LexEVS 6.0 Design Document - Detailed Design - Loader Post-Processor

Document Information

Author: Craig Stancl
Email: Stancl.craig@mayo.edu
Team: LexEVS
Contract: CBITT BOA Subcontract# 29XS223
Client: NCI CBIIT
National Institutes of Heath
US Department of Health and Human Services

Revision History

Version	Date	Description of Changes	Author
1.0	5/14/10	Initial Version Approved via Design Review	Team

In order to facilitate extra Processing at the end of an Ontology load, LexEVS 6.0 will support *Loader Post Processors*.

A Loader Post Processor is logic that will be executed when the actual content load to the database is complete, but before the Lucene Indexing occurs. It will be implemented as an *Extension* - meaning that users may place jars in the LexEVS class path and introduce Loader Post Processors without the need to recompile.

The Loader Post Processor interface is straightforward:

```
public interface LoaderPostProcessor extends GenericExtension {

public void runPostProcess(AbsoluteCodingSchemeVersionReference reference);
}
```


Note

This is considered a *GenericExtension* and must be registered and declared as such. A Loader Post Processor may apply any logic that is required -- there are not constraints as to the scope. A Loader Post Processor may update database content, delete content, reference other loaded ontologies, etc.

An example Loader Post Processor is shown below -- this example will update the Approximate Number of Loaded Entities field of a Coding Scheme:

```
package org.LexGrid.LexBIG.Impl.loaders.postprocessor;

import org.LexGrid.LexBIG.DataModel.Core.AbsoluteCodingSchemeVersionReference;
import org.LexGrid.LexBIG.DataModel.InterfaceElements.ExtensionDescription;
import org.LexGrid.LexBIG.Exceptions.LBException;
import org.LexGrid.LexBIG.Exceptions.LBParameterException;
import org.LexGrid.LexBIG.Extensions.Generic.GenericExtension;
import org.LexGrid.LexBIG.Extensions.Load.postprocessor.LoaderPostProcessor;
import org.LexGrid.LexBIG.Impl.Extensions.AbstractExtendable;
import org.LexGrid.LexBIG.Impl.Extensions.ExtensionRegistryImpl;
import org.LexGrid.codingSchemes.CodingScheme;
import org.lexevs.dao.database.service.codingscheme.CodingSchemeService;
import org.lexevs.dao.database.service.entity.EntityService;
import org.lexevs.locator.LexEvsServiceLocator;

public class ApproxNumOfConceptsPostProcessor extends AbstractExtendable implements LoaderPostProcessor {

private static final long serialVersionUID = 2828520523031693573L;

public static String EXTENSION_NAME = "ApproxNumOfConceptsPostProcessor";

public void register() throws LBParameterException, LBException {
   ExtensionRegistryImpl.instance().registerGenericExtension(
      super.getExtensionDescription());
   }

   @Override
   protected ExtensionDescription buildExtensionDescription() {
       ExtensionDescription ed = new ExtensionDescription();
       ed.setDescription("ApproxNumOfConceptsPostProcessor");
       ed.setName(EXTENSION_NAME);
       ed.setExtensionBaseClass(GenericExtension.class.getName());
       ed.setExtensionClass(this.getClass().getName());

       return ed;
   }

   public void runPostProcess(AbsoluteCodingSchemeVersionReference reference) {
       EntityService entityService = LexEvsServiceLocator.getInstance().getDatabaseServiceManager().
getEntityService();
       CodingSchemeService codingSchemeService = LexEvsServiceLocator.getInstance().getDatabaseServiceManager().
getCodingSchemeService();

       String uri = reference.getCodingSchemeURN();
       String version = reference.getCodingSchemeVersion();

       long entities = entityService.getEntityCount(uri, version);

       CodingScheme codingScheme = codingSchemeService.getCodingSchemeByUriAndVersion(uri, version);

       codingScheme.setApproxNumConcepts(entities);

       codingSchemeService.updateCodingScheme(uri, version, codingScheme);
   }
}
```