Project Control File

by Bill Rich

Describes the Project Control File content and use.

Table of contents

1. Overview

The Project Control File is a list of files to process with some detail to control what happens to each file. It is used in the tool kit and the process to control when and what to do for each source file in the project. Information as to the name of the source file, what type of file it is, whether to translate it or not, and how to form the target file path and name must be included in the Project Control File.

The Project Control File is written as an XML file with the main container being the `files` element. The `files` element contains a set of `file` elements. Each `file` element describes a single file to process. The information describing each file is:

- the file type
- whether to translate the file or not
- the source file name
- the target file name pattern

The format and layout of the Project Control File is very simple, the file begins with a `files` elements and ends with a close `files` element in the typical XML file coding practices. We also include a `comment` before the `files` element to show when the project control file was created. This can be important information when the project is large and continually changing. The `comment`, of course, is not essential and not used by any of the classes in the toolkit. Following is an example of a project control file:

```
<!-- Created on 2005/07/27. -->
<Files>
<File type="LRB" translate="XL" inputname="LRB01.java" outputname="NLS/@lang@/LRB01_@lang@.java"/>
<File type="PRB" translate="XL" inputname="PRB01.properties"
outputname="NLS/@lang@/PRB01_@lang@.properties"/>
<File type="JSP" translate="XL" inputname="JSP01.jsp" outputname="NLS/@lang@/JSP01.jsp"/>
<File type="HTML" translate="XL" inputname="HTML05.html" outputname="NLS/@lang@/HTML05.html"/>
<File type="XML" translate="XL" inputname="XML01.xml" outputname="NLS/@lang@/XML01.xml"/>
<File type="MC"  translate="XL" inputname="MC01.mc" outputname="NLS/@lang@/MC01.mc"/>
```

```
<File type="RC"  translate="XL" inputname="RC01.rc" outputname="NLS/@lang@/RC01.rc"/>
<File type="JS"  translate="XL" inputname="JS01.js" outputname="NLS/@lang@/JS01.js"/>
<File type="CSS" translate="XL" inputname="CSS01.css" outputname="NLS/@lang@/CSS01.css"/>
<File type="SQL" translate="XL" inputname="SQL01.sql" outputname="NLS/@lang@/SQL01.sql"/>
</Files>
```

Note that this sample file contains one of each type of file that can be processed by the G11NToolKit.

Return to: Top of page

2. File Type

The `type` attribute identifies the type of file being described by the `file` element. There are a restricted set of file types that can be processed and they are case sensitive when used in the Project Control File. They are:

`LRB`
Java `ListResourceBundle` file. This file contains strings that may be translated. There are guidelines as to how to code this file that can be found at Java ListResourceBundle page. If these guidelines are followed by the developers these files can be handled directly by the G11NToolKit and the strings that need translation can be extracted to a string extract file that can be processed by some translation tools.

`PRB`
Java `PropertyResourceBundle` file. This file contains strings that may be translated. There are guidelines as to how to code this file that can be found at Java PropertyResourceBundle page. If these guidelines are followed by the developers these files can be handled directly by the G11NToolKit and the strings that need translation can be extracted to a string extract file that can be processed by some translation tools.

`JSP`
Java Server Page file. JSP technology uses XML-like tags that encapsulate the logic that generates the content for the page. It allows formatting with HTML or XML tags.

`HTML`

HyperText Markup Language file. HTML files contain the formatting instructions for Web pages as well as the content of the pages.

XML

EXtensible Markup Language file. XML was designed to describe data and to focus on what data is. There are elements, attributes, and attribute values that make up the language. None of these are fixed. It is generally the attribute values that are of interest in localization.

MC

Application Message Catalog file. Application message catalogs are used to organize messages from an application program. They generally consist of entries for each message with a message identifier and the message text. It is generally the message text that is of interest in localization.

RC

Resource Catalog file. Like many of the other file types the RC file contains identifiers and values for resources used in a program. It is generally the values that are of interest to localization. There may be some keyword elements that are also of interest. The G11NToolKit focuses on the values that may need translation for extract to the string extract file.

JS

JavaScript file. JavaScript is a scripting language - a scripting language is a lightweight programming language. JavaScript was designed to add interactivity to HTML pages and is nothing more than lines of executable computer code. JavaScript can contain strings that may need localization which are very difficult to identify in the code.

CSS

Cascading Style Sheet file. A Cascading Style Sheet is a markup language to format data for presentation of information on computers and other devices. It can contain some strings that may need localization. Having CSS files contain strings should be discouraged as much as possible.

SQL

SQL file. An SQL file can be either a Data Definition Language (DDL) file or a Data Manipulation Language (DML) file. Generally DDL files should not contain strings that may need localization but DML files may.

Return to: Top of page

3. Translate Flag

The translate attribute represents the Translate Flag. It says that the file should be translated (`XL`) or not translated (`NXL`). This flag is handy to maintain a complete list of files in a project but not need to translate every one of them.

Return to: Top of page

4. Source File Name

The source file name identifies the source file that may need translation. The name should begin at a directory just below the `Product` directory. The `Product` directory is specified in the `Project` properties file or on the command line when executing the `l10nproj.xml` Ant file. The `Product` directory name will be added in front of the source file name as needed in the process. The source file name is also used as the context file name for the extracted strings.

Return to: Top of page

5. Target File Name

The target file name is really a file name pattern that will be completed when it is needed in the process. There is currently one insert variable that can be placed in a target file name pattern. That is the `@lang@` variable. It indicates where the language code should be placed in the target file name. The language is usually specified on the command line when executing the `l10nproj.xml` Ant file. The language should be a two character language identifier. Note that in the sample it is used as both a directory name and a modifier for the Java `ListResourceBundle` file name. In the case of the `LRB` file it will also be used to rename the class defined in the file since the class name and the file name must match.

Return to: Top of page