Wirehose-Rapid Revelopment

The WireHose frameworks are designed to help you build content management and portal applications faster. You can use them to build new applications from scratch, or you can add WireHose functionality to an existing codebase, usually without subclassing WireHose objects.

Xcode templates

WireHose provides Xcode templates for creating new WireHose applications and frameworks. It also includes templates for rapidly building new channels, channel factories, taggable and indexable entities, as well as components for rendering and editing channels and resources.

Using interfaces instead of subclassing

WireHose defines several Java interfaces which you can implement, enabling you to add WireHose support to your existing enterprise object classes. **WHEnterpriseObject** provides methods for automatically discovering entities which implement an interface, and performing fetches against all entities which support a given interface. **WHFetcher** defines an interface for fetching, caching and retrieving objects based on an entity name or interface, and **WHCachingDataSource** provides a default implementation, which also doubles as an EODataSource so you can use it to drive display groups. Most interfaces provide a default implementation, so the amount of code you need to write to support a particular interface is minimal.

Creating entities at runtime

WireHose makes extensive use of entity inheritance, taking advantage of the fact if an entity is not visible at runtime, any database rows described that entity are simply unavailable to the application. This is a simple but effective way to partition objects between separate application instances which share identical codebases and differ only in configuration files or launch arguments. For example, if you are deploying multiple news portals, users connecting to the Seattle portal should only see Seattle-area traffic cams, and Portland users should only see Portland-area traffic cams, but both should have access to national newsfeeds.

WHEnterpriseObject provides several methods for dynamically creating subentities at runtime, so you don't have to manually model many common types of inheritance in EOModeler. These methods are especially convenient for horizontal and single-table inheritance.

WHEnterpriseObject also includes special support for "affiliate-based inheritance", which is used by many WireHose entities. In this approach, you define an attribute on your base entity called "affiliate". This attribute is used with a restricting qualifier to identify subentities. **WHApplicationHelper** and **WHTag** use a "default affiliate" to determine which subentity should be used. WHApplicationHelper also lets an application access multiple affiliates simultaneously, and WHEnterpriseObject can automatically create subentities for all available affiliates.

Database independence

WHEnterpriseObject makes it simple to switch between databases when an application is launched. EOModel connection dictionaries can be replaced on the fly through command-line arguments, so you can use a different database during development versus deployment, or deploy multiple instances of the same application against multiple database backends. WireHose also makes extensive use of attribute prototypes so models don't have to be modified even when switching between different database vendors. In addition, since databases vary in how boolean values are supported, WHEnterpriseObject provides utility methods which transparently set and retrieve booleans, whether the columns are defined as INT, BOOLEAN or CHAR.