[TopCoder]

User Community Management 1.0 Component Specification

1. Design

The TopCoder registration process to become a member of TopCoder is going to be redesigned. The main goal is to allow for a minimal set of required info that the user needs to enter to sign up (handle, email, and password).

The existing TC web registration process will ask a new user to enter a lot of data when he/she tries to register to TC web-site. The registration process takes a lot of time, so many users can simply break the registration process and leave non-registered.

Thus there is a need of an easy to use and user friendly web-application for supporting registration on the new users and management of registered user profiles at TC web-site.

The main goal of this application is to simplify registration of the new users on TC web-site by minimizing the count of mandatory data fields for new account registration, and to improve usability of user profile management for the registered users. Any additional profile information will be requested from the user by the system as it is needed. For example, if a user registers to compete in an assembly contest the site would prompt them for any required info that they have not yet entered.

This component is responsible for implementing user referral and card/badges Struts actions.

1.1 Design Patterns

Strategy pattern – actions defined in this component use injected DataAccessInt and AuditDAO implementation instances as strategies.

DTO pattern – ReferralData is a DTO that is used for passing data from Struts action to JSP page.

MVC pattern – Struts uses this pattern, and Struts actions defined in this component are part of it.

1.2 Industry Standards

IoC, EJB, JavaBeans, XML (for configuration)

1.3 Required Algorithms

1.3.1 Logging

All Struts actions defined in this component must perform logging of errors and debug information with use of Logging Wrapper component. Logging should be performed in execute() methods only. The logger instance should be obtained with use of getLog() method. If this method returns null, then logging is turned off.

Errors must be logged at ERROR level, and method entry/exit information at DEBUG level.

Specifically, logging will be performed as follows, if logging is turned on:

- Method entrance and exit will be logged with DEBUG level.
  - Entrance format: [Entering method {*className.methodName}*]
  - Exit format: [Exiting method {*className.methodName}]*. Only do this if there are no exceptions.
- Method request and response parameters will be logged with DEBUG level (note that Struts actions obtain input and return output parameters via action attributes – see attributes with <<input>> and <<output>> stereotypes in TCUML, thus these attributes must be logged instead of method parameters):
  - Format for request parameters: [Input parameters[{request_parameter_name_1}:{ request_parameter_value_1}, {request_parameter_name_2}:{ request_parameter_value_2}, etc.)]]
  - Format for the response parameters: [Output parameter {response_value}] . Only do this if there are no exceptions.
  - If a request or response parameter is complex, use its toString() method. If that is not implemented, then print its value using the same kind of name:value notation as above. If a child parameter is also complex, repeat this process recursively.

- All exceptions will be logged at ERROR level, and automatically log inner exceptions as well.
  - Format: Simply log the text of exception: [Error in method {*className.methodName}*: Details {*error details*}]

In general, the order of the logging in a method should be as follows:

1. Method entry
2. Log method entry
3. Log action input parameters
4. If error occurs, log it and skip to step 7
5. Log method exit
6. Log action output values
7. Method exit

1.3.2 Authorization

All the actions defined in this component are available for authorized users only (it's assumed that Flash client performs authorization when requesting TopCoder member card data). The authorization is out of the scope of this component since it's performed with use of AuthInterceptor defined in Basic Struts Actions 1.0 component.

So it's valid to assume that session always contains not anonymous BasicAuthentication object as an attribute.

1.3.3 Changes to query tool

Currently TopCoder member card displays algorithm, design and development ratings of the coder. But this component (specifically, BaseRatedUserCommunityManagementAction and RetrieveCardDataAction) provides a support for other 10 competition tracks (you can see them in any member profile).

To achieve this it's required to make some changes to the database that holds a query used for serving "card_profile_info" requests, i.e. "topcoder_dw" database.

SVN path for "card_profile_info" query that retrieves data that is next processed by RetrieveCardDataAction is below:

https://coder.topcoder.com/internal/database/scripts/trunk/topcoder_dw/query_tool/query_card_profile_info.txt

And below is the content of this file with changes required to be made (marked with red color for clarity):

```
SELECT c.handle
     , c.member_since
     , r.num_ratings as num_competitions
     , r.highest_rating
     , r.rating as algorithm_rating
     , hsr.rating as high_school_rating
     , mr.rating as marathon_rating
     , p.path || i.file_name as image_path
     , cr.rank
     , cr.percentile
     , round(r1.rating) AS design_rating
     , round(r2.rating) AS development_rating
     , round(r3.rating) AS conceptualization_rating
     , round(r4.rating) AS specification_rating
     , round(r5.rating) AS architecture_rating
     , round(r6.rating) AS assembly_rating
     , round(r7.rating) AS test_suites_rating
     , round(r8.rating) AS test_scenarios_rating
     , round(r9.rating) AS ui_prototype_rating
     , round(r10.rating) AS ria_build_rating
```

```
    , cal.date as last_match
    , (SELECT max(final_points)
         FROM room_result rr, round r, round_type_lu rt
        WHERE rr.coder_id = c.coder_id
          AND rr.rated_flag = 1
          and rt.round_type_id = r.round_type_id
          and r.round_id = rr.round_id
          and rt.algo_rating_type_id = 1
          AND rr.division_id = 1) as best_div1
    , (SELECT max(final_points)
         FROM room_result rr, round r, round_type_lu rt
        WHERE rr.coder_id = c.coder_id
          and rt.round_type_id = r.round_type_id
          and r.round_id = rr.round_id
          and rt.algo_rating_type_id = 1
          AND rr.rated_flag = 1
          AND rr.division_id = 2) as best_div2
 FROM coder c
    , OUTER rating r
    , OUTER algo_rating hsr
    , OUTER algo_rating mr
    , OUTER(coder_image_xref cix, image i, path p)
    , OUTER coder_rank cr
    , OUTER tcs_dw:user_rating r1
    , OUTER tcs_dw:user_rating r2
    , OUTER tcs_dw:user_rating r3
    , OUTER tcs_dw:user_rating r4
    , OUTER tcs_dw:user_rating r5
    , OUTER tcs_dw:user_rating r6
    , OUTER tcs_dw:user_rating r7
    , OUTER tcs_dw:user_rating r8
    , OUTER tcs_dw:user_rating r9
    , OUTER tcs_dw:user_rating r10
    , OUTER (round ro, calendar cal, round_type_lu rt)
WHERE c.coder_id = @cr@
  AND r.coder_id = c.coder_id
  AND hsr.coder_id = c.coder_id
  AND mr.coder_id = c.coder_id
  AND cix.coder_id = c.coder_id
  AND cix.display_flag = 1
  AND cr.coder_id = c.coder_id
  AND cr.coder_rank_type_id = 2
  AND cix.image_id = i.image_id
  AND i.image_type_id = 1
  AND i.path_id = p.path_id
  AND r1.user_id = c.coder_id
  AND r2.user_id = c.coder_id
  AND r3.user_id = c.coder_id
  AND r4.user_id = c.coder_id
  AND r5.user_id = c.coder_id
  AND r6.user_id = c.coder_id
  AND r7.user_id = c.coder_id
  AND r8.user_id = c.coder_id
  AND r9.user_id = c.coder_id
  AND r10.user_id = c.coder_id
  AND r1.phase_id = 112
  AND r2.phase_id = 113
```

```
AND r3.phase_id = 134
AND r4.phase_id = 117
AND r5.phase_id = 118
AND r6.phase_id = 125
AND r7.phase_id = 124
AND r8.phase_id = 137
AND r9.phase_id = 130
AND r10.phase_id = 135
AND r.last_rated_round_id = ro.round_id
AND ro.calendar_id = cal.calendar_id
and ro.round_type_id = rt.round_type_id
and rt.algo_rating_type_id = 1
AND cr.algo_rating_type_id = 1
AND hsr.algo_rating_type_id = 2
AND mr.algo_rating_type_id = 3
```

1.3.4 XML validation

RetrieveCardDataAction is the only Struts action defined in this component that accepts an input parameter – coderId. Struts XML validation mechanism is used to validate this request parameter. The following file can be used for this:

RetrieveCardDataAction-validation.xml

```
<!DOCTYPE validators PUBLIC
        "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
        "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<validators>
    <field name="coderId">
        <field-validator type="required">
            <message>The coderId must be specified</message>
        </field-validator>
        <field-validator type="int">
            <param name="min">1</param>
            <message>The coderId must be positive</message>
        </field-validator>
    </field>
</validators>
```

1.4 Component Class Overview

BaseDataAccessUserCommunityManagementAction [abstract]

This is a base class for all Struts actions defined in this component that need to access some data in persistence. It holds DataAccessInt instance injected by Spring.

BaseRatedUserCommunityManagementAction [abstract]

This is a base class for all Struts actions defined in this component that need to check whether the user is rated or not. It just defines a protected method isRated().

The injected DataAccessInt instance for actions that extend this class must represent "topcoder_dw" database.

BaseUserCommunityManagementAction [abstract]

This is a base class for all Struts actions defined in this component. It holds a logger instance and defines methods for retrieving basic authentication data of the active user.

CardHelper

This is a helper class that provides methods for checking whether the user's member card is unlocking and for locking/unlocking this card.

DownloadBadgesAction

This is a Struts action that is responsible for showing badge download info to the user. This action simply checks whether the user is rated and throws an exception if not. Actual response rendering is performed with use of JSP.

PreviewCardAction

This is a Struts action that is responsible for previewing TopCoder member card. This action checks whether the user is rated and whether the member card is unlocked. Actual response rendering is performed with use of JSP.

ReferralData

This class is a container for information about a single referred coder. It is a simple JavaBean (POJO) that provides getters and setters for all private attributes and performs no argument validation in the setters. This class is used to pass referral data from Struts action to JSP.

RetrieveCardDataAction

This is a Struts action that is responsible for retrieving data that is used for rendering TopCoder member card. This action retrieves member card data from persistence. This action generates response directly without using JSP.

The injected DataAccessInt instance for this action must represent "topcoder_dw" database.

ShowCardDescriptionAction

This is a Struts action that is responsible for showing description of TopCoder member cards. This action does nothing. Actual response rendering is performed with use of JSP.

ShowCardInstructionsAction

This is a Struts action that is responsible for showing member card instructions to the user. This action simply checks whether the member card is unlocked. Actual response rendering is performed with use of JSP.

UnlockCardAction

This is a Struts action that is responsible for unlocking TopCoder member card. The action should be configured to refer to JSP page with card instructions in case of success.

ViewReferralsAction

This is a Struts action that is responsible for showing a list of referrals to the user. This action retrieves referrals data from persistence. Actual response rendering is performed with use of JSP.

The injected DataAccessInt instance for this action must represent "informixoltp" database.

1.5 Component Exception Definitions

UserCommunityManagementInitializationException

This exception is thrown by BaseUserCommunityManagementAction and its subclasses when the class is not initialized properly with Spring dependency injection.

1.6 Thread Safety

This component is not thread safe. But it can be used in thread safe manner when all actions are used within request scope.

All actions defined in this component are mutable and not thread safe. But they will be used in thread safe manner by Struts. It's assumed that “request” scope will be set up for all Struts actions in Spring configuration, thus actions will be accessed from a single thread only.

All actions will be initialized by Spring via setter injection, and their configuration parameters won't change after this. The injected Log, DataAccessInt and AuditDAO instances are expected to be thread safe.

ReferralData is mutable and not thread safe entity.

CardHelper is immutable and thread safe helper class.

UnlockCardAction uses TransactionManager to perform all database changes in a single transaction. Other actions don't use transactions since they don't modify data in persistence.

2. Environment Requirements

2.1 Environment

Development language: Java 1.6
Compile Target: Java 1.6
QA Environment:

- Java 1.6.0_04 http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Apache 2.2.4 http://archive.apache.org/dist/httpd/
- jira 4.1.2 http://www.atlassian.com/software/jira/ArchiveDownloads.jspa
- wiki 2.7 http://confluence.atlassian.com/display/DOC/Confluence+2.7+Release+Notes
- Jive Professional 4.2.5 http://www.jivesoftware.com/
- Informix 11.5 http://www.ibm.com/developerworks/downloads/im/dsexp/
- JBoss-4.0.4.GA http://sourceforge.net/projects/jboss/files/JBoss/JBoss-4.0.4.GA/
- JavaScript (jQuery 1.4.4) http://api.jquery.com/category/version/1.4.4/
- Log4j 1.2.15: http://logging.apache.org/log4j/1.2/index.html
- Unit Testing JUnit 1.4 www.junit.org
- Spring 2.5.6 http://s3.amazonaws.com/dist.springframework.org/release/SPR/spring-framework-2.5.6-with-dependencies.zip
- Hibernate 3.6 http://www.hibernate.org/downloads.html

2.2 TopCoder Software Components

Base Exception 2.0 – defines BaseRuntimeException used by custom exception defined in this component.

Logging Wrapper 2.0 – is used for logging errors and debug information.

User Profile and Audit Back End 1.0 – defines AuditDAO used in this component.

NOTE: The default location for TopCoder Software component jars is../lib/tcs/COMPONENT_NAME/COMPONENT_VERSION relative to the component installation. Setting the tcs_libdir property in topcoder_global.properties will overwrite this default location.

2.3 Third Party Components

Struts 2.2.1.1 (http://struts.apache.org/2.2.1.1/index.html)

NOTE: The default location for 3rd party packages is ../lib relative to this component installation. Setting the ext_libdir property in topcoder_global.properties will overwrite this default location.

3. Installation and Configuration

3.1 Package Name

com.topcoder.web.reg.actions.miscellaneous

3.2 Configuration Parameters

3.2.1 Configuration of BaseUserCommunityManagementAction subclasses

It's assumed that subclasses of BaseUserCommunityManagementAction will be initialized by Spring using a setter dependency injection. The following properties must be injected:

Property	Description	Values
authenticationSessionKey	The key used for obtaining BasicAuthentication instance from the session attributes.	String. Not empty. Required.

log	The Logging Wrapper logger to be used for logging errors and debug information. If not specified, logging is not performed.	Log. Optional.

3.2.2 Configuration of BaseDataAccessUserCommunityManagementAction subclasses

It's assumed that subclasses of BaseDataAccessUserCommunityManagementAction will be initialized by Spring using a setter dependency injection. The following properties must be injected:

Property	Description	Values
dataAccess	The DataAccessInt instance to be used for accessing user data in persistence.	DataAccessInt. Required.

Note that all subclasses of BaseRatedUserCommunityManagementAction and RetrieveCardDataAction use "topcoder_dw" database. And ViewReferralsAction – "informixoltp" database.

Additionally see parameters mentioned in the section 3.2.1.

3.2.3 Configuration of UnlockCardAction

It's assumed that UnlockCardAction will be initialized by Spring using a setter dependency injection. The following properties must be injected:

Property	Description	Values
auditDAO	The audit DAO used for auditing member card unlock operation.	AuditDAO. Required.

Additionally see parameters mentioned in the sections 3.2.1 and 3.2.2.

3.3 Dependencies Configuration

Logging Wrapper should be configured to use Log4j library.
Please see docs of other dependency components to configure them properly.

4. Usage Notes

4.1 Required steps to test the component

- Extract the component distribution.
- Follow Dependencies Configuration.
- Execute 'ant test' within the directory that the distribution was extracted to.

4.2 Required steps to use the component

Please see the demo.

4.3 Demo

Steps to run the demo

1. Execute "ant deploy" to deploy the demo
2. Start jboss
3. Enter http://127.0.0.1:8080/usercommunity/ in the browser, you will see this page

My Id: 100
My Name: topcoderuser
(The username/id is just mocked, check MockBasicAuthentication.java for more details.)

View referrals
Download badges
retrive card data

4. Click “View referrals”, the referrals handle and rating is displayed.

referrals:
handle: ivern
rating: 456

5. Click “download badges”, a simple “ok” is displayed.
6. Click “retrieve card data”, you’ll get a xml file.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <memberStats>
    <handle>hello-c</handle>
    <photo>/i/m/hello-c_big.jpg</photo>
    <algorithmRating>910</algorithmRating>
    <algorithmRatingMax>8</algorithmRatingMax>
    <rank>12</rank>
    <percentile>1.20</percentile>
    <memberSince>01.01.1970</memberSince>
    <lastMatchDate>01.01.1970</lastMatchDate>
    <bestDiv1>1</bestDiv1>
    <bestDiv2>2</bestDiv2>
    <competitions>3</competitions>
    <highSchoolRating>4</highSchoolRating>
    <marathonRating>5</marathonRating>
    <designRating>6</designRating>
    <developmentRating>9</developmentRating>
    <conceptualizationRating>10</conceptualizationRating>
    <specificationRating>11</specificationRating>
    <architectureRating>12</architectureRating>
    <assemblyRating>13</assemblyRating>
    <testSuitesRating>14</testSuitesRating>
    <testScenariosRating>15</testScenariosRating>
    <uiPrototypeRating>16</uiPrototypeRating>
    <riaBuildRating>17</riaBuildRating>
  - <algorithmRatingDistribution>
      <bucket>test1</bucket>
      <bucket>test2</bucket>
      <bucket>test3</bucket>
    </algorithmRatingDistribution>
  </memberStats>
```

4.3.1 *Sample Spring beans configuration*

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans-2.5.xsd">

  <!-- put here configuration for all DAO beans -->

  <bean id="viewReferralsAction" scope="prototype"
        class="com.topcoder.web.reg.actions.miscellaneous.ViewReferralsAction">
    <property name="authenticationSessionKey" value="authenticationkey" />
    <property name="log" ref="actionsLog" />
    <property name="dataAccess" ref="dataAccess" />
  </bean>

  <bean id="downloadBadgesAction" scope="prototype"
        class="com.topcoder.web.reg.actions.miscellaneous.DownloadBadgesAction">
    <property name="authenticationSessionKey" value="authenticationkey" />
    <property name="log" ref="actionsLog" />
    <property name="dataAccess" ref="dataAccess" />
  </bean>

  <bean id="showCardInstructionsAction" scope="prototype"
```

```
        class="com.topcoder.web.reg.actions.miscellaneous.ShowCardInstructionsAction">
    <property name="authenticationSessionKey" value="authenticationkey" />
    <property name="log" ref="actionsLog" />
  </bean>

  <bean id="showCardDescriptionAction" scope="prototype"
        class="com.topcoder.web.reg.actions.miscellaneous.ShowCardDescriptionAction">
    <property name="authenticationSessionKey" value="authenticationkey" />
    <property name="log" ref="actionsLog" />
  </bean>

  <bean id="retrieveCardDataAction" scope="prototype"
        class="com.topcoder.web.reg.actions.miscellaneous.RetrieveCardDataAction">
    <property name="authenticationSessionKey" value="authenticationkey" />
    <property name="log" ref="actionsLog" />
    <property name="dataAccess" ref="dataAccess" />
  </bean>

  <bean id="previewCardAction" scope="prototype"
        class="com.topcoder.web.reg.actions.miscellaneous.PreviewCardAction">
    <property name="authenticationSessionKey" value="authenticationkey" />
    <property name="log" ref="actionsLog" />
    <property name="dataAccess" ref="dataAccess" />
  </bean>

  <bean id="unlockCardAction" scope="prototype"
        class="com.topcoder.web.reg.actions.miscellaneous.UnlockCardAction">
    <property name="authenticationSessionKey" value="authenticationkey" />
    <property name="log" ref="actionsLog" />
    <property name="dataAccess" ref="dataAccess" />
    <property name="auditDAO" ref="auditDAO" />
  </bean>

  <bean id="actionsLog" class="com.topcoder.util.log.LogManager" factory-method="getLog">
    <constructor-arg value="com.topcoder.web.reg.actions.miscellaneous"
type="java.lang.String"/>
  </bean>

  <!-- Create audit DAO -->
  <bean id="auditDAO" class="com.topcoder.web.reg.actions.miscellaneous.MockAuditDao"/>

  <bean id="dataAccess"
class="com.topcoder.web.reg.actions.miscellaneous.MockDataAccessInt"/>
</beans>
```

4.3.2 *Sample Struts actions mapping*

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.1//EN"
    "http://struts.apache.org/dtds/struts-2.1.dtd">
<struts>
    <package name="default" extends="struts-default">

        <!-- define global error pages here -->

        <!-- define interceptors here -->

        <action name="ViewReferralsAction" class="viewReferralsAction">
            <result name="success">viewReferrals.jsp</result>
        </action>

        <action name="DownloadBadgesAction" class="downloadBadgesAction">
            <result name="success">downloadBadges.jsp</result>
        </action>

        <action name="ShowCardInstructionsAction" class="showCardInstructionsAction">
            <result name="success">cardInstructions.jsp</result>
        </action>

        <action name="PreviewCardAction" class="previewCardAction">
            <result name="success">previewCard.jsp</result>
```

```
        </action>

        <action name="UnlockCardAction" class="unlockCardAction">
            <result name="success">cardInstructions.jsp</result>
        </action>

        <action name="RetrieveCardDataAction" class="retrieveCardDataAction">
            <result name="success" type="stream">
                <param name="contentType">text/xml</param>
                <param name="inputName">xmlStream</param>
                <param name="bufferSize">1024</param>
            </result>
        </action>

    </package>
</struts>
```

4.3.3 Usage

This component will be accessed by Struts only, thus the programmatic usage is not available.

4.3.3.1 ViewReferralsAction

The following JSP can be used to render ViewReferralsAction output:

```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="s" uri="/struts-tags"%>
<html>
  <head>
    <title>Referrals</title>
  </head>
  <body>
    <table>
      <s:iterator value="referrals">
        <tr>
          <td><s:property value="handle" /></td>
          <td><s:property value="rating" /></td>
        </tr>
      </s:iterator>
    </table>
  </body>
</html>
```

Assuming that the user has two referrals, the final result can be the following:

```
<html>
  <head>
    <title>Referrals</title>
  </head>
  <body>
        referrals:
    <table>

        <tr>
          <td>handle: ivern</td>
        </tr>
        <tr>
          <td>rating: 456</td>
        </tr>

    </table>
  </body>
</html>
```

4.3.3.2 RetrieveCardDataAction

Let's take a co-pilot of this project for example – hello-c (coder ID = 22691760).

RetrieveCardDataAction should return the following XML feed for this user (red color is used to show the difference between expected output of this component and output generated by the existing code; real statistics data for 2011-05-12 are used):

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
 <memberStats>
<handle>hello-c</handle>
<photo>/i/m/hello-c_big.jpg</photo>
<algorithmRating>910</algorithmRating>
<algorithmRatingMax>8</algorithmRatingMax>
<rank>12</rank>
<percentile>1.20</percentile>
<memberSince>01.01.1970</memberSince>
<lastMatchDate>01.01.1970</lastMatchDate>
<bestDiv1>1</bestDiv1>
<bestDiv2>2</bestDiv2>
<competitions>3</competitions>
<highSchoolRating>4</highSchoolRating>
<marathonRating>5</marathonRating>
<designRating>6</designRating>
<developmentRating>9</developmentRating>
<conceptualizationRating>10</conceptualizationRating>
<specificationRating>11</specificationRating>
<architectureRating>12</architectureRating>
<assemblyRating>13</assemblyRating>
<testSuitesRating>14</testSuitesRating>
<testScenariosRating>15</testScenariosRating>
<uiPrototypeRating>16</uiPrototypeRating>
<riaBuildRating>17</riaBuildRating>
 <algorithmRatingDistribution>
<bucket>test1</bucket>
<bucket>test2</bucket>
<bucket>test3</bucket>
 </algorithmRatingDistribution>
 </memberStats>
```

Note that the current member card of hello-c is available here:

http://www.topcoder.com/i/card/tc_card.swf?postButton=true&memberID=22691760

And up-to-date statistics of this user can be found here:

http://www.topcoder.com/tc?module=MemberProfile&cr=22691760

4.3.3.3 Other actions

Other actions are quite trivial since they just check whether the user is rated and/or whether the member card is unlocked. Thus they are omitted here.

5. Future Enhancements

RetrieveCardDataAction can be enhanced to provide other data to be rendered in the TopCoder member card.