Direct Struts Action 3 Version 1.0 Component Specification

1. Design

As we progress in upgrading the Direct application, the next thing we are focusing on is the complete flow of launching a contest. This includes everything from entering the basic data for a contest to payment, spec review, provisioning, etc. We currently have a basic flow for launching a contest created in the service layer and being assembled for the html version of Direct. This contest will dig deeper into the process of launching a contest to address the requirements introduced and clarified by the requirements that are referenced in this contest.

This component provides the struts actions to get wiki link, get contest receipt, send contest receipt by email, view game plan, view project's contests, get active contest prize, increase prize of active contest, get active contest schedule, and extend active contest schedule.

1.1 Design Patterns

Actions can be treated like the *Controller* part of the *MVC* pattern.

Actions are also used s*trategically* by the Struts framework.

1.2 Industry Standards

None.

1.3 Required Algorithms

1.3.1 Validation.

Validation here will comprise the user input as specified in the use cases provided in the ARS for each relevant action. Validation checking will be performed using Struts built-in annotations and in setters or in executeAction method when validation cannot be handled by the built-in annotations. For example, such annotation would be added to the setProjectId method of the ContestReceiptRetrievalAction: @FieldExpressionValidator(key = "i18n.contestReceiptRetrievalAction.projectIdValues", fieldName = "projectId", expression = "projectId >= 0 ", message = "projectId must be > 0"). For the valid values see the variable docs and ARS.

1.4 Component Class Overview

ContestReceiptRetrievalAction
This action returns the contest receipt.

ActiveContestScheduleRetrievalAction
This action returns the active contest shedule.

ActiveContestScheduleExtensionAction
This action is used to extend the active contest shedule.

GamePlanRetrievalAction
This action returns the game plan.

ActiveContestPrizeRetrievalAction
This action returns the active contest prize.

ContestReceiptSendingAction
This action sends the contest receipt to the given emails.

ProjectContestDataRetrievalAction
This action returns the project contest and the project summary data.

WikiLinkRetrievalAction
This action returns the wiki link.

ActiveContestPrizeUpdatingAction
This action updates the active contest prize.

1.5 Component Exception Definitions

None.

1.6 Thread Safety

All the actions are not thread safe because they are mutable, but they will be used in a thread safe manner by the Struts framework. JSONResult will be used thread-safely as well.

2. Environment Requirements

2.1 Environment

– Development language: Java1.5, J2EE 1.5
– Compile target: Java1.5, J2EE 1.5
– Application Server: JBoss 4.0.2

2.2 TopCoder Software Components

– Struts Framework 1.0: defines AbstractAction, AggregateDataModel and other classes used in this design.
– Contest Service Façade 1.0: defines ContestServiceFacade and entities used in this design.
– Direct Service Façade 1.0: defines DirectServiceFacade and entities used in this design.

2.3 Third Party Components

- Struts 2.1.8.1 (http://struts.apache.org/2.x/index.html)

3. Installation and Configuration

3.1 Package Name

com.topcoder.direct.services.view.action.contest.launch

3.2 Configuration Parameters

Sample 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>
    <constant name="struts.devMode" value="true" />
    <constant name="struts.convention.action.disableScanning" value="true" />

    <package name="tcs-default" extends="struts-default"
        abstract="true">
        <!-- Setup interceptors stack -->
        <interceptors>
            <interceptor name="validationErrors" class="validationErrorsInterceptor" />
            <interceptor name="mock" class="mockActionInterceptor" />

            <interceptor-stack name="defaultTCSStack">
                <interceptor-ref name="mock" />
                <interceptor-ref name="validationErrors" />
                <interceptor-ref name="defaultStack" />
            </interceptor-stack>
        </interceptors>

        <!--
            Make the default one used for all actions unless otherwise
            configured.
        -->
        <default-interceptor-ref name="defaultTCSStack" />

    </package>

    <package name="default" namespace="/" extends="tcs-default">

        <action name="activeContestPrizeIncreasingAction"
class="activeContestPrizeIncreasingAction">
            <result name="input">/input.jsp</result>
            <result name="success">/success.jsp</result>
        </action>

        <action name="activeContestPrizeRetrievalAction"
class="activeContestPrizeRetrievalAction">
            <result name="input">/input.jsp</result>
            <result name="success">/success.jsp</result>
        </action>

        <action name="activeContestScheduleExtensionAction"
class="activeContestScheduleExtensionAction">
            <result name="input">/input.jsp</result>
            <result name="success">/success.jsp</result>
        </action>

        <action name="activeContestScheduleRetrievalAction"
class="activeContestScheduleRetrievalAction">
            <result name="input">/input.jsp</result>
            <result name="success">/success.jsp</result>
        </action>

        <action name="contestReceiptRetrievalAction"
class="contestReceiptRetrievalAction">
            <result name="input">/input.jsp</result>
            <result name="success">/success.jsp</result>
        </action>

        <action name="contestReceiptSendingAction" class="contestReceiptSendingAction">
            <result name="input">/input.jsp</result>
            <result name="success">/success.jsp</result>
        </action>

        <action name="gamePlanRetrievalAction" class="gamePlanRetrievalAction">
            <result name="input">/input.jsp</result>
            <result name="success">/success.jsp</result>
        </action>

        <action name="projectContestDataRetrievalAction"
class="projectContestDataRetrievalAction">
            <result name="input">/input.jsp</result>
            <result name="success">/success.jsp</result>
        </action>
        <action name="wikiLinkRetrievalAction" class="wikiLinkRetrievalAction">
            <result name="input">/input.jsp</result>
            <result name="success">/success.jsp</result>
        </action>

    </package>

</struts>
```

3.3 Dependencies Configuration

None.

4. Usage Notes

4.1 Required steps to test the component

1. Extract the component distribution.
2. Follow the dependencies configuration.
3. **IMPORTANT NOTES**
    a. The tests in this submission make use of the StrutsSpringTestCase class in order to properly test the annotation validations to make sure they are accurate. The annotation validations are very important to unit test since we can't assume any struts2 annotation expression is correct without a corresponding unit test. All the struts2 dependencies are provided in test_reflib folder.
    b. Note that validations are based on TCUML and ARS, as confirmed by designer in forum: http://forums.topcoder.com/?module=Thread&threadID=678204&start=16
    c. GUI demo is required for this component, instructions are provided below on how to deploy it.
4. Download JBoss 4.0.2 and install it locally .(http://www.jboss.org/jbossas/downloads/).
5. Set your JAVA_HOME to use JDK 1.5 as compiler.
6. Edit build-dependencies.xml and configure jboss.home and junit.jar path.
7. Execute 'ant deployToJBoss' within the directory that the distribution was extracted to in order to deploy the demo to JBoss.
8. For instructions on how to use demo, see section 4.3.
9. Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

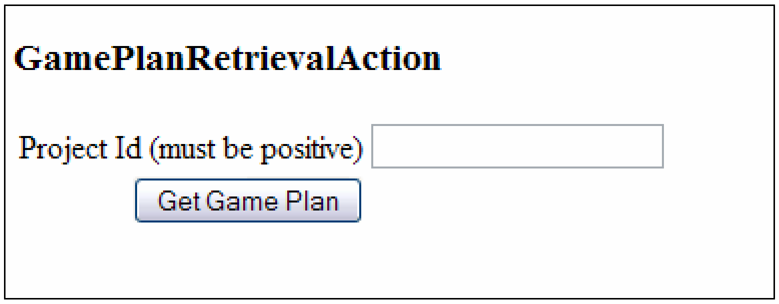
See demo below.

4.3 Demo

1. Be sure you have deployed demo to JBoss, refer to section 4.1 for details.
2. Start JBoss.
3. Navigate to http://localhost:8080/demo/input.jsp

   a. This assumes you use port 8080 for JBoss, you can modify this if you use some other port.

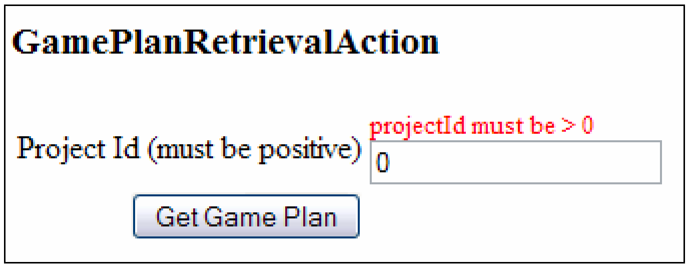
4. You will be presented with the following screen.



GamePlanRetrievalAction

Project Id (must be positive)

5. You can put in a non positive number and press the “Get Game Plan” button and you will get a validation error.

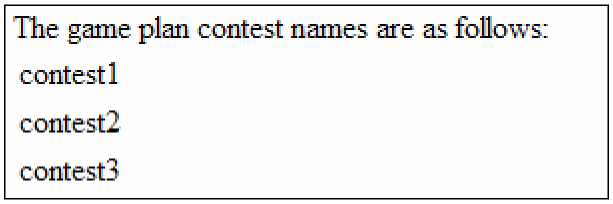


GamePlanRetrievalAction

projectId must be > 0

Project Id (must be positive) 0

6. Now put in 1 for the project ID and press the button again, and you will get a list of the contest names for the game plan. After the form is submitted, the List<ContestPlan> data is put on the value stack and can be obtained by the JSP.



The game plan contest names are as follows:

contest1

contest2

contest3

The following JSP was used in the demo, and it utilizes the GamePlanRetrievalAction.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<%@taglib uri="/struts-tags" prefix="s"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Game Plan Retrieval Action</title>

<link href="<s:url value="/css/main.css"/>" rel="stylesheet"
type="text/css"/>

</head>
<body>

<!-- display any action errors -->
```

```
<div style="color: #ff0000;"><s:iterator value="actionErrors">
    <span><s:property escape="false" /></span>
</s:iterator></div>

<h3>GamePlanRetrievalAction</h3>
<s:form action="gamePlanRetrievalAction">
    <table>
        <tr>
            <td>Project Id (must be positive)</td>
            <td><s:textfield name="projectId"/></td>
        </tr>
        <tr>
            <td><s:submit value="Get Game Plan" /></td>
            <td></td>
        </tr>
    </table>
</s:form>

</body>
</html>
```

The usage of other actions is similar.

5. Future Enhancements

None.