AndromdaAndCocoon

Step-by-step tutorial to create a small Cocoon application using AndroMDA

What is AndroMDA

To answer that question, it's important to understand what is MDA first. MDA stands for Model-Driven Architecture. It's a concept developed by the OMG to normalize an high-level approach of development. The idea of MDA is based on the excellent expressivity that UML can have. Of course you can always use UML notation freely to draw some rapid sketches, but if you are willing to respect a few semantic rules and use notation tools like stereotypes, tagged values and a few others, your model can contain so much information that it can be used as a basis for automatic code generation. And if you push the concept a little further and use all the tools of UML notation cleverly, you can even reach the stage of "executable UML", where you just compile your UML model and get a working application. But this is just an exageration. Some people at OMG advocate this but a lot of professional architects claim that it's not UML's job, that it is excellent to describe concepts and architecture of an application, but not its internal code. As always it's a matter of balance.

That said, the great power of MDA is to make it possible to automate the generation of many boilerplate tasks and encourage best practices like Design Patterns use. Have you ever dreamt of getting rid of all those bugs in your configuration files or DAO pattern implementation. If you could just specify those details at a higher level and let some automatic system read this information for you and generate all the stuff accordingly ?

Well, then stop dreaming, because it's reality ! It exists. And the best Open Source solution for that (and when I say the best, I'm speaking of a real work-of-art, an example of Open Source and "best practice" development) : AndroMDA ! No I'm not even a member of the project (maybe one day I'll write my own cartridge, I have a few in mind, and then...) and Chad Brandon didn't pay me to say that. But you'll see, when you'll come to use it, you won't be able to think of developing an application without using it.

Prerequisites

You have to understand the basic concepts behind AndroMDA. Take a look at AndroMDA's website and more precisely at the overview. It's very useful. Then you have to understand UML. Not all UML but at least class diagrams. AndroMDA does use other diagrams, especially in Bpm4Struts cartridge (cartridges are just plugins, modules to generate a particular kind of code... but you should really read their overview :-P) but we will only use Spring and Java cartridge in this tutorial so class diagrams will be enough.

Of course you should have heard of Java... more than that, you should have excellent knowledge of it. And you are expected to know what is Spring and Hibernate. You'll see that one of the advantages of AndroMDA is that you don't have to understand all the details of those technologies, but you should at least understand the basic concepts to be able to understand the generated co You shouldn't wince hearing of :

- Inversion of Control (IoC) or Dependency Injection
- Data Access Object pattern
- Object-Relational Mapping (ORM)

Finally we are gonna use Maven intensively (AndroMDA is mainly a Maven plugin) so you're expected to know of it (and if you don't, believe me it's worth the time to learn) and have a working installation of Maven 1. You should also install the Community Edition of MagicDraw (version 9.5) because, even if AndroMDA can work with any UML design tool capable of exporting **standard** XMI (and I insist on **standard**, because I've tried a few like Borland Together which exports in XMI but it's not standard at all), it's preferable to use the same as the one they use internally. They used to work with Poseidon and it's likely to be very compatible with AndroMDA but I find the interface of MagicDraw so much easier to use. Anyway from now I'll assume you'll work with MagicDraw.

That's it ! We are ready to begin now...

To be followed...