OGNL replacement

OGNL replacement

This page is dedicated to replacing OGNL from all developer facing locations. The first step to tackling this is to catalog them all.

Catalog

The catalog contains all the locations that OGNL is used that developers have access to. An example of a place that OGNL is used that developers don't have access to is the processing HTTP request parameters into JavaBean properties. Here is the list:

- Struts configuration
    - Result parameters
        - This isn't done by the framework, but rather in a base-class, which is carefully extended by some result implementations but not others. Leaving it up to the various implementations sounds like a recipe for making this plugability difficult. Why not push this into the framework and free the results implementations to focus on implementing the result rather than determining their configuration. Suggested here: http://www.nabble.com/Dynamic-Results-tt15488767.html#a15504929
    - Any other locations where OGNL can be used in configuration files?
- Struts taglibs for nearly all attributes
    - And lots of custom tags built to satisfy specific projects. A clean solution can also be used for these custom tags.
- validation xml files
- default type conversion (
- com.opensymphony.xwork2.util.XWorkBasicConverter
- com.opensymphony.xwork2.util.XWorkTestCaseHelper
- com.opensymphony.xwork2.util.InstantiatingNullHandler
- com.opensymphony.xwork2.util.OgnlValueStack.ObjectAccessor
- com.opensymphony.xwork2.util.ObjectProxyPropertyAccessor
- com.opensymphony.xwork2.util.OgnlValueStack
- com.opensymphony.xwork2.interceptor.ParametersInterceptor
- com.opensymphony.xwork2.util.XWorkCollectionPropertyAccessor
- com.opensymphony.xwork2.util.XWorkListPropertyAccessor
- com.opensymphony.xwork2.util.XWorkMapPropertyAccessor
- com.opensymphony.xwork2.util.XWorkMethodAccessor
- com.opensymphony.xwork2.util.XWorkObjectPropertyAccessor

Issues

The specific issues that need to be addressed are:

- OGNL is inconsistent and incompatible with the UEL
- Inconsistent use of %{} and ${} notation in tags vs results
- There's no evaluate now vs. evaluate later concept (UEL ${} vs #{})
- Inconsistent default evaluation in tags and results (some evaluate expression, some don't. Users have no way of knowing without checking the javadoc/code)
- Type conversion is somewhat difficult and not aware of servlet objects
- Security risks keep appearing
- Currently impossible to access certain servlet scopes (JSP include parameters for example)

Goals

- Pluggable EL or simply UEL-compliant??
- @OGNLDependent annotation to place on all classes / methods / fields to indicate reliance on OGNL (useful for eventual refactoring)?