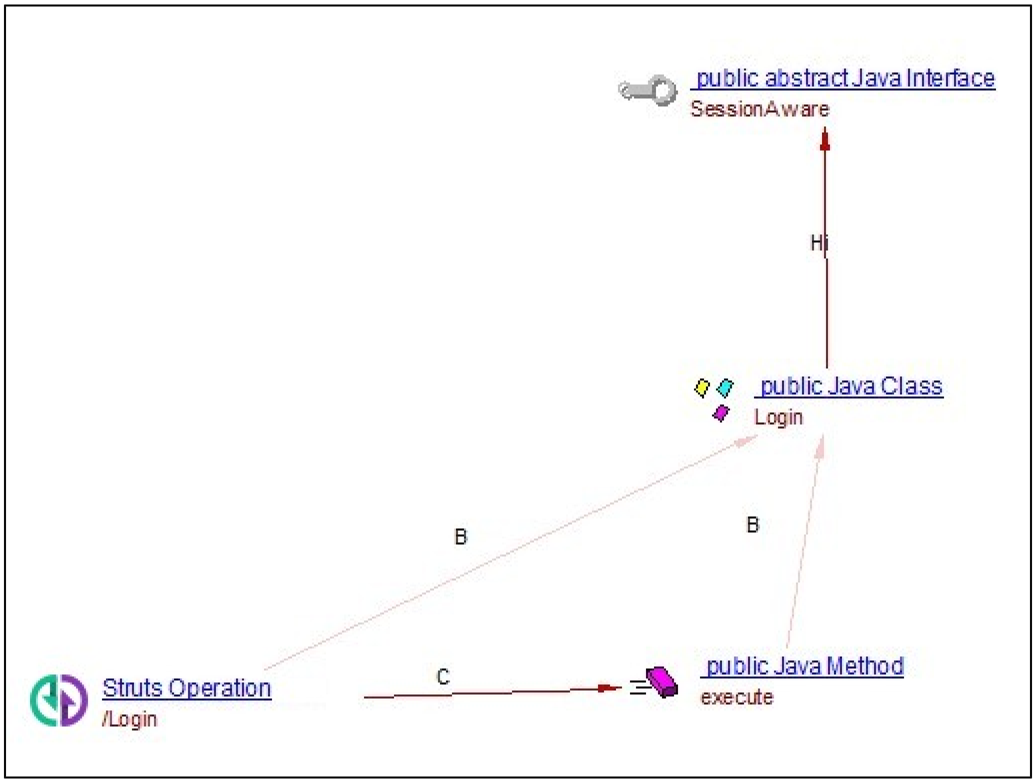
The extension will search for all java classes implementing "**org.apache.struts2.interceptor.SessionAware**".

For example:

```
import org.apache.struts2.interceptor.SessionAware;

public class Login extends ActionSupport implements SessionAware {


        public String execute(){

                User user = getPortfolioService().authenticateUser( getUsername(), getPassword() );
                if ( user == null )
                {
                        /* User not valid, return to input page. */
                        return INPUT;
                }
                else{
                        session.put( Struts2PortfolioConstants.USER, user );
                }

                return SUCCESS;
        }

}
```

The extension will create a **Struts Operation** with the name "**/Login**" (this name is found in a struts xml config file, or the class name by default), and a **call link** to the "**execute**" method.

For example:



Classes inheriting from com.opensymphony.xwork2.ActionSupport

The extension will search for all java classes implementing "**com.opensymphony.xwork2.ActionSupport**".

For example:

```
import com.opensymphony.xwork2.ActionSupport;

public class ManageFieldsPVAction extends ActionSupport{
        private static final long serialVersionUID = -8126525052448102790L;

        private String first_name;
        private String last_name;

        public String execute() {
                //if (first_name==null)
                //        return "login";
                //setFirst_name(first_name+=" with added thingies from your favorite Action!!!");
                return "success";

        }
...
}
```

The extension will create a **Struts Operation** with the name "**/ManageFieldsPV**" (this name is found in the class name without the Action part), and a call link to the "**execute**" method.

Struts 2.x + Spring

StrutsSpringObjectFactor

Spring offers additional indirection for Struts 2 using **org.apache.struts2.spring.StrutsSpringObjectFactory**.

```
<struts>
...
        <constant name="struts.objectFactory" value="org.apache.struts2.spring.StrutsSpringObjectFactory" />
...
    <package name="chapterSixPublic" namespace="/chapterSix" extends="struts-default">

                <action name="Login" class="myBean">

....
    </package>

</struts>
```

Again the real class will be indicated in a bean.

```
<bean name="myBean" class="manning.chapterSix.Login"/>
```

Modelling when the execute method is defined in a parent Struts class

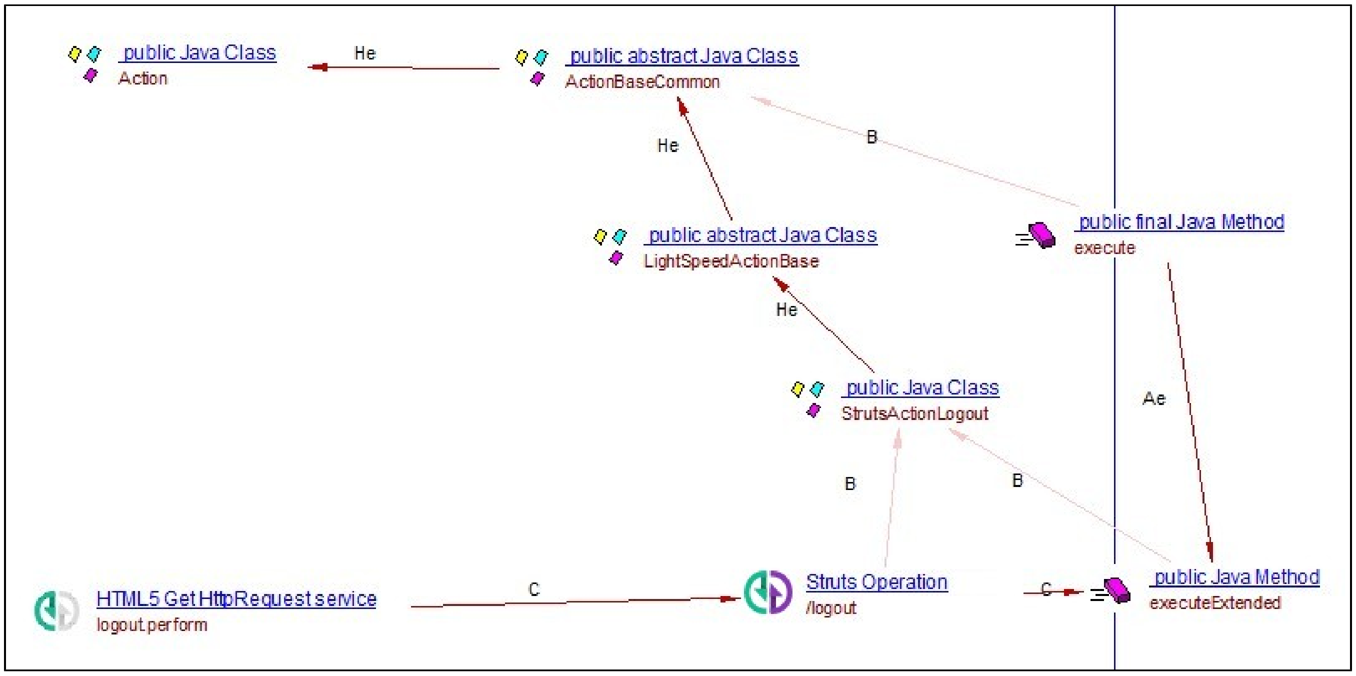
In both struts 1 and struts 2 it is common that the **"execute"** method is not defined within the referred java class but in a parent of that class. This **"execute'** method then calls a method which is overridden in the referred java class. In that case, a call link is created between the Struts Operation and this method.

Let's for instance consider the struts 1 application using the following config xml file (which defines that a **/logout** URL refers to the "**com.company.lightspeed.struts.logon.StrutsActionLogout**" java class):

```
<struts-config>
...
  <action-mappings>
         <action
        path="/logout"
        type="com.company.lightspeed.struts.logon.StrutsActionLogout">
            <forward name="loggedOut" path="lightspeed/LoggedOut.jsp" />
        </action>
....
  </action-mappings>
</struts-config>
```

If no "execute" method is found in "**com.company.lightspeed.struts.logon.StrutsActionLogout**" java class, the extension searches for the "**execute**" method in the class hierarchy.

For example:



In this example, the "**execute**" method is found in the parent java class **ActionBaseCommon.** The extension will analyze this **"execute"** method to see if a method of the "referred" java class (here **StrutsActionLogout**) is called from "**execute**" (here the "**execute**" method calls "**executeExtended**" which is in the "referred" java class **StrutsActionLogout)**. The extension will create a **Struts Operation** named "**/logout**" (name found in the Struts xml file) and a **c all link** between the **operation** and the "**executeExtended**" method.

Supported Apache Struts versions

Apache Struts	Supported
1.x	✓
2.x	✓

Supported client frameworks

Client framework	Supported
jQuery	✓
AngularJS	✓

Function Point, Quality and Sizing support

This extension provides the following support:

- **Function Points (transactions)**: a green tick indicates that OMG Function Point counting and Transaction Risk Index are supported
- **Quality and Sizing**: a green tick indicates that CAST can measure size and that a minimum set of Quality Rules exist

Function Points (transactions)	Quality and Sizing
✓	✓

CAST AIP compatibility

This extension is compatible with:

CAST AIP release	Supported
8.3.x	✓
8.2.x	✓
8.1.x	✓
8.0.x	✓
7.3.4 and all higher 7.3.x releases	✓

Supported DBMS servers

This extension is compatible with the following DBMS servers:

CSS	Oracle	Microsoft
✓	✓	✓

Prerequisites

✓	An installation of any compatible release of CAST AIP (see table above)

Dependencies with other extensions

Some CAST extensions require the presence of other CAST extensions in order to function correctly. The **Apache Struts** extension requires that the following other CAST extensions are also installed:

- JEE Analyzer
- HTML5/JavaScript extension
- Web Services Linker
- com.castsoftware.internal.platform (internal extension)

Note that:

- when using the **CAST Extension Downloader** to download the extension and the **Manage Extensions** interface in **CAST Server Manager** to install the extension, any dependent extensions are **automatically** downloaded and installed for you. You do not need to do anything.
- the JEE Analyzer is not a dependency, but since this extension is always installed with AIP, you do not need to do anything.

For a proper support of SelectAction, use **HTML5/JavaScript 2.0.7** and **Web Services Linker 1.6.7**.

Download and installation instructions

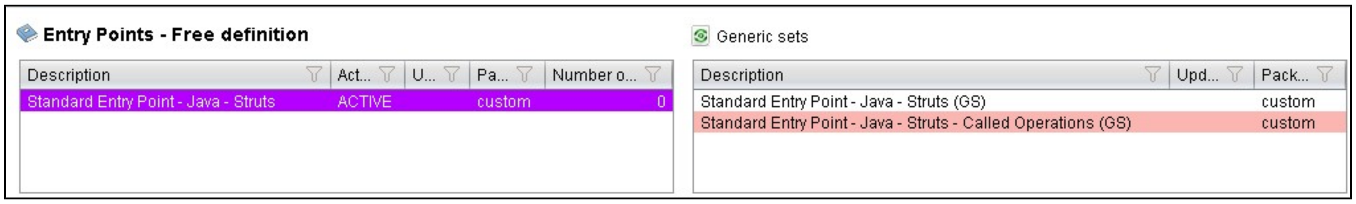
Please see:

- Download an extension
- Install an extension

The latest release status of this extension can be seen when downloading it from the CAST Extend server.

CAST Transaction Configuration Center (TCC) setup

If you are using the extension with **CAST AIP 8.3.x**, an Apache Struts specific setup is **automatically imported** when the extension is installed. These items will be available in the CAST Transaction Configuration Center:



Entry Points - Free definition

Description	Act...	U...	Pa...	Number o...
Standard Entry Point - Java - Struts	ACTIVE		custom	0

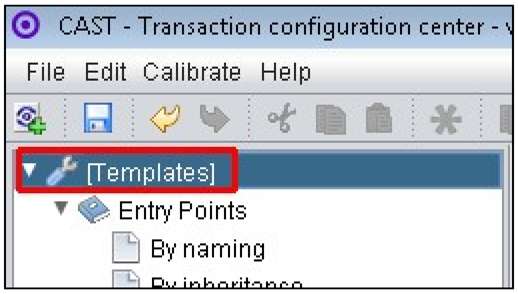
Generic sets

Description	Upd...	Pack...
Standard Entry Point - Java - Struts (GS)		custom
Standard Entry Point - Java - Struts - Called Operations (GS)		custom

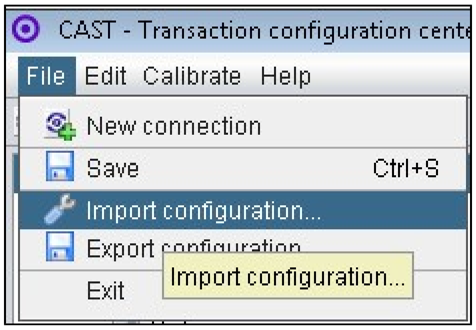
Manual import action for CAST AIP 8.2.x

If you are using the extensions with CAST AIP 8.2.x, you can manually import the TCC setup, should you want to:

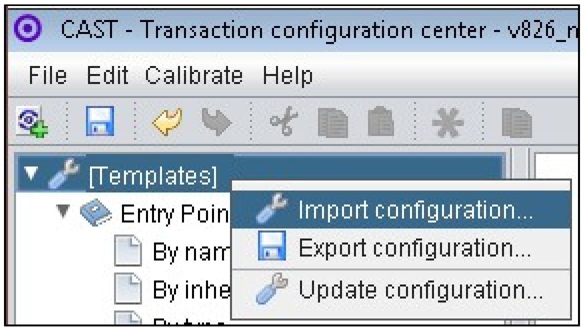
- Locate the .TCCSetup file in the extension folder: **Configuration\TCC\Base_Java_Struts.TCCSetup**
- In the CAST Transaction Configuration Center, ensure you have selected the **Templates** node:



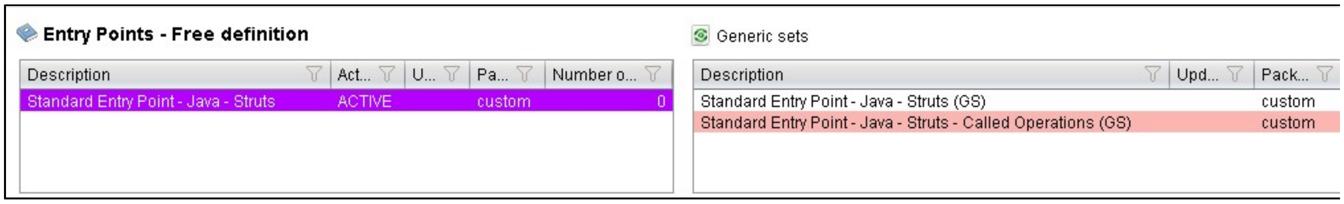
- This .TCCSetup file is to be imported into the CAST Transaction Calibration Center using either the:
  - **File** > **Import Configuration** menu option:



  - Or right clicking on the **Template node** and selecting **Import Configuration**:



- The import of the "**Base_Java_Struts.TCCSetup**" file will provide you with a sample Transaction Entry point in the **Free Definition** node under **Templates**:



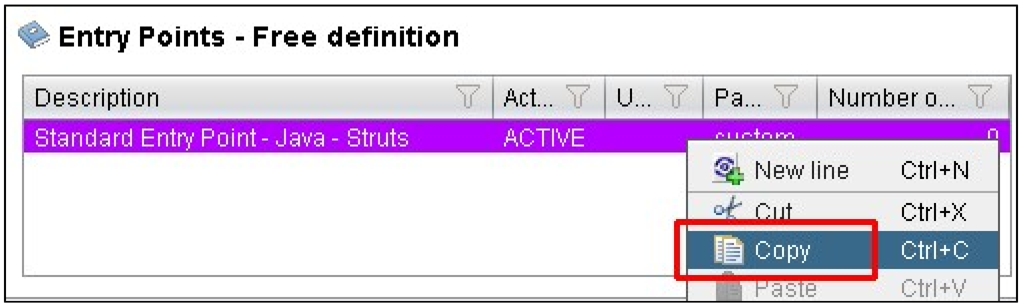
Entry Points - Free definition

Description	Act...	U...	Pa...	Number o...
Standard Entry Point - Java - Struts	ACTIVE		custom	0

Generic sets

Description	Upd...	Pack...
Standard Entry Point - Java - Struts (GS)		custom
Standard Entry Point - Java - Struts - Called Operations (GS)		custom

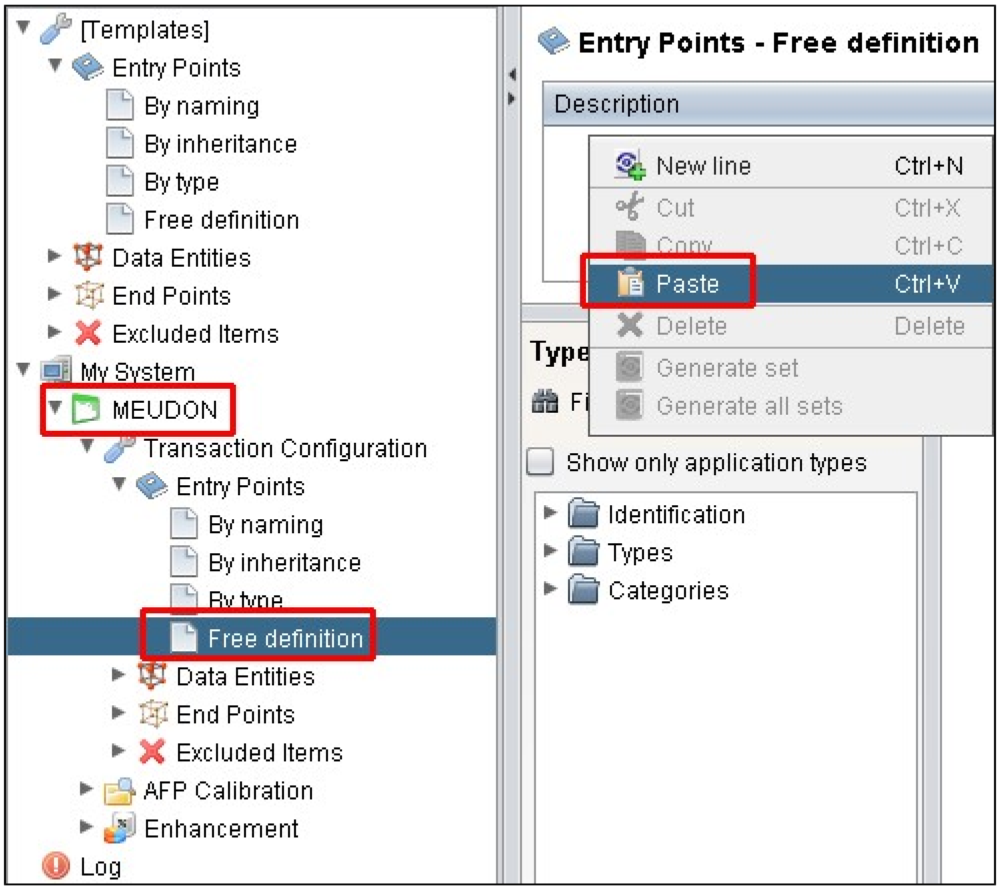
- Now right click the "**Standard Entry Point**" item and select copy:



Entry Points - Free definition

Description	Act...	U...	Pa...	Number o...
Standard Entry Point - Java - Struts	ACTIVE		custom	0

- Paste the item into the **equivalent node** under the **Application**, for example, below we have copied it into the **Application MEUDON**:



- Repeat for any additional items or generic sets that have been imported from the .TCCSetup file.

Packaging, delivering and analyzing your source code

Once the extension is installed, no further configuration changes are required before you can package your source code and run an analysis. The process of packaging, delivering and analyzing your source code does not change in any way:

- **Package and deliver** your application.
- **Analyze** your delivered application source code.

What results can you expect?

Objects

The following objects are displayed in CAST Enlighten:

Icon	Description
	Struts Operation

Current known limitations

- The Class name to URL naming convention of the convention plugin is not supported
- When Dynamic Method Invocation is used, the method **foo** of action **hello** can be accessed with both the urls : hello!foo.action and hello.action?method:foo. Since the recommended approach is to use the first url and in order to limit the number of objects created, the analyzer supports only the first approach.

Structural Rules

The following structural rules are provided:

1.3.1-funcrel	https://technologies.castsoftware.com/rules?sec=srs_struts&ref=||1.3.1-funcrel
1.3.0-funcrel	https://technologies.castsoftware.com/rules?sec=srs_struts&ref=||1.3.0-funcrel
1.3.0-beta1	https://technologies.castsoftware.com/rules?sec=srs_struts&ref=||1.3.0-beta1
1.3.0-alpha2	https://technologies.castsoftware.com/rules?sec=srs_struts&ref=||1.3.0-alpha2
1.3.0-alpha1	https://technologies.castsoftware.com/rules?sec=srs_struts&ref=||1.3.0-alpha1