XML Data Adapter Proof-of-Concept

Technical Development Considerations

Glen Oliff
Stephanie Liu-Barnes
Alexis Karnauskas

August 31, 2007

Introduction

This document provides a list of items to consider when editing or augmenting the HydroXC proof-of-concept XML data adapter. The content within this document is written for Java developers and assumes an overall understanding of Java code and architecture.

Handling XML Schema Updates

Most of the essential files in the HydroXC XML data adapter proof-of-concept are XML files. As such, each XML file type has its own associated schema file. Through the advancements in JAXB (Java API for XML Binding) 2.0 (available in Java 1.6, which is also known as Java 6), the XML marshalling and unmarshalling to Java objects is made possible by classes generated off of schema files.

Moving forward, anytime a schema file is updated (location mapping, parameter mapping, or even the HydroXC schema itself), you should rerun the associated shell scripts to update the generated XML binding classes. These shell scripts (written for Windows, easily ported to UNIX) exist in the high level directory upon installation. The naming format is rebind_<type>.cmd, where <type> corresponds to the schema file for which updates were made against. These scripts aid in wrapping up the proper call to Java's xjc utility, which generates the java classes off of the schema to the appropriate output location, and with the proper package structure.

Parameter Annotations for the Writing Component

In the SHEFHydroXCExtractor.java file, there is a reference to the ParameterAnnotationProcessor class. Apex has updated the code coming from HydroXC XML with custom annotation processing for mappings between bean methods. Apex did not update the "from" shef to HydroXC handling to utilize this type of functionality. Augmenting the "from" functionality to handle this will enable much quicker extensibility for addition of parameters in the future. We have pointed this out to highlight a "best practice" moving forward for future data adapter development.

Extraction API for Future Development

In the future, when multiple HydroXC data adapters have been developed, it will be necessary to create an extraction/insertion API that each data adapter could call on to push to/pull from the HydroXC XML binding classes. This would expedite development and limit the amount of redundant error handling for cases where "node doesn't exist", or "data not found" errors occur. Currently, you will see that this is missing from the proof-of-concept data adapter. There are quite a few instances of DataElement.get<thing>.getOtherThing() throughout. With this extraction API created, these would be replaced by a call to otherThing = extractor.getOtherThing(dataElement, thing). Apex wanted to explicitly call this out as a first step upon creation of the next data adapter type.