Tokenizer User's Guide

by Bill Rich

User's Guide for the G11NToolKit Tokenizers.

Table of contents

1. Overview

The G11NToolKit `Tokenizers` will extract strings from source files and replace them with token identifiers. The source files offered to the `Tokenizers` must be of a known type and must have a specific Tokenizer identified to process them.

Each `Tokenizer` extracts the strings from a file and leaves a unique token in place of each extracted string. The token is what enables the `Detokenizers` to put the string back into the target file. It is important that the tokens assigned by the `Tokenizers` to the strings are not changed in any way. If you change the tokens or their association with the strings, the `Detokenizers` may not be able to put the strings where they belong in the target files.

Each Tokenizer writes all of its extracted strings into an XLIFF formatted file called a `StrFile`. All StrFiles have the file extension `.str` and are encoded as UTF-8. These are source only XLIFF files that may need to be transformed before a particular translation tool can use them. Since they are XLIFF files in XML format they are easily transformed by applying an XSLT file to them.

A `Tokenizer` is written to process a specific type and style of source file. They may be useful for another type or style of file but this is not guaranteed nor recommended. Tokenizers are provided for the following types of files:

File Type	Tokenizer Class	Description
Java Properties	`com.g11ntoolkit.tokenize`	These files usually have an extension of `.properties`. They are a set of key and value pairs with the key separated from the value by the equal sign (=). See the Java web site for more details on how to code these files. Comments in the files are ignored by the Tokenizer.

Java List Resource Bundle	`com.g11ntoolkit.tokenize`	These files usually have an extension of `.java`. The G11NToolKit Tokenizer assumes that these files are normal Java List Resource Bundles. This means that each file has a content section that is an array of entries each with a key and a value. In general the value is a quoted string but the key may or may not be a quoted string. The file will also contain a get method for the contents that returns the whole content section. This is all spelled out on the Java web site but is not guaranteed by the Java Compiler. There are many ways to write an LRB, but, the Tokenizer provided in the tool kit does not support them all. Some variations on the standard may work. Check carefully to make sure the Tokenizer handles the formats of the files you intend to process.
Java Server Page	`com.g11ntoolkit.parser.h`	These files usually have an extension of `.jsp`. There are standards that govern the content and style of these files.

		The Tokenizer provided with the G11NToolKit assumes the files follow these standards.
HTML	`com.g11ntoolkit.parser.h`	These files usually have an extension of `.html` or `.htm`. There are standards that govern the content and style of these files. The Tokenizer provided with the G11NToolKit assumes the files follow these standards.
Cascading Style Sheet	`com.g11ntoolkit.tokenize`	These files usually have an extension of `.css`. There are standards that govern the content and style of these files. The Tokenizer provided with the G11NToolKit assumes the files follow these standards.
JavaScript	`com.g11ntoolkit.tokenize`	These files usually have an extension of `.js` or other extensions. These files are basically text files that contain JavaScript code. The Tokenizer for them extracts each line of text for translation and leaves many of the decisions up to the L10N engineer as to which strings actually need translation.

Resource Catalog	`com.g11ntoolkit.tokenize`	These files usually have an extension of `.rc`. These files contain C code describing the resources for a C program. The Tokenizer assumes that quoted strings need translation and extracts only those strings.
Structured Query Language (SQL) Data Definition Language (DDL) or Data Manipulation Language (DML)	`com.g11ntoolkit.tokenize`	These files usually have an extension of `.sql`. These files can contain translatable strings. The Tokenizer tries to find the translatable strings and extract them to the `StrFile`.
Message Catalog	`com.g11ntoolkit.tokenize`	These files usually have an extension of `.mc`. These files contain message strings that may need translation. There are standards that control the content and style of these files. The Tokenizer assumes the standards are followed.
XML	`com.g11ntoolkit.tokenize`	These files usually have an extension of `.xml`, but, may have any other extension desired by the user. These files must be standard XML files and include the XML file header. The Tokenizer uses a set of regular expressions to find and

		extract the strings that need translation.
XSL	`com.g11ntoolkit.parser.h`	These files usually have an extension of `.xsl`, but, may have any other extension desired by the user. These files must be standard XSL files and include the XSL file header.

If a file is not one of these then chances are that there is no `Tokenizer` for it. If a file type that is not in the list is required, a `Tokenizer` extension can be implemented to handle the file correctly.

Each tokenizer is written as an independent tool that can be executed from the command line. To make things easier the process control (`L10NProcess...`) Ant files are available and contain Ant targets to determine when to use each of the tokenizers based on the contents of the Project Contol File. All tokenizers are controlled by targets in the `L10NProcess-toktasks.xml` file.

Return to: Top of page