COO Generator 1.0 Component Specification

1. Design

This component is used to generate COO report on a given TopCoder component contest. The report would include the winning and second place designers/developers as well as all of the reviewers. The component dependencies are also included. The report can be in PDF format or XML format.

1.1 Design Patterns

Façade

It is used by DefaultCOOReportGenerator that is a façade of the other interfaces: ContestDataRetriever, ComponentDependencyExtractor and ComponentManager.

Strategy

The interfaces defined in com.topcoder.management.contest.coo are Strategies. Their implementations are also defined in this component. DefaultCOOReportGenerator acts as context for ContestDataRetriever, ComponentDependencyExtractor and ComponentManager implementations.

Template method

BaseCOOReportSerializer#serializeCOOReportToFile calls serializeCOOReportToByteArray as template method.

1.2 Industry Standards

XML, PDF, MS Excel, Open Office, PDF

1.3 Required Algorithms

1.3.1 Obtain Connection Through DB Connection Factory

The following is an example on obtaining the Connection object:

```
Connection connection =
dbConnectionFactory.createConnection(connectionName);
```

1.3.2 Reading from the Database

The following is an example of executing SELECT statement:

```
Statement select = connection.createStatement();
ResultSet result = select.executeQuery("SELECT blabla....");
while (result.next())
{
	// process results one row at a time
	// use result.getInt(columnNumber);
}
```

1.3.3 Update the database through transaction

The following is an example of executing INSERT statement inside a transaction. Note that all operations that modify the database must use transaction.

```
connection.setAutoCommit(false);
Statement statement = connection.createStatement();
statement.executeUpdate("INSERT INTO blabla...");
// do some other updates

connection.commit();
```

1.3.4 *Logging*

The logging is done through Logger Wrapper. The following needs to be log.
- Exception thrown as well as the stack trace at ERROR level
- List of dependencies extracted, list of components added/retrieved and contest data retrieved at INFO level
- Log method entry/exit at DEBUG level

Developers are free to improve on the logger.

1.3.5 *Table definition for third party libraries*

A new database table is to be created. The name of the table is third_party_library.
The table has the following columns and data types
- name. VARCHAR
- version. VARCHAR
- url. VARCHAR
- license. VARCHAR
- usage_comments. VARCHAR
- path. VARCHAR
- alias. VARCHAR
- notes. VARCHAR
- category. VARCHAR

This is used by DBComponentManager to export the given XLS file into database (into third_party_library table as above). It is also used to query the given 3rd party dependency by the given name and version.

1.3.6 *Serialize Report to PDF*

Open Office document is used as template for PDF report generation. The Open Office document would contain a set of placeholders that would be replaced by the actual values during report generation.

The PDF report generation is performed by Open Office API:
- Replace all the place holders
- Export to PDF

For pseudocode on the PDF report generation, see PDFCOOReportSerializer#serializeCOOReportToByteArray.

The set of place holders are as follow:
- All properties of ContestData (e.g. componentName, contestEndDate, etc)
- projected to indicate project id
- "dependency" to represent all dependencies
- "dependency-external" to represent third party dependencies

- “dependency-internal” to represent TC dependencies
- “dependency-compile-external”, “dependency-test-external”, “dependency-compile-internal”, “dependency-text-internal”. These, respectively, represent the dependencies based on the type and category.

1.3.7 *Serialize Report to XML*

An XML file is used as template for XML report generation. The XML file would contain a set of place holders. TopCoder Document Generator component is used to replace all the placeholders with the actual values during XML Report Generation.

For pseudocode on the XML report generation, see XMLCOOReportSerializer#serializeCOOReportToByteArray.

The set of place holders for XML report is the same as those for PDF report.

1.4 Component Class Overview

COOReportGenerator
This interface represents report generator. It has single method to generate COO report based on the given project id.

COOReportSerializer
This interface represents COO Report Serializer.

ComponentDependencyExtractor
This interface represents component dependency extractor. It consists of single method to extract dependencies from the given build file.

ComponentManager
Represents the component manager.

ContestDataRetriever
This interface represents contest data retriever.

Component
Represents a component.

ComponentDependency
Represents component dependency.

COOReport
This class represents a COO Report.

ContestData
This class represents a contest data. This class is used by COOReport to represents the contest data.

DependencyCategory
This enum represents the dependency category.

DependencyType
This enum represents the dependency type.

DefaultCOOReportGenerator
This class is the default implementation of COOReportGenerator interface.

DBContestDataRetriever
This class represents DBContestDataRetriever. It implements ContestDataRetriever interface.

BaseDBConnector
This class represents the base class for classes that would perform some database connection in this component.

DBComponentManager
This class represents DBComponentManager. It implements ComponentManager interface.

BaseCOOReportSerializer
This class represents the base class for report serializers provided in this component.

PDFCOOReportSerializer
This class represents PDF COO Report Serializer.

XMLCOOReportSerializer
This class represents XML COO Report Serializer.

JavaComponentDependencyExtractor
This class represents Java Component Dependency Extractor.

DotNetComponentDependencyExtractor
This class represents .NET Component Dependency Extractor.

1.5 Component Exception Definitions

COOReportGeneratorException

This exception signals that there is error in performing method(s) of COOReportGenerator interface.

InvalidContestCategoryException

This exception signals that the category of the contest is invalid.

COOReportSerializerException

This exception signals that there is error in performing method(s) of COOReportSerializer interface.

ContestDataRetrieverException

This exception signals that there is error in performing method(s) of ContestDataRetriever interface.

ComponentDependencyExtractorException

This exception signals that there is error in performing method(s) of ComponentDependencyExtractor interface.

ComponentManagerException

This exception signals that there is error in performing method(s) of ComponentManager interface.

ConfigurationException

This exception is used to signal that there is error in configuration, e.g. some missing/invalid values.

1.6 Thread Safety

The interfaces defined in this component do not require its implementations to be thread safe.

The entity classes defined in this component are not thread safe since they are mutable. In normal case, the entity classes would not be shared across threads. Hence, the thread safety discussion here assumes that the entity classes are used in thread safe manner. If this is not true, the component is not thread safe.

The base abstract classes: BaseDBConnector and BaseCOOReportSerializer are thread safe since they are immutable, the dependent classes are used in thread safe manner

The concrete implementation classes: PDFCOOReportSerializer, XMLCOOReportSerializer, JavaComponentDependencyExtractor, DBComponentManager DotNetComponentDependencyExtractor, DBContestDataRetriever are thread safe since they are immutable and use the dependent classes in thread safe manner.

Thread safety of DefaultCOOReportGenerator depends on the thread safety of the dependent classes used. If the implementation classes defined in this component are used, then DefaultCOOReportGenerator is thread safe.

Hence in general, the components are thread safe as long as the entity classes are used in thread safe manner.

2. Environment Requirements

2.1 Environment

Java 1.5

2.2 TopCoder Software Components

Configuration API 1.0 – it is used to configure this component.

Base Exception 2.0 – it is used as base of the custom exceptions.

Logging Wrapper 2.0 – it is used to do logging.

Object Factory 2.1 – it is used to create the back end entities DAOs.

Object Factory Configuration API Plugin 1.0 – it is used to allow object factory to work with Configuration API

DB Connection Factory 1.1 – It is used to create database connection

Excel Utility 2.0 – It is used to read Excel File

Document Generator 3.0 – It is used to generate document from template

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

- Open Office API 2.4.3: http://api.openoffice.org/. It is used to open OpenOffice/MS Word template, replace fields and export to PDF

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.management.contest.coo

com.topcoder.management.contest.coo.impl

com.topcoder.management.contest.coo.impl.contestdataretriever

com.topcoder.management.contest.coo.impl.componentmanager

com.topcoder.management.contest.coo.dependencyextractor

com.topcoder.management.contest.coo.serializer

3.2 Configuration Parameters

Configuration parameters for DefaultCOOReportGenerator

Parameter	Description	Values
objectFactoryConfiguration	The child configuration object for object factory	ConfigurationObject. Required.
loggerName	The logger name.	String. Optional.
svnPassword	The svn password.	String. Optional.
svnUsername	The svn username	String. Optional.
componentManagerKey	Object factory key for ComponentManager implementation	String. Required.
dotNetComponentDependencyExtractorKey	Object factory key for ComponentDependency-Extractor implementation for .NET build file	String. Required.
javaComponentDependencyExtractorKey	Object factory key for ComponentDependency-Extractor implementation for .Java build file	String. Required.
contestDataRetrieverKey	Object factory key for ContestDataRetriever implementation	String. Required.

Configuration parameters for BaseDBConnector, DBContestDataRetriever, DBComponentManager

Parameter	Description	Values
dbConnectionFactoryConfig	The child configuration object for DB Connection Factory Implementation	ConfigurationObject. Required.
loggerName	The logger name.	String. Optional.
connectionName	The connection name	String. Required.

Configuration parameters for BaseCOOReportSerializer, PDFCOOReportSerializer, XMLCOOReportSerializer

Parameter	Description	Values
templateFilename	The template filename	String. Required.
loggerName	The logger name.	String. Optional.
defaultOutputFilename	The default output file name	String. Optional.

For PDFCOOReportSerializer, the default value for defaultOutputFilename is "COO_#contestId.pdf"

For XMLCOOReportSerializer, the default value for defaultOutputFilename is "COO_#contestId.xml"

Configuration parameters for JavaComponentDependencyExtractor and DotNetComponentDependencyExtractor

Parameter	Description	Values
loggerName	The logger name.	String. Optional.

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

See demo

4.3 Demo

```
// create COOReportGenerator implementation
// assume configuration already exists
COOReportGenerator generator = new
    DefaultCOOReportGenerator(configuration);
```

```
// generate COOReport for projectId  = 10
COOReport report = generator.generateCOOReport(10);
// serialize the report to PDF
// assume pdfSerializerConfig already exists
// assume that “PDF Template.doc” is used as the template
COOReportSerializer pdfSerializer = new
    PDFCOOReportSerializer(pdfSerializerConfig);

// serialize to file: “project.pdf”
pdfSerializer.serializeCOOReportToFile(report,”project.pdf”);
// serialize to default file
pdfSerializer.serializeCOOReportToFile(report,null);
// the result would be stored in file: “COO_10.pdf”
// serialize to byte array
byte[] pdfReport = pdfSerializer.serializeCOOReportToByteArray(report);
// the resulting PDF output is equivalent to “PDF Report.doc”
// (just that it is in .doc format instead of pdf format).

// serialize the report to XML
// assume xmlSerializerConfig already exists
// assume that “XML template.xml” is used as the template
COOReportSerializer xmlSerializer = new
    XMLCOOReportSerializer(xmlSerializerConfig);

// serialize to file: “project.xml”
xmlSerializer.serializeCOOReportToFile(report,”project.xml”);
// serialize to default file
xmlSerializer.serializeCOOReportToFile(report,null);
// the result would be stored in file: “COO_10.xml”
// serialize to byte array
byte[] xmlReport = xmlSerializer.serializeCOOReportToByteArray(report);
    // the resulting XML output is equivalent to “XML Report.xml”

// exporting the third party list XML file into database
// assume componentManagerConfig already exists
ComponentManager manager = new
    DBComponentManager(componentManagerConfig);
    // add the list defined in “third_party_list.xls” to the database
manager.addComponents(“third_party_list.xls”);
```

Each row in the excel file (in each sheet) would be added as a row in database. For example:

Name	Version	url	License	Usage_comments	Path	Alias	Notes	category
Ant	1.6.5	http://ant. apache.org	Apache 2.0	Build Tool	../tcs/lib/third_party/ ant/1.6.5/ant.jar			Java

5. Future Enhancements

None