Software Documentation : Java Custom Distribution Tool

This page last changed on Jan 04, 2010 by volodymyrk.

1. Scope

1.1 Overview

For each component design TopCoder provides a distribution which includes directory structure, build files, requirements documents etc. Creating these distributions manually for each component is very laborious and so a special tool was developed to automate the process. The tool takes a Requirements Specification file, component name, package name etc. and creates a distribution based on a script. Different languages (e.g. Java, .Net, Flex etc.) have different scripts which produce different distributions. The tool is written on C++ and has command-line interface.

The goal of this competition is to further automate the process by excluding any human intervention in creating design distributions. The Java component needs to replace the current tool and provide an API for the application (cockpit) to create the distributions automatically.

The current tool and its source is provided as a reference. Designers are free to reuse the overall design of the current tool however they are responsible for any decision they make.

1.2 Logic Requirements

1.2.1 Existing functionality

The component will fully support the existing functionality of the current tool. This means that it will be able to create the same distributions that the current tool creates if given the same inputs. Designers need to get familiar with the current tool and the scripts used for different languages. Below is a short description of the current tool's capabilities.

1.2.1.1 Inputs

The inputs of the current tool are:

1. Language (e.g. Java, .Net, Flex etc.)
2. RS document (a PDF file on a filesystem).
3. Component name
4. Package name
5. Version number (optional)
6. Description (optional)

1.2.1.1 Supported commands

1. The tool can create distribution directories. Note that directory structure under src directory depends on the package name.
2. The tool can copy files.
3. The tool can copy files and replace the variables in it (e.g. the component name in the version file etc.)
4. The tool also supports defining new variables but that is just a convenient feature to reduce size of the scripts.

1.2.2 Distributions configuration

The component needs to be flexible enough to support any distribution structure and should not depend on any of the inputs. For example, it will be possible to create new distribution types or change the existing ones by simply changing the scripts and/or configuration files, no code changes will be needed. Also, the component will not depend on the preconfigured inputs like the RS or component name, instead it should be flexible enough to support any input.

Limitations of the current tool

Some of the hardcoded inputs of the current tool (RS, component name etc.) are required and the tool won't work if a distribution does not need them. This should be accounted for in the new tool.

1.2.3 Executing command

Besides the commands supported by the current tool the component will support running external commands from a configured directory.

For example, it will be able to call ant or MSBuild to pack the created distribution into a single archive file.

1.2.4 Inputs

The tool needs to support two types of inputs:

1. String properties. String properties can be used to set the language, component name, version number, package name etc.
2. Files on a filesystem. These inputs can be used to set RS document or any other supporting documentation.

1.2.5 Exceptions

In case of any error the component will throw proper exceptions. At minimum the following errors will be accounted for:

1. The configuration is missing or invalid.
2. The input is wrong or incomplete.
3. An error occurred during creating a distribution (e.g. missing file, filesystem error, external command executing error etc.).

1.2.6 Thread-Safety

The component needs to be thread-safe. Since the component reads and writes from the filesystem a special care should be taken to make sure it can be run from different threads simultaneously.

1.2.7 Scripts

Designers are responsible for providing sample configuration and/or script files for Java and .Net distributions. These scripts will duplicate the Java and .Net scripts for the current tool.

1.3 Required Algorithms

Designers need to describe the algorithms involved in script parsing and processing (if any).

1.4 Example of the Software Usage

The component will be used by cockpit to automatically create component design distributions when a user launches component competition.

1.5 The current tool

The current tool and its source (for Microsoft Visual C++ 2005) is provided as a reference.

Steps needed to set up the tool:
1) Unpack DistributionTool.zip archive.
2) Set "DIST_HOME" environment variable to the directory where you unpacked the DistributionTool.zip.
3) Add the directory where you unpacked the DistributionTool.zip into the "PATH" environment variable.

1.6 Future Component Direction

The main concern is to make the component as flexible as possible so that only configuration change is needed if a new language or distribution must to be supported later. Therefore any flexibility enhancement to the component is encouraged.

However **any enhancement needs to be approved** either in forum or in email with managers to eliminate over-complicating the component with useless functions.

The following enhancements are already approved:

1. File masks support. When copying an input file it will be useful to change the file name but leave the extension as is. For example, user can provide RS document in RTF or PDF format and the tool will only change the file name but leave the extension as is (i.e. .rtf or .pdf).
2. RTF->PDF and DOC->PDF conversion. User can provide RS document in different formats and it will be useful if the tool could automatically export it to PDF format.

2. Interface Requirements

2.1.1 Graphical User Interface Requirements

None, only API interface will be provided.

2.1.2 External Interfaces

None.

2.1.3 Environment Requirements

- Development language: Java 1.5
- Compile target: Java 1.5, Java 1.6

2.1.4 Package Structure

com.topcoder.util.distribution

3. Software Requirements

3.1 Administration Requirements

3.1.1 What elements of the application need to be configurable?

The whole logic for creating distributions needs to be configurable.

3.2 Technical Constraints

3.2.1 Are there particular frameworks or standards that are required?

None.

3.2.2 TopCoder Software Component Dependencies:

- Base Exception 2.0
- Configuration API 1.0
- Configuration Persistence 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:

Any third party library needs to be approved.

3.2.4 QA Environment:

- Java 1.5
- RedHat Linux 4
- 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
- Sample scripts and/or configuration files for Java and .Net distirbutions

3.4.2 Help / User Documentation

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