CTS Universal, beta-03

Neel Smith <neel_smith@users.sourceforge.net>

Version of Aug 26, 2010, coordinated with the beta-03 release
CTS Universal, built with the beta-06 version of the cts library.

Table of Contents

About this release

This document accompanies a beta version of CTS Universal, an implementation of the CTS protocol for identifying and retrieving texts by canonical reference. The binary released is compiled against versions of the JPA libraries older than those currently released with Google's AppEngine package, but runs unchanged when hosted on Google's servers, and passes 100% of the tests in version 3 of the CTS test suite. For local use with the `dev_appserver.sh` script in Google's App Engine SDK, this version has been tested only on a limited range of versions.

Caution

Documentation for the pre-release vesion of CTS Universal is still modest. Up to date information is available from the cts3 sourceforge site *http://cts3.sourceforge.net/*

Prerequisites

CTS Universal is written to the Java servlet standard: all versions of it require Java to run.

To use the App Engine version of CTS Universal, you must also have Google's App Engine SDK for Java installed. If you don't already, you can download an installer *from this page* [http://code.google.com/appengine/downloads.html]. If you want to install your service on a Google host, you'll need a Google App Engine account: follow the instructions on the *Google App Engine main page* [http://code.google.com/appengine/].

The distribution of CTS Universal for RDBMS includes a self-contained jetty servlet container with CTS Universal installed: you can run this servlet container directly. The distribution also includes a `.war` file you can be install in any servlet container.

Overview: managing a CTS

A CTS has two parts: a repository of texts, and a TextInventory, which is a catalog of the texts in the repository. In CTS Universal, the repository is stored in a data source accessible through the

Java Persistence Architecture (JPA). CTS Universal includes utilities for uploading tabular data and for deleting data, using JPA. The inventory is managed in a local XML file validating against the TextInventory Relax NG schema.

To manage a CTS, you therefore need to manage three tasks:

1. configure and run the service
2. add texts to the JPA repository
3. edit the inventory for your repository

Configuring and running the server

Defining administrators

Summary

Edit `cts3/configs/pass.xml`

CTS Universal includes servlets for uploading and deleting data. The file `cts3/configs/pass.xml` contains a list of users who are allowed to use these servlets to edit the data store. If you are running CTS Universal for Big Table, the application will use Google's user authentication: the list in `cts3/configs/pass.xml` should include the username for the authorized google account.

If you are running CTS Universal with a relational database, the servlet will check a given login name against this list, and grant permission to upload or data data to any supplying a name found in the list.

Caution

If you need real security, or are worried about someone hacking in to your data store, do not rely on this checking for security.

Configuring a Google application

Summary

Edit `cts3/WEB-INF/appengine-web.xml`

If you want to install CTS Universal compiled for Big Table on a Google host, you must first get a Google account (see prerequisites), and create a Google application.

Then, in the file `cts3/WEB-INF/appengine-web.xml`, supply the name of your Google application, and a version number (integers only).

Starting the service

Summary

Local appengine: **dev_appserver.sh cts3**

Appengine on google: **appconfig.sh update cts3**

Stand-alone service: **java -jar jetty/start.jar**

CTS Universal can be run in any servlet container. If you have downloaded CTS Universal compiled for Big Table, you can use the scripts from the Google SDK to run the service locally, or upload CTS

Universal to a Google host. To run the service within a local App Engine host, use the command **dev_appserver.sh cts3**

The service should then be available from `http://localhost:8080`.

To upload your CTS to a Google app, use the command **appconfig.sh update cts3**

The service should then be available from `http://YOURAPPNAME.appspot.com`.

If you have downloaded CTS Universal compiled for Relational Databases, the package includes a stand-alone servlet container. You can start this conatainer with the command **java -jar jetty/start.jar**

The service should then be available from `http://localhost:8081/cts3`.

Managing the data repository

Loading and deleting data

Summary

Go to `http://YOURCTS/dbadmin/admin`

Loading data

From the home page of your CTS installation, you will find a link to a servlet at `dbadmin/admin`. (Access to this servlet is limited to users listed in `cts3/configs/pass.xml`: see above.) From this page you can load data in tabular format, or delete data identified by CTS URN.

CTS Universal includes a special directory named `uploads` where you can place data files for uploading. The admin servlet will automatically offer links to any files found in the updates directory. The sample files in this distribution include one file with the data used by version 3 of the standard CTS test suite; the other files are a small selection of literary and technical texts used at `http://cts-demo.appspot.com`. You can load any or all of these files to test your CTS installation.

In addition, if you are running CTS Universal compiled for Relational Databases, you may identify a URL with download to load. (This option is not currently implemented for CTS Universal compiled for Big Table.)

If the entire file is successfully parsed and loaded, you will see a success message at the end of the , Complete. Successfully uploaded all data into repository. If you do not see this message, some of your data failed to load. (This can happen if your connection times out when uploading very large data sets; you can work around this by breaking up your upload in multiple, smaller files: see the following section, "Preparing new data to load.") The servlet avoids reloading data already in the data store. If you need to replace an entire text, you should first delete it, as described in the following section, then reload. Of course, you could also directly edit data in your Big Table or RDBMS data source if you need to correct individual errors.

Deleting data

The admin servlet lets you delete records from your service by CTS URN. Be sure that you understand the the scope of the URN you are using. For example, if you have multiple versions of a work in your CTS, and identify that work by a URN at the work level, you will delete all versions of the work.

To help prevent unintended disasters, the admin servlet provides a search form where you can search for records by URN. You can use this to be sure you understand the scope of a URN before using the form to delete records by URN.

There is also a (self-explanatory) option to delete all records in your data store.

Preparing new data to load

CTS Universal includes a small, standalone application named XML Tabulator that creates tabular data for uploading to CTS Universal from XML source texts described in a TextInventory.

If you are reading this file, XML Tabulator should be correctly installed already. You can move the tabulator directory anywhere you like, but do not move the location of the lib directory and Tabulator.jar file within the directory.

You'll need one or more XML texts, and a TextInventory that validates against the version 3.0 TextInventory schema documenting them. The Tabulator package is distributed with a directory called "testcorpus" that includes a sample inventory, the Relax NG schema it validates against, and the XML files described in the inventory. You can test your validator installation with these data sets, or use them as a model for creating your own inventory of xml texts to tabulate.

On some operating systems, you should be able to start the XML Tabulator by double-clicking `Tabulator.jar`.

In any case, **java -jar Tabulator.jar** should work.

In the main application window, you must first select a TextInventory, a directory with the XML texts documented in the TextInventory, and a directory where the new tabular files should be saved. You may optionally define the number of citation units to store in an individual file. The default of 300 units is fine when running locally with **dev_appserver.sh**, but for most texts is probably too high for Google's backend to process without timeout. The optimum size will depend on the typical size of your citable units. For works cited by poetic line, 100-200 units is probably a better initial guess; for prose works with large sections, 100 units might be a safer first guess.

Select one or more texts from the list at the bottom of the pane, then from the "Tabulate" menu, choose "Tabulate selected online texts in inventory" (control-T for short, or apple-T on Macintosh systems).

Text files will be added to the specified output directory, with the file name defined by the work component of the text's CTS URN plus "-tab-N.txt" where N is a number for each file written for that text.

If you place the tabular files in your server's `uploads` directory, or post them to the internet, you can use the admin servlet to load them in your CTS, as described in the previous section.

Managing the TextInventory

Caution

The TextInventory for the CTS test suite, included by default in the `inventory` directory, is a complex document because the test data set is deliberately designed to test difficult cases. You may need only a very simple inventory for your application. To get started editing your own TextInventory, the inventory for the cts-demo set of documents is a better model to start from.

When you add or delete data from your CTS repository, you have to be sure that you update your TextInventory to reflect the current contents of your CTS.

In the `inventories` directory, you will find a sample inventory for the CTS test suite, and two Relax NG schemes that can be used to validate TextInventory documents that you create.