Software Documentation : Java Custom Component Dependency Report Generator

This page last changed on May 12, 2008 by zjq.

1. Scope

1.1 Overview

A TopCoder component may have many dependencies, including other TopCoder components or third-party software. This component's task is to generate the detailed dependency report for one or more components. The current version of the component depends on the Component Dependency Extractor component, and uses the extracted dependency information from Extractor component.

1.1.1 Version

1.0

1.2 Logic Requirements

1.2.1 Format of Report

Component dependency report should contain all the components' name, version and language and the dependencies' information. The information for a single dependency may contain its dependency type, category, name, version and path.

1.2.1.1 CSV Format

Each Line of the generated CSV format report should contain the component language-name-version followed by all the dependencies. Each dependency entry could contain its type, category, name, verison and path.

Here is an example: (dependency path not included in the report)

```
java-logging_wrapper-2.0.0,[int][compile]base_exception-2.0.0,[int][compile]typesafe_enum-1.1.0,
[int][compile]object_formatter-1.0.0,[ext][compile]log4j-1.2.14,[ext][test]junit-3.8.2
```

1.2.1.2 XML Format

XML format report should specify the component name and version, with dependencies added as child elements.

Here is an example: (dependency path not included in the report)

```
<component name="logging_wrapper" version="2.0.0" language="java">
<dependency type="internal" category="compile" name="base_exception" version="2.0.0" />
<dependency type="internal" category="compile" name="typesafe_enum" version="1.1.0" />
<dependency type="internal" category="compile" name="object_formatter" version="1.0.0" />
<dependency type="external" category="compile" name="log4j" version="1.2.14" />
<dependency type="external" category="test" name="junit" version="3.8.2" />
</component>
```

NOTE: The details of the report format are subject to change, the principle here is to include all the needed information in the report and make the report easy to understand and use.

1.2.2 Data Feed

The dependency information can be passed to report generator programmatically, or loaded from persistence.

1.2.3 Indirect Dependency

This component should provide the functionality to generate all the dependencies for a component, including direct dependencies and indicret dependencies. (eg. dependency of dependency). The dependency information for a set of components should be passed to the generator component, the information also can be loaded from persistence.

1.2.4 Run Generator from Command Line

The report generator should be able to run as a standalone application. The parameters can be passed by command line options or configuration file. We may want to generate the dependency report for all the components in the catalog, the dependency information can be persisted in a file using Component Depenedency Extractor component.

1.3 Required Algorithms

N/A.

1.4 Example of the Software Usage

This component can be used to generate the dependency report for components. When we need to build a component, this component can be used get all direct/indirect dependencies for the component.

1.5 Future Component Direction

- Support more report format.

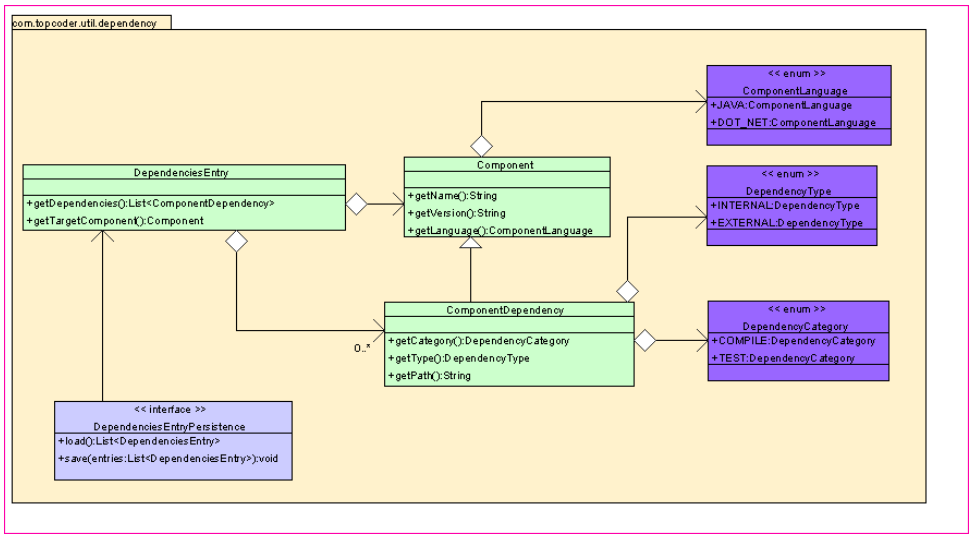
2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None.

2.1.2 External Interfaces

Design must adhere to the class diagram below.



2.1.3 Environment Requirements

- Development language: Java 1.5
- Compile target: Java 1.5 & 6

2.1.4 Package Structure

com.topcoder.util.dependency.report

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

- Location of the persistence file to load the dependency information.
- Location to save the dependency report.
- Format of the dependency report.
- Dependency types to be included in the report.
- Dependency categories to be included in the report.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

XML is required for the format of the configuration file.

3.2.2 TopCoder Software Component Dependencies:

- Command Line Utility 1.0
- Component Dependency Extractor 1.0

**Please review the TopCoder Software component catalog for existing components that can be used in the design.

3.2.3 Third Party Component, Library, or Product Dependencies:

None.

3.2.4 QA Environment:

- Solaris 7
- RedHat Linux 7.1
- Windows 2000
- Windows 2003

3.3 Design Constraints

The component design and development solutions must adhere to the guidelines as outlined in the TopCoder Software Component Guidelines. Modifications to these guidelines for this component should be detailed below.

3.4 Required Documentation

3.4.1 Design Documentation

- Use-Case Diagram
- Class Diagram
- Sequence Diagram
- Component Specification

3.4.2 Help / User Documentation

- Design documents must clearly define intended component usage in the 'Documentation' tab of Poseidon.