weave + meta

An object-relational dream team

What is weave?

weave is a framework for the web which tries to alleviate the repetitive and sometimes cumbersome tasks of the average web developer. It does so by imposing an MVC[1] architecture in which weave acts as the controlling part of the application. At the time of this writing, weave is split across several packages and consists of an Annotation API and a set of Interfaces. Along with the specification of these there are two implementations available, called FRIGGA and SKULD. Skuld is just an experimental branch for exploring new ideas without breaking existing code.

The view-part of your application is handled via XSL-T[2] (other mechanisms are imaginable). You need to provide templates written in XSL-T which transform the output of your application to a full blown website.

That's where the model-part comes in: your application. Typically it consists of a set of modules, based on a generic module implementation provided by weave, and a controller, which in most cases won't be anything else but a generic controller (also provided by weave) which knows which modules are to be loaded at startup.

Beyond that, weave does not only ease the way your application talks to the outer world, but also how your application works inside. It does so by also providing a solution to the typical problem of Object/Relational Mapping.

The ORM framework provided by weave tries to solve the impedance mismatch between the object-oriented approach for creating a data model and the relational approach by creating a new *object-relational* approach. A relational model is neither a relational one only nor an object oriented: it is both. The object-relational model is developed as a whole and tries to pay attention to both worlds.

weave in that scenario provides interfaces (including annotations) which back the ORM-Framework, but the actual development of the object-relational model takes place somewhere else.

[1] model, view, controller – a design pattern that separates business logic from the rest of your application

[2] eXtensible Style Language – http://www.w3.org/TR/xslt

The object-relational model

(still to come)

meta

The euphonious name of META was chosen since it actually is something which talks *about* things. In fact, it merely consists of a set of XML[3] documents, including XML Schemas and XSL-Transforms. The workflow with meta is the following:

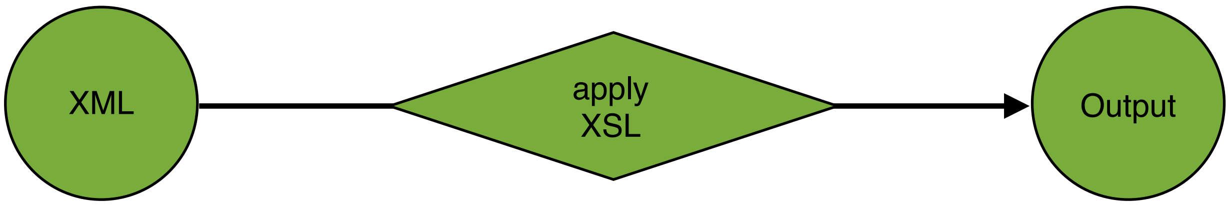
(1) create a description of your object-relational model in XML,

(2) apply some XSL-Transforms,

where “some XSL-Transforms” is a variety of stylesheets which allow you to automatically extract documentation, java-code, or an sql-script from a single XML file you created within the first step. At the time of this writing there are stylesheets available for automatically generating DAOs[4] and DTOs for Java (POJOs[5] that talk to the database instead of you talking to the database), ERM[6] and UML[7] class diagrams using graphViz[8], PostgresQL-Scripts that may be used to set up your database and various XHTML documents providing documentation based on the description given in your initial XML document.

The most complex task is creating the DXOs (DAO/DTO). Note that using META you are not limited to Java or any concrete programming model. You could as well go and grab the templates used for creating the code, extend or replace them to your likings and generate PHP or Haskell[9] from it. You might equally well create JavaBeans, Scala[10] or classes targeting the JPA[11] and JBoss' Hibernate Framework.

So, how does it work? The work flow of weave is as follows:



This is just the general work-flow. The different XSL-Transforms (they can be found in the directory `meta/lib`) currently available are:

erm.xsl

This one is used to generate a DOT file resembling an Entity-Relationship-Model which can than be used for creating various output formats using graphviz (which offers different heuristics for laying out such a Model).

[3] eXtensible Markup Language – http://www.w3.org/TR/xml/

[4] Database Access Object – http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html

[5] Plain Old Java Object – http://www.roseindia.net/ejb/introduction-to-pojo.shtml

[6] Entity-Relationship-Model – http://www.cs.purdue.edu/homes/clifton/cs541/ER.pdf

[7] Unified Modeling Language – http://www.omg.org/gettingstarted/what_is_uml.htm

[8] Graph Visualization Software – http://www.graphviz.org/

[9] Accessing SQL with Haskell – http://passingcuriosity.com/2008/accessing-sql-databases-with-haskell-hsql/

[10] Scala – http://www.scala-lang.org/

[11] Java Persistence API – http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html

uml.xsl

This one is like erm.xsl but it generates a DOT file resembling a UML class diagram, that is: a diagram containing tables which are laid out as boxes and model entities and relationships as classes.

sql.xsl

Creates an SQL-installer script for the PostgreSQL[12] Database-Management-System which may be used to initially setup the database. The resulting script will basically contain CREATE TABLE statements, PRIMARY KEY definitions and some CONSTRAINTs.

xhtml.xsl

Applying this template will result in a nicely formatted XHTML page which resembles the original XML input in a clean and appealing way.

Other special templates

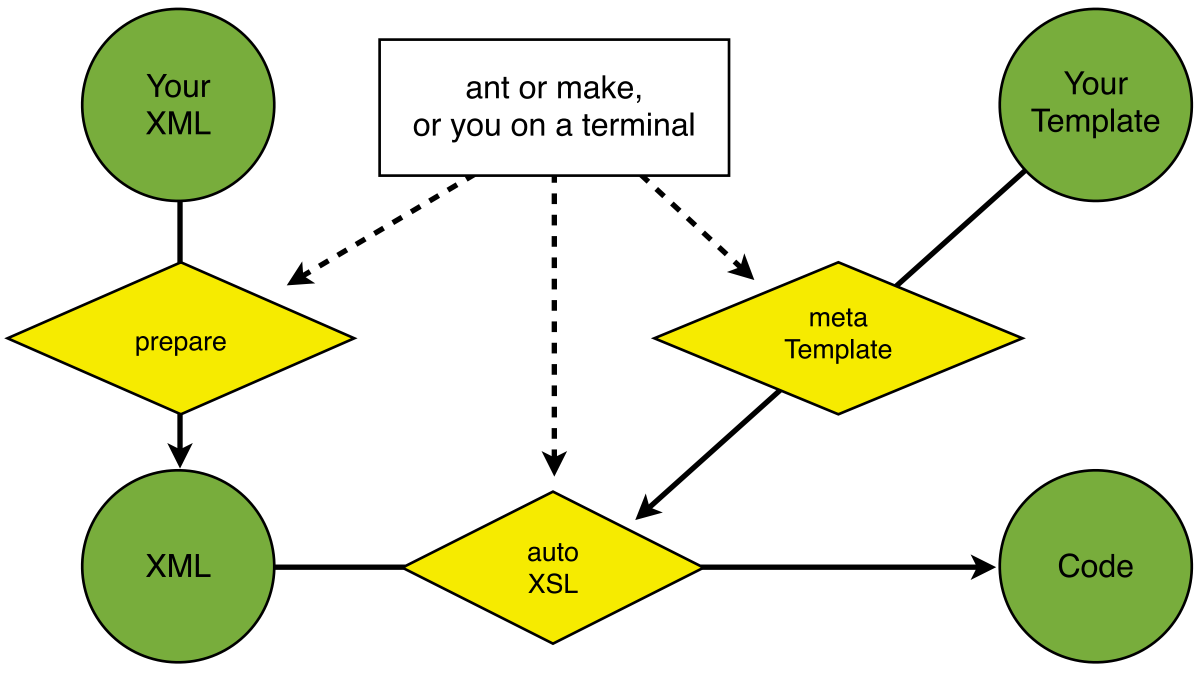
java2.metaTemplate.xsl

The name of this template is a bit misleading, since it is not purely special to generating java code. In fact it is a meta-template, that is, it creates an XSL-T based on some other XML file. The reason for this is rather pragmatic: While XSL-T is a powerful tool its syntax is incredibly ugly to mix it with code from a language which is not based on XML.

prepare.xsl

In fact, before any of the templates mentioned above can be used you have to apply this stylesheet. As the name states, it prepares your XML file into a more complete format which allows for easier processing by the other templates. It will also warn you about inconsistencies, like referencing an entity which does not exist.

Complete work flow (for generating code)



12 PostgreSQL – http://www.postgresql.org/

Using weave + meta

While you’re generally free to create anything you want using META techniques, there are already some useful implementations coming along with META which we’ll discuss here. The default template for code generation is **java2.xsl** which plays nicely along with weave. Using the default setup in conjunction with weave will result in the following Java-classes being generated:

(1) a DAO (typically called `RelationManager`), which acts as a kind of central controller

(2) a DTO for each entity

(3) a DTO for each relationship

The RelationManager will act as some kind of factory.

(more to follow)