YmtdEmbeddedUsage

Embedded Usage

This is an example of using Velocity within an application.

```
    OutputStreamWriter writer = new OutputStreamWriter(output, encoding);
    Template template = RunTime.getTemplate(filename);
    template.merge( context, writer );
```

In other words, the above is translated into:

1. Create a Writer.
2. Ask the Velocity RunTime to retrieve the Template.
3. Merge the Context with the Template.

Velocity templates can contain **any** type of data. It can be XML, SQL, HTML, plain text, internationalized text, anything. The parser is very robust in terms of only parsing out what it cares about and ignoring the rest. As demonstrated above, it is possible to embed Velocity's template engine into any other Java code. The advantage of this is that it provides users with a single template system for all sorts of uses.

For example, in the PetStore example on the J2EE website, there is a JavaBean which sends an email confirmation when an order is complete. The unfortunate part about this example is that in order to change the contents of the email, one needs to edit Java code. Of course this example could have been done differently to use a JSP based template as the base format. However, the implementation of this is more difficult than simply creating a Java object which resides in the Context that can take as arguments the template to render.

Another example of this is the Texen and Anakia tools that come with Velocity. The advantage again is the ability to use Velocity as an embedded tool to produce other useful applications that are not bound strictly to a web application. While it is most likely possible to do this with the JSP engine, it is significantly more difficult to do so.

You make the decision.

YmtdTaglibs <- Previous | Next -> YmtdImplementation