[TOPCODER]

Open Registration Validator Component v1.0 Specification

1. Design

The Generic Registration Framework is a framework that will be used to perform registrations for any type of TopCoder competition, including Architecture, Design, Development, Assembly, Testing, Bug Races, as well as future types of competitions.

This component defines a registration validator that ensures that the window for registration for a given role in a contest is open at the time of the registration request.

1.0.1 Quick Overview of definitions and concepts

This class implements RegistrationValidator to perform the necessary validation as defined by the Generic Registration Framework 1.0 validation contracts.

1.0.1.1 Resource Bundling the validation messages

This component provides an additional feature which is the ability to define the validation messages (both success and failure) in a resource bundle. This offers the following advantages:

- Users can easily internationalize the messages
- Users can easily change ALL messages with no need to recompile any code.

In addition to that, the actual utilization of the resource bundling here is completely transparent since users of the validator contract interface need NOT be aware of the resource bundling as everything is handled through the constructor and even comes with a default resource bundle name. The bundling is also a non-stopper in case it fails, as the validator will simply ignore it and provide the default messages.

1.1 Design Patterns

Strategy Pattern: The validator is expected to be used as a strategy in the Generic Registration Framework which is why resource bundling initialization is done through construction only, to preserve the already existing plug-in contract.

1.2 Industry Standards

None.

1.3 Required Algorithms

There are no complicated algorithms here. Please consult the UML method documentation for specific issues and details.

1.4 Component Class Overview

OpenRegistrationValidator:
This validator validates both registration and unregistration to ensure that the window for registration for the given role is still open. This validator makes sure to use both the Contest.getRegistrationStart() and getRegistrationEnd() dates, applying any offsets from the contest role.
When comparing the dates, it is important to include the offsets of the role registration times. For instance, a reviewer position may open 24 hours after the contest's registration starts, so the start time is actually contest.registrationStart + role.registrationStartOffset. The same applies to the registration end offset.
The date comparison is inclusive of the current time, to ensure registration is open and available at the exact moment specified in the contest and role information.
If there isn't sufficient data to determine if registration is allowed or not, due to missing or null date or offset values, the InsufficientDataException from the Registration Framework component will be thrown.
Note that this validator is wired to utilize a resource bundle when possible to fetch the specific validation messages.
We do not have to worry about locale which was not provided. By default, we will assume that it would be the current locale. Note that it is automatically utilized by the Resource Bundle Version 1.0 component.
Utilization of the bundle is optional and if the bundle is not found/present then the validator will simply fall back on the default/hard-coded messages, no exception will be thrown.

1.5 Component Exception Definitions

1.5.1 Custom Exceptions

No custom exceptions have been defined. We do utilize an exception from the *Registration Framework 1.0*:

InsufficientDataException:
This exception will be thrown, if there isn't sufficient data to determine if registration is allowed or not, due to missing or null date or offset values.

1.5.2 System and general java exceptions

IllegalArgumentException
Thrown when illegal argument is passed. This will include empty strings (i.e. strings that when trimmed have length 0) as well as when null argument is passed.

1.6 Thread Safety

As per contract set out in the Registration Framework 1.0 component, validators are expected to be thread-safe and this component follows that. The current validator is thread-safe as it is immutable.

2. Environment Requirements

2.1 Environment

- Development language: Java1.5

- Compile target: Java1.5, 1.6

2.2 TopCoder Software Components

- **Registration Framework 1.0:** This is the main component for which we are building the validator plug-in. It is there that the validator contract is defined.
- **Resource Bundle Version 1.0:** This is the component that provides a simplified access to resource bundling.

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

None.

3. Installation and Configuration

3.1 Package Name

com.topcoder.registration.validators

3.2 Configuration Parameters

3.2.1 Resource Bundle configuration

Parameter Name	Description	Value(s)
validation.registration.open.registration.success	This is the message id that the resource bundle should store the specific success message for successful registration validation. Please note that the current implementation will always assume current locale. (***Optional***)	Each value must be a valid localized string which holds the specific message. The following message should be configured: Registration is validated by OpenRegistrationValidator.
validation.registration. open.registration.failure.late	This is the message id that the resource bundle should store the specific failure message for an unsuccessful registration validation. Please note that the current implementation will always assume current locale. (***Optional***)	Each value must be a valid localized string which holds the specific message. The following message should be configured: Registration validation failed because registration is closed for the contest and role submitted.
validation.registration. open.registration.failure.early	This is the message id that the resource bundle should store the specific failure message for an unsuccessful registration validation. Please note that the current implementation will always assume current locale. (***Optional***)	Each value must be a valid localized string which holds the specific message. The following message should be configured: Registration validation failed because registration is not open yet for the contest and role submitted.

validation.unregistration.open.registration.success	This is the message id that the resource bundle should store the specific success message for successful registration validation. Please note that the current implementation will always assume current locale. (***Optional***)	Each value must be a valid localized string which holds the specific message. The following message should be configured: Unregistration is validated by OpenRegistrationValidator.
validation.unregistration. open.registration.failure.late	This is the message id that the resource bundle should store the specific failure message for an unsuccessful registration validation. Please note that the current implementation will always assume current locale. (***Optional***)	Each value must be a valid localized string which holds the specific message. The following message should be configured: Unregistration validation failed because registration is closed for the contest and role submitted.
validation.unregistration. open.registration.failure.early	This is the message id that the resource bundle should store the specific failure message for an unsuccessful registration validation. Please note that the current implementation will always assume current locale. (***Optional***)	Each value must be a valid localized string which holds the specific message. The following message should be configured: Unregistration validation failed because registration is not open yet for the contest and role submitted.

3.3 Dependencies Configuration

None at this point.

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

No specific steps need to be taken.

4.3 Demo

3.3.1 Setting up the resource bundle

All that we need to do here is create a simple properties file to be placed in a directory that is in the class path of the validator.
A properties file simply stores information about the messages for the validator. A properties file is in plain-text format. You can create the file with just about any text editor. The default properties file, which is called RegistrationFrameworkBundle.properties, contains the following lines:

```
# open registration validation failure for registration because the registration
# is done too late
validation.registration.open.registration.failure.late = Registration validation
failed because registration is closed for the contest and role submitted.

# open registration validation failure for registration because the registration
# is done too early
validation.registration.open.registration.failure.early = Registration validation
failed because registration is not open yet for the contest and role submitted.

# open registration validation success for registration
validation.registration.limit.success = Registration is validated by
OpenRegistrationValidator.
```

```
# open unregistration validation failure for unregistration because the
registration
# is done too late
validation.unregistration.open.registration.failure.late = Unregistration
validation failed because registration is closed for the contest and role
submitted.

# open unregistration validation failure for unregistration because the
# registration is done too early
validation.unregistration.open.registration.failure.early = Unregistration
validation failed because registration is not open yet for the contest and role
submitted.

# open unregistration validation success for unregistration
validation.unregistration.open.registration.success = Unregistration is validated
by OpenRegistrationValidator.
```

3.3.2 Creating a validator with or without a specific bundle

```
// We can create a default validator
RegistrationValidator validator = new OpenRegistrationValidator();

// or we can create a specific validator based on some specific bundle
RegistrationValidator validator = new OpenRegistrationValidator(bundleName);
```

3.3.3 Showing the validator in action

Here we will assume that the following input exists:

We have an open registration from:
May 28th 2008 at 9:00 am until May 31st 2007 at 9:00 am.
We will also assume an offset of 1 hour each way.

```
// Here is how we would run the code assuming some "reviewer" role has been
// passed and assuming that the current time is May 29th 2008 10:00 PM
// Note that the user is not relevant to this validation.
ValidationResult result = validator.validateRegistration(testContest
                              , user
                              , reviewerRole);
. . . this will be successful since registration is currently open. We would
. . . expect to get the following message in result.getDescription():
. . . "Registration is validated by OpenRegistrationValidator."

// Here is how we would run the code assuming some "reviewer" role has been
// passed and assuming that the current time is June 1st 2008 10:00 PM
// Note that the user is not relevant to this validation.
result = validator.validateRegistration(testContest
                              , user
                              , reviewerRole);
. . . this will fail since registration is currently closed. We would
. . . expect to get the following message in result.getDescription():
. . . "Registration validation failed because registration is closed for the
. . .  contest and role submitted."

// Here is how we would run the code assuming some "reviewer" role has been
// passed and assuming that the current time is May 25th 10:00 PM
// Note that the user is not relevant to this validation.
result = validator.validateRegistration(testContest
                              , user
                              , reviewerRole);
. . . this will fail since registration is currently not open. We would
. . . expect to get the following message in result.getDescription():
. . . "Registration validation failed because registration is not open
. . .   yet for the contest and role submitted.

// Here is how we would run the code assuming some "reviewer" role has been
// passed and assuming that the current time is May 29th 2008 10:00 PM
// Note that the user is not relevant to this validation.
```

```
ValidationResult result = validator.validateUnregistration(testContest
                              , user
                              , reviewerRole);
. . . this will be successful since unregistration is currently open. We would
. . . expect to get the following message in result.getDescription():
. . . "Unregistration is validated by OpenRegistrationValidator."

Please note that the rest of unregistration would follow the exact same pattern as
registration.
```

5. Future Enhancements

None at this point.