Prerequisite Document Manager 1.0 Component Specification

1. Design

This component provides CRUD operations on document, document name; CRUD operations on member answer; CRUD operations on competition document (document which is specific for concrete competition and role). Component runs as stateless EJB. This component is used Hibernate JPA implementation to work with persistence. It is used by TopCoder Prerequisite Service and can be used for the other services.

This component is used as the persistence layer for prerequisite service. This design uses DTO and DAO to perform the persistence actions. Five data classes follow POJO, all the methods are getters and setters. They can be managed by JPA. DocumentManager interface uses DAO pattern. An EJB3 Stateless Session Bean implements this interface using JPA and Hibernate technologies.

Authorization is required as defined in the spec. This design uses programmatic way to do authorization check instead of security annotations to achieve flexible authorization as requirement on authorization is always changing.

This design provides more configuration properties than the requirements to make this component more flexible. The configuration is done by ENC (Environment Naming Context) since this component uses EJB3 beans. For detail information, please refer to the configuration parameters part in this document.

1.1 Design Patterns

DAO, DTO. CompetitionDocument, Document, MemberAnswer, DocumentVersion, DocumentName are DTOs while DocumentManager and DocumentManagerBean are DAOs.

1.2 Industry Standards

EJB3 Stateless Session Bean, JPA, Hibernate, Java Bean

1.3 Required Algorithms

1.3.1 Logging

Only DocumentManagerBean class in this component will use the logging mechanism.

All methods in the class should log in the following manner:

- Method entrance and exit at INFO level: the log message should contain the name of method and the parameters
- All business logic events (for example authenticating or interacting with backend persistence layer) at INFO level: the log message should contain a description of event and result
- All exceptions and errors at ERROR level. This includes illegal arguments: the log message should include the message of exception and the type of exception
- Any additional logging is at the developer’s discretion.

Log format: plus all informations. The developer is free to add other informations into log message.

1.3.2 *Authorizing user request*

In each method in the DocumentManagerBean, the caller should be validated.
In the configuration properties, there is a property named “allowedRoleNames” which is used to identity what roles can call methods in the DocumentManagerBean.

Role of the caller can be retrieved from SessionContext interface since this component run as a EJB stateless session bean.

The algorithm is very simple; just iterate all the allowed roles to check if the caller is in one of them:

```
boolean isValid = false;
for (String roleName : allowedRoleNames) {
   if (sessionContext.isCallerInRole(roleName)) {
      isValid = true;
      break;
   }
}

if (!isValid) {
   throw AuthorizationException;
 }
```

1.3.3 *Database schema*

```
-- SQL file for Informix database

/* This is base table to save documnets. Each document contain id, create date and
description. */
CREATE TABLE document (
document_id INTEGER NOT NULL, -- auto generated primary key
create_date DATETIME YEAR TO FRACTION(3) NOT NULL,
description LVARCHAR(1000),
PRIMARY KEY(document_id)
);

/* This table used to save document version. Primary key is unique auto generated version_id.
Document id and version together is potential primary key.
Tables saves also date of version, id of document name for this version and its content*/
CREATE TABLE document_version (
document_version_id INTEGER NOT NULL, -- auto generated primary key
-- combination of document id and version is potential primary key
document_id INTEGER NOT NULL,
version INTEGER NOT NULL,
document_name_id INTEGER NOT NULL, -- foreign key to name
version_date DATETIME YEAR TO FRACTION(3) NOT NULL,
content TEXT NOT NULL, -- this type should be used for content of any size
PRIMARY KEY(document_version_id) FOREIGN KEY(document_id), -- foreign key to document
table
REFERENCES document(document_id) FOREIGN KEY(document_name_id), -- foreign key to
document_name table
REFERENCES document_name(document_name_id)
);

/* This table is used to save name of document. Ususally document name will not be changed
from version to version, but it such case is possible.
*/
CREATE TABLE document_name (
document_name_id INTEGER NOT NULL, -- auto generated primary key
name VARCHAR(254) NOT NULL, -- name of document
PRIMARY KEY(document_name_id)
);

/*This tables is used to relate document version, competition and role. It defines which
version of document should be signed by user in specific role.
*/
CREATE TABLE competition_document (
competition_document_id INTEGER NOT NULL, -- auto generated primary key
-- combination of 3 next fields is potential primary key
document_version_id INTEGER NOT NULL,
competition_id INTEGER NOT NULL,
role_id INTEGER NOT NULL,
PRIMARY KEY(competition_document_id),
FOREIGN KEY(document_version_id) REFERENCES document_version(document_version_id) --
foreign key to document_version table
);

/*This table is used to save member answer on specific document in specific competition
for specific role.
It saves competition document id, member id e.g. user id, answer which can be true or
false e.g.
document is accepted or rejected, date of answer.
*/
CREATE TABLE member_answer (
member_answer_id INTEGER NOT NULL,
--combination of competition_document_id and member_id is potential primary key as user
can answer for the same competition document only once
competition_document_id INTEGER NOT NULL,
member_id INTEGER NOT NULL,
answer BOOLEAN NOT NULL,
answer_date DATETIME YEAR TO FRACTION(3) NOT NULL,
PRIMARY KEY(member_answer_id),
FOREIGN KEY(competition_document_id) REFERENCES
competition_document(competition_document_id)-- foreign key to competition document
table
);
```

1.4 Component Class Overview

Package com.topcoder.service.prerequisite.document

Interface DocumentManager
This interface provides great number of utility operations with documents like CRUD operations on document; CRUD operations on member answer; CRUD operations on competition document (document which is specific for concrete competition and role).

This should be used by PreRequisiteService implementation and also it can be used to define more services.

Package com.topcoder.service.prerequisite.document.model

Class CompetitionDocument
This is a data class for competition document. It is an entity that can be managed by JPA. This entity is for competition_document table.

A competition document has the following attributes:
Competition document id, competition id, role id
A competition document also contains the following reference attributes:
Document version, a set of member answers.

Class Document
This is a data class for a document. It is an entity that can be managed by JPA. This entity is for document table.

A document has the following attributes:
Document id, create date, description
A document also contains the following reference attributes:
A set of document versions

Class DocumentName
This is a data class for document name. It is an entity that can be managed by JPA. This entity is for document_name table.

A document name has the following attributes:
Document name id, name
A document name also contains the following reference attributes:
A set of document versions

Class DocumentVersion
This is a data class for document version. It represents concrete document version.
It is an entity that can be managed by JPA. This entity is for document_version table.

A document version has the following attributes:
Document version id, document version, date of version, content
A document version also contains the following reference attributes:
Document, document name, a set of competition documents

Class MemberAnswer
It is an entity that can be managed by JPA. This entity is for member_answer table.

A member answer has the following attributes:
Member answer id, member id, accept or reject answer, date of answer
A member answer also contains the following reference attributes:
Competition document

Package com.topcoder.service.prerequisite.document.ejb

Class DocumentManagerBean
This is an EJB3 stateless session which implements the DocumentManager interface.
This implementation utilizes JPA to interact with the persistence layer.

In this implementation, the following properties are configurable:
loggerName, persistenceUnitName, allowedRoleNames.

Interface DocumentManagerLocal
This is the local interface for DocumentManagerBean.
When the bean is deployed, the local interface can be looked up via "DocumentManagerBean/local".

Interface DocumentManagerRemote
This is the remote interface for DocumentManagerBean.
When the bean is deployed, the remote interface can be looked up via "DocumentManagerBean/remote".

1.5 Component Exception Definitions

Component uses three standard exceptions: IllegalArgumentException, IllegalStateException.

IllegalArgumentException is thrown for invalid arguments, for example, argument is null, string argument is empty, etc.

IllegalStateException is thrown when session context is not injected in EJB3 stateless session bean.

.
This component defines following custom exceptions:

Class DocumentManagerException
This exception extends the BaseCriticalException. It is also the parent exception class for all the other custom exceptions in this design.

Class ConfigurationException
This exception is thrown if the configuration is bad: missing required properties or properties in wrong format.

Class AuthorizationException
This exception is thrown if the authorization check for a user request fails.

Class DocumentPersistenceException
This exception is thrown if some errors occur in the persistence layer.

Class CompetitionDocumentAnsweredException
This exception is thrown if the competition document has been answered by the user while the user wants to answer the competition document again.

1.6 Thread Safety

This component uses EJB3 stateless session bean and JPA implementation so the environment of this component is an EJB container. EJB container routes each request to a different instance of stateless session bean class which means stateless session bean is not required to be threading safety.

The five entity classes are not thread safety as they follow POJO. They are all mutable.

There is one stateless session bean in this design which is DocumentManagerBean. The init() can change the state of this EJB bean. But as this class is managed by EJB container, the init() will only be called once after the constructor of this bean. For this aspect, this bean can be counted as thread safety. The persistence layer is thread safe too because this bean uses container-managed transaction. Transaction attributes are REQUIRED for all methods (that has to be interacted with the persistence layer) in this bean class.

2. Environment Requirements

2.1 Environment

Development language: Java 5.0

Compile target: Java 5.0 and Java 6.0

QA Environment

- RedHat Linux 9
- Informix 10
- JBoss 4.2

2.2 TopCoder Software Components

- Logging Wrapper 2.0 – It is used to perform logging by DocumentManagerBean

- Base Exception 2.0 – It is used to support custom exceptions

2.3 Third Party Components

- EJB 3.0 – The EJB3 Stateless Session Bean is used in this component
- JPA 1.0 – This component follows java persistence interface when interacting with backend persistence layer
- Hibernate 3.2 and higher – Hibernate 3.2 implements JPA 1.0 standard and is used as the recommend JPA provider.

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.service.prerequisite.document
com.topcoder.service.prerequisite.document.model
com.topcoder.service.prerequisite.document.ejb

3.2 Configuration Parameters

The following configuration properties are supported:

DocumentManagerBean:

Property Name	Property Description
loggerName	This property describes the name of the logger. A valid value is a String, it is optional, when it is absent, then no logging will be performed.
persistenceUnitName	This property describes the name of the persistence unit. It is used to retrieve EntityManager from SessionContext. A valid value is a String, it is required.
allowedRoleNames	This property describes role names that are allowed to access methods in this bean. A valid value is a non-empty String, it is required. Each role name is concatenated with a comma.

Here is a sample ejb-jar.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar>
  <enterprise-beans>
    <session>
      <ejb-name>DocumentManagerBean</ejb-name>
      <ejb-class>
        com.topcoder.service.prerequisite.document.ejb.DocumentManagerBean
      </ejb-class>
      <env-entry>
        <env-entry-name>loggerName</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>document_manager</env-entry-value>
      </env-entry>
      <env-entry>
        <env-entry-name>persistenceUnitName</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>document_manager </env-entry-value>
      </env-entry>
      <env-entry>
        <env-entry-name>allowedRoleNames</env-entry-name>
        <env-entry-type>java.lang.String</env-entry-type>
        <env-entry-value>admin,manager,Administrator</env-entry-value>
      </env-entry>
    </session>
  </enterprise-beans>
</ejb-jar>
```

3.3 Dependencies Configuration

Refer to Logging Wrapper 2.0 component for its configuration.

Sample configuration files for JPA are provided under /docs directory.

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

This component should be packaged and deployed to an application server. Sample configuration files are provided under /docs directory.

4.3 Demo

This demonstration demonstrates the scenario how with use DocumentManager to finish assignment document management. Sample configuration is provided under /docs directory.

```
 // look up the DocumentManager via JNDI
InitialContext ctx = new InitialContext();
DocumentManager docMgr = (DocumentManager) ctx.lookup("DocumentManagerBean/remote");

 // add assign document

 Document doc = new Document();

 doc.setCreateDate(new Date());

 doc.setDescription("assignment document for user confirmation!");

 doc = documentManager.addDocument(doc);
```

```
        // add assign document name
        DocumentName docName = new DocumentName();
        docName.setName("Assignment Document");
        docName = documentManager.addDocumentName(docName);

        // add the first assignment document version
        DocumentVersion docVersion = new DocumentVersion();
        docVersion.setDocumentVersion(1);
        docVersion.setVersionDate(new Date());
        docVersion.setContent("assignment document");
        docVersion.setDocument(doc);
        doc.getDocumentVersions().add(docVersion);
        docVersion.setDocumentName(docName);
        docName.getDocumentVersions().add(docVersion);
        docVersion = documentManager.addDocumentVersion(docVersion);

        // add a competition document that requires assignment document
        CompetitionDocument competitionDoc = new CompetitionDocument();
        competitionDoc.setRoleId(new Long(1));
        competitionDoc.setCompetitionId(new Long(1));
        competitionDoc.setDocumentVersion(docVersion);
        docVersion.getCompetitionDocuments().add(competitionDoc);
        competitionDoc =
documentManager.addCompetitionDocument(competitionDoc);

        // do some updates to assignment document version as requirement changes
        DocumentVersion newVersion =
documentManager.getDocumentVersion(doc.getDocumentId(), null);
        newVersion.setContent("updated assignment document");
        documentManager.updateDocumentVersion(newVersion);

        // record user answer to the competition document
        MemberAnswer memberAnswer = new MemberAnswer();
        memberAnswer.setAnswer(true);
        memberAnswer.setCompetitionDocument(competitionDoc);
        memberAnswer.setMemberId(new Long(1));
        memberAnswer.setAnswerDate(new Date());
        memberAnswer = documentManager.addMemberAnswer(memberAnswer);
        // remove the user answer
documentManager.deleteMemberAnswer(memberAnswer.getMemberAnswerId());
```

5. Future Enhancements

- Add more methods to DocumentManager interface.