Jive Forum Services 1.1 Component Specification

1. Design

TopCoder uses Jive Forums for its site, which provides an API for programmatically managing them.

This component will provide some higher level services that will help setting up the structure of forums, groups and permissions used in TopCoder Software forums.

TopCoder will provide Jive Forums API to developers that sign up for developing the component.

Users can either use the JiveForumManager class to manage Jive Forum directly, or they can use the remote and local interface of the JiveForumServiceBean class to manage Jive Forum using EJB 3.0 technology.

New content is in **red**, modified content is in **blue**, old content is in **black**.

Version 1.1 changes:

a mock JiveForumSerivce implementation MockJiveForumService is defined, it will keep an in-memory storage of the watches, user roles, and categories created. This will be used for testing the services without having to set up a full Jive installation. For instance, when a user calls "setUserRole", they should then be able to call "getUserRole" to retrieve the role just added. No interactions with the Jive API are necessary.

1.1 Design Patterns

Strategy Pattern – The client can use the JiveForumService interface and its implementations as a strategy.

1.2 Industry Standards

Jive Forums 4.2.5 API

EJB 3.0

1.3 Required Algorithms

1.3.1 Publication of Services

Since the access to the forums is done through an administrator account, those services must not be made public without securing them first.

1.3.2 Create Category for Software Forums

Refer to the implementation note of the JiveForumManager.createCategory method for necessary details about this requirement.

```
1. Create a category and set properties for Software Forums :
ForumCategory newCategory =
forumFactory.getForumCategory(categoryConfiguration.getRootCategoryId())
         .createCategory(categoryConfiguration.getName(),
                         categoryConfiguration.getDescription());
newCategory.setProperty(PROPERTY_ARCHIVAL_STATUS,
         PROPERTY_ARCHIVAL_STATUS_ACTIVE);
newCategory.setProperty(PROPERTY_COMPONENT_ID,
         categoryConfiguration.getComponentId() + "");
newCategory.setProperty(PROPERTY_COMPONENT_VERSION_ID,
         categoryConfiguration.getVersionId() + "");
newCategory.setProperty(PROPERTY_COMPONENT_VERSION_TEXT,
         categoryConfiguration.getVersionText());
newCategory.setProperty(PROPERTY_MODIFY_FORUMS, "true");
```

```
2. Create groups and permissions for Software Forums :
GroupManager groupManager = forumFactory.getGroupManager();
Group swAdminGroup = groupManager.getGroup(GROUP_SOFTWARE_ADMINS);
Group moderatorGroup =
       groupManager.createGroup(GROUP_SOFTWARE_MODERATORS_PREFIX +
       newCategory.getID());
Group userGroup = groupManager.createGroup(GROUP_SOFTWARE_USERS_PREFIX +
                                           newCategory.getID());
moderatorGroup.setDescription(newCategory.getName());
userGroup.setDescription(newCategory.getName());

// set permissions.
PermissionsManager categoryPermissionsManager =
                        newCategory.getPermissionsManager();
for (int i = 0; i < MODERATOR_PERMS.length; i++) {
       categoryPermissionsManager.addGroupPermission(moderatorGroup,
           PermissionType.ADDITIVE, MODERATOR_PERMS[i]);
}
for (int i = 0; i < REGISTERED_PERMS.length; i++) {
       categoryPermissionsManager.addGroupPermission(userGroup,
            PermissionType.ADDITIVE, REGISTERED_PERMS[i]);
}
for (int i = 0; i < ADMIN_PERMS.length; i++) {
       categoryPermissionsManager.addGroupPermission(swAdminGroup,
           PermissionType.ADDITIVE, ADMIN_PERMS[i]);
}
if (!category.isPublic()) {
      for (int i = 0; i < SW_BLOCK_PERMS.length; i++) {
           categoryPermissionsManager.addAnonymousUserPermission(
                PermissionType.NEGATIVE, SW_BLOCK_PERMS[i]);
           categoryPermissionsManager.addRegisteredUserPermission(
                PermissionType.NEGATIVE, SW_BLOCK_PERMS[i]);
      }
}

3. If a Moderator User Id is specified, user id must be assigned to the
moderators group for the category, and a watch for the user must be
created for the category.

if (categoryConfiguration.getModeratorUserId() > 0) {
      Call setUserRole(categoryConfiguration.getModeratorUserId(),
          newCategory.getID(), UserRole.MODERATOR)
          to add user to the moderators group.
      Call watch(categoryConfiguration.getModeratorUserId(),
          newCategory.getID(), EntityType.ForumCategory)
          to create the watch for the user to the created category.
}

4. If a template category is specified. Copy the contents of the template
category into the newly created category.
if (categoryConfiguration.getTemplateCategoryType() != null) {
    // template category type is specified; get the configured template
category id.
    templateCatId = categoryTemplateIds.get(
                        categoryConfiguration.getTemplateCategoryType());
} else {
    templateCatId = categoryConfiguration.getTemplateCategoryId();
}

if (templateCatId > 0) {
    // template category id is specified. Copy the contents.
   Call templateCategory = forumFactory.getForumCategory(templateCatId);
```

```
   Then all the forums, threads and announcements must be copied from the
templateCategory to the newly created category, maintaining the structure.
   The copy operation is intended to be a deep clone.
}

finally return the newCategory.getID().
```

1.4 Component Class Overview

1.4.1 Package com.topcoder.forum.service

- **JiveForumManager**: This class provides methods to manage the Jive Forum. It provides functionalities to set up the structure of forums, groups and permissions used in TopCoder Software forums.
- **TCAuthToken**: It represents the authorization token used to plug into the Jive Forum APIs.
- **UserRole**: Enum class for the roles to be assigned to the user.
- **EntityType**: Enum class for entity types.
- **CategoryType**: Enum class for category types.
- **CategoryConfiguration**: This class contains configuration properties to be used in JiveForumManager to create the category.

1.4.2 Package com.topcoder.forum.service.ejb

- **JiveForumService**: This interface defines contract to manage the Jive Forum.
- **JiveForumServiceLocal**: This EJB Local interface extends the JiveForumService interface.
- **JiveForumServiceRemote**: This EJB Remote interface extends the JiveForumService interface.

1.4.3 Package com.topcoder.forum.service.ejb.bean

- **JiveForumServiceBean**: This stateless bean class implements the JiveForumServiceLocal and JiveForumServiceRemote interfaces.

1.4.4 Package com.topcoder.forum.service.ejb.mock

- **MockJiveForumService**: This is a mock JiveForumSerivce implementation that will keep an in-memory storage of the watches, user roles, and categories created. This will be used for testing the services without having to set up a full Jive installation. For instance, when a user calls "setUserRole", they should then be able to call"getUserRole" to retrieve the role just added. No interactions with the Jive API are necessary.

1.5 Component Exception Definitions

1.5.1 System Exceptions

- **IllegalArgumentException**: It is thrown when the passed-in argument is illegal. NOTE: A string is empty if its length is 0 after being trimmed.

1.5.2 Custom Exceptions

- **JiveForumManagementExcetion**: This exception extends the BaseCriticalException. It's the general exception for the whole component, and all the other custom exceptions should extend from it.
- **ForumEntityNotFoundException**: This exception extends the JiveForumManagementException. It is thrown when the forum entity (category, forum, thread) is not found in the Jive Forum.

- **UserNotFoundException**: This exception extends the JiveForumManagementException. It is thrown when the user is not found in the Jive Forum.
- **ServiceConfigurationException**: This exception extends the BaseRuntimeException. It is thrown from the JiveForumServiceBean if the bean is not configured properly to be initialized.

1.6 Thread Safety

The CategoryConfiguration class is mutable, but this class is not supposed to be shared in multiple threads. So it won't affect the thread-safety of this design. The TCAuthToken class is immutable and thread-safe. The enum classes are thread-safe.

The JiveForumManager class is immutable, and the Jive Forum APIs are thread-safe, so this class is thread-safe.

The JiveForumServiceBean is a stateless session bean. It must be thread-safe as it will be accessed from multiple threads by multiple users when deployed in the EJB container. The variables in JiveForumServiceBean are set only once in the initialize method immediately after it is created, and their values will never be changed afterwards. So it is thread-safe too.
The new class MockJiveForumService is thread safe because EJB3 container will properly handle thread safety issues.

2. Environment Requirements

2.1 Environment

- Java 5.0+
- Jive Forums 4.2.5
- Informix 10
- JBoss 4.2.1 GA with EJB 3.0

2.2 TopCoder Software Components

Base Exception 2.0 This component's custom exceptions extend from it.

2.3 Third Party Components

Jive Forums 4.2.5:
http://www.jivesoftware.com/builds/docs/prof/4.2.5/documentation/javadoc/api/index.html

3. Installation and Configuration

3.1 Package Name

com.topcoder.forum.service

com.topcoder.forum.service.ejb

com.topcoder.forum.service.ejb.bean

com.topcoder.forum.service.ejb.mock

3.2 Configuration Parameters

Configuration for JiveForumServiceBean (in "env-entry" elements of the deployment descriptor)

Parameter Name	Parameter Description	Parameter Value
AdminUserId	Represents the user id used to	Type: java.lang.lang.

	manage the Jive Forum. Required.	The value must be parsable to long value. And the long value must be > 0.
ApplicationCategoryId	Represents the template category id for the application software forum. Required	Type: java.lang.lang. The value must be parsable to long value. And the long value must be > 0.
ComponentCategoryId	Represents the template category id for the component software forum. Required	Type: java.lang.lang. The value must be parsable to long value. And the long value must be > 0.
AssemblyCompetitionCategoryId	Represents the template category id for the assembly competition software forum. Required	Type: java.lang.lang. The value must be parsable to long value. And the long value must be > 0.
TestingCompetitionCategoryId	Represents the template category id for the testing competition software forum. Required	Type: java.lang.lang. The value must be parsable to long value. And the long value must be > 0.

3.3 Dependencies Configuration

None.

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

Deploy this component in an EJB 3.0 container, and then follow the demo to call the beans. The beans should be configured in ejb-jar.xml file like this:

```
<enterprise-beans>
  <session>
  <ejb-name>JiveForumService</ejb-name>
  <remote>com.topcoder.forum.service.ejb.JiveForumServiceRemote</remote>
  <local>com.topcoder.forum.service.ejb.JiveForumServiceLocal</local>
  <ejb-class>
       com.topcoder.forum.service.ejb.bean.JiveForumServiceBean
  </ejb-class>
  <session-type>Stateless</session-type>
  <transaction-type>Container</transaction-type>
  <env-entry>
       <env-entry-name>AdminUserId</env-entry-name>
       <env-entry-type>java.lang.String</env-entry-type>
       <env-entry-value>1</env-entry-value>
  </env-entry>
  <env-entry>
       <env-entry-name>ApplicationCategoryId</env-entry-name>
       <env-entry-type>java.lang.String</env-entry-type>
       <env-entry-value>1</env-entry-value>
  </env-entry>
  <env-entry>
       <env-entry-name>ComponentCategoryId</env-entry-name>
```

```
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>1</env-entry-value>
    </env-entry>
    <env-entry>
        <env-entry-name>AssemblyCompetitionCategoryId</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>1</env-entry-value>
     </env-entry>
     <env-entry>
        <env-entry-name>TestingCompetitionCategoryId</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>1</env-entry-value>
      </env-entry>
    </session>
</enterprise-beans>

The configuration file for the MockJiveForumService can be like this:
<enterprise-beans>
  <session>
  <ejb-name>JiveForumService</ejb-name>
  <remote>com.topcoder.forum.service.ejb.JiveForumServiceRemote</remote>
  <local>com.topcoder.forum.service.ejb.JiveForumServiceLocal</local>
  <ejb-class>
        com.topcoder.forum.service.ejb.mock.MockJiveForumService
  </ejb-class>
  <session-type>Stateless</session-type>
  <transaction-type>Container</transaction-type>
  </session>
</enterprise-beans>


And we can configure it in the jboss-service.xml like this:
<jboss>
    <enterprise-beans>
        <session>
            <ejb-name>JiveForumService</ejb-name>
            <jndi-name>JiveForumService</jndi-name>
        </session>
    </enterprise-beans>
</jboss>
```

4.3 Demo

```
// It needs jndi.properties in classpath
InitialContext ctx = new InitialContext();
JiveForumServiceRemote bean = (JiveForumServiceRemote) ctx.lookup(
                "JiveForumService/remote");

long userId = 1;
long entityId = 1;

// The user (identified by userId) watches the forum (identified by
//entityId)
bean.watch(userId, entityId, EntityType.FORUM);

// check whether the user is watching the forum or not. True should be
//returned
boolean watched = bean.isWatched(userId, entityId, EntityType.FORUM);
System.out.println("Watched: " + watched);
```

```
long categoryId = 1;

// set moderator role to the user (identified by userId) for the
// category (identified by categoryId)
bean.setUserRole(userId, categoryId, UserRole.MODERATOR);

// get user role. UserRole.MODERATOR is expected.
UserRole role = bean.getUserRole(userId, categoryId);
System.out.println("The role: " + role);

// create CategoryConfiguration with templateCategoryId set
CategoryConfiguration categoryConfig = new CategoryConfiguration();
categoryConfig.setName("Test");
categoryConfig.setDescription("Test Desc");
categoryConfig.setRootCategoryId(10);
categoryConfig.setComponentId(1);
categoryConfig.setVersionText("v1.0");
categoryConfig.setVersionId(1);
categoryConfig.setTemplateCategoryId(1);

// create category. Contents from template category will be copied into
//it
long newCatId = bean.createCategory(categoryConfig);

// change CategoryConfiguration to use the templateCategoryType
categoryConfig.setTemplateCategoryId(-1);
categoryConfig.setTemplateCategoryType(CategoryType.COMPONENT);

// create category. Contents from the component template category will
// be copied into it.
newCatId = bean.createCategory(categoryConfig);

// Use JiveForumManager to operate the service
Map<CategoryType, Long> categoryTemplateIds = new HashMap<CategoryType,
   Long>();
categoryTemplateIds.put(CategoryType.APPLICATION, new Long(1));
categoryTemplateIds.put(CategoryType.COMPONENT, new Long(2));

long adminUserId = 1;
JiveForumManager manager = new JiveForumManager(adminUserId,
          categoryTemplateIds);

//Create the category
categoryConfig.setTemplateCategoryType(CategoryType.APPLICATION);
newCatId = manager.createCategory(categoryConfig);
```

Demo for the MockJiveForumService:

If MockJiveForumService is deployed to EJB3 container, it can be used same as the above demo, so no duplication here, while the following code shows its programmatic usage.

```
// create mock service
MockJiveForumService service = new MockJiveForumService();
// add a watch, note that the watch is just stored in-memory
long userId = 123;
long entityId = 456;
service.watch(userId, entityId, EntityType.FORUM);

// test if user is watch a entity
// true should be returned
boolean watching = service.isWatched(userId, entityId, EntityType.FORUM);
// false should be returned
boolean watching = service.isWatched(userId, entityId,
EntityType.FORUM_THREAD);
// false should be returned
boolean watching = service.isWatched(userId, 789, EntityType.FORUM);

// set user role
long categoryId = 3;
service.setUserRole(userId, categoryId, UserRole.NO_ACCESS);
// get user role
// UserRole.NO_ACCESS should be returned
UserRole role = service.getUserRole(userId, categoryId);

// create category
CategoryConfiguration categoryConfig = new CategoryConfiguration();
categoryConfig.setName("Test");
categoryConfig.setDescription("Test Desc");
categoryConfig.setRootCategoryId(10);
categoryConfig.setComponentId(1);
categoryConfig.setVersionText("v1.0");
categoryConfig.setVersionId(1);
categoryConfig.setTemplateCategoryId(1);

long newCatId = service.createCategory(categoryConfig);

// get all category configurations
// the returned collection should contain one category configuration,
// and it is just the categoryConfig
Collection<CategoryConfiguration> configs =
service.getAllCategoryConfigurations();

// get category configuration by category id
// the returned config is just the categoryConfig
CategoryConfiguration config =
service.getCategoryConfigurationById(newCatId);
```

5 Future Enhancements

None.