

HiveMindElementImprovementProposal

Problem Statement

(Apr 29 2004, 1.0-alpha-4 just released)

HiveMind gives the developer three options to process configuration point contributions and convert them into a list of objects.

1. Use the `<conversion>` element in the `<configuration-point>` definition. 2. Use `<rules>` in the `>configuration-point>` definition. 3. Not use `<conversion>` or `<rules>` elements in the `<configuration-point>` definition and obtain raw elements for manual processing of contributions outside of HiveMind.

This proposal applies to the third of above three options. When choosing to use the third option a list *org.apache.hivemind.Element*'s are returned by the [HiveMindRegistry](#) when the contributions of a `configuration-point` are obtained. These xml elements are then used directly or are parsed externally to HiveMind. For those who have looked at the architecture of the eclipse platform, this is the only option available as `<conversion>` and `<rules>` elements do not exist in eclipse's plugin descriptors.

When a developer chooses this option they obviously have to deal with *org.apache.hivemind.Element* objects as HiveMind uses its own *Element* and *Attribute* objects and so no other xml processing library understands them. At the moment the *org.apache.hivemind.Element* object has no compatibility methods enabling conversion to a *org.w3c.Element* or to string representation of the XML element. This means that working with *org.apache.hivemind.Element* objects requires an additional utility method to convert the Element to a string to enable it to be used with other xml libraries that the developer may be using.

Solution

Included within the *org.apache.hivemind.Element* object a minimal number of utility methods that allow the developer who chooses to receive raw elements instead of using one of HiveMind's `<conversion>` or `<rules>` xml->object methods can do so more easily.

I would suggest minimally a `String asXML()` method which returns a *String* representation of the Element. Other methods could be included obviously although care should be taken to respect the [YAGNI](#) principal.

Comments

[HowardLewisShip](#): It would not be impossible to create an adaptor that emulated W3C Dom, based on data obtained from HiveMind's Element.

In addition, if you have very sophisticated XML you want to parse, then put it in a separate file and reference it as a *Resource*. You can then use the XML parser of your choice.

[DanielFeist](#)

I agree with your comments but still believe a simple `asXML()` method which returns the string representation of the HiveMind element is not much to ask and very useful for people choosing to deal with these elements directly rather than using `<conversion>` or `<rules>` element in the module descriptor. Nearly all XML libraries such as dom4j, jdom etc have methods for returning a string representation of the element.

[AlbertKwong](#)

The *org.apache.avalon.configuration* package provides a Configuration object that's pretty easy to use. The [HiveMind](#) Element object can be enhanced to provide functions similar to the avalon package, such as `getAttributeAsInteger` (String name, int defaultValue). It also aids the transition from Avalon to [HiveMind](#). I would volunteer to work on it if necessary.