# Tutorial MDArte

Primeiros passos do
*framework* MDArte

Maio 2014

# Contents

*Chapter 1*

# Preparação do Ambiente

Nesta seção detalharemos o processo de preparação do ambiente de desenvolvimento com o AndroMDA, onde serão enumeradas as ferramentas utilizadas e seus respectivos procedimentos de instalação. Ferramentas necessárias:

- Máquina Virtual Java - JDK
- JBoss (versão 4.2.3-GA)
- Maven (versão 1.0.2)
- Magic Draw (versão 9.5)
- Eclipse Indigo (versão 3.7.1)

## 1.1 JDK

É necessário que o JDK esteja instalado no computador. O download pode ser feito em http://java.sun.com/ ou utilizando algum repositório, como mostra abaixo:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java6-installer
```

## 1.2 JBoss

É necessário que o JBoss esteja instalado no computador. O download pode ser feito em http://www.jboss.com/. Após a instalação do JBoss, é necessário configurar a variável `JBOSS_HOME`, a qual deve especificar o caminho de instalação do JBoss. Esse caminho para o JBoss é necessário para a instalação de aplicações (deployment).

Neste tutorial será considerado que o jboss estará instalado na pasta abaixo.

```
/home/<user>/Work/programs/jboss
```

## 1.3 Maven

O Maven[1] é uma ferramenta de automação e gerenciamento de projetos. O download pode ser feito através do endereço http://maven.apache.org/start/download.html e sua instalação consiste em descompactar o arquivo obtido em um diretório local.

A versão compatível é a 1.02. O Maven, durante sua execução, faz acesso a repositórios remotos, de onde poderão ser obtidos diversos artefatos necessários às tarefas de automação. Por exemplo bibliotecas (arquivos `*.jar`) necessárias para compilação e execução de um projeto podem ser automaticamente obtidas.

Para especificar o repositório que o Maven deve acessar é necessário criar um arquivo, chamado `build.properties`, no diretório home do usuário, por exemplo `/home/<usuario>`.

Abaixo temos exemplos do arquivo build.properties para uso com repositórios externos, em repositório externo que possui um proxy para acesso à internet, e com um repositório proxy configurado na rede local.

1. Repositórios remoto quando não é necessário utilizar proxy para acesso à internet:

```
maven.repo.remote=http://www.ibiblio.org/maven,http://team.andromda.org/maven
```

2. Repositórios remoto quando é necessário utilizar proxy para acesso à internet. No exemplo, o IP do proxy é `10.0.2.15`:

```
maven.repo.remote=http://www.ibiblio.org/maven,http://team.andromda.org/maven maven
maven.proxy.port=8080
```

3. Repositório proxy na rede local. O repositório proxy contém as bibliotecas `*.jar` que o Maven poderia requisitar no repositório remoto na Internet. Esse repositório deverá ser utilizado quando não é possível ou não é desejável o acesso ao repositório remoto:

```
maven.repo.remote=http://<host>:<port>
```

Onde `<host>` é o nome da máquina utilizada como proxy e `<port>` é o número da porta do serviço do proxy. Por exemplo: `maven.repo.remote=http://146.164.34.92/repositorio/`

É importante observar que o Maven, de acordo com as tarefas executadas, irá fazer o download dos artefatos necessários e guardá-los em um cache local na estação de trabalho em um diretório chamado `.maven` localizado no diretório home de cada usuário da estação.

Para evitar o acesso a servidores na Internet é possível a instalação de um proxy específico do Maven na rede local. Neste tutorial de configuração, não abordaremos a configuração desse proxy.

Neste tutorial será considerado que o jboss estará instalado na pasta abaixo.

```
/home/<user>/Work/programs/maven
```

## 1.4 Varíaveis de Ambiente

Adicionar no final do arquivo `/home/<usuario>/.bashrc` o seguinte código:

```
if [ -f ~/.bashrc_mdarte ]; then
  . ~/.bashrc_mdarte
fi
```

[1] http://maven.apache.org/

Criar o script onde ficará todas as variáveis do ambiente.

```
touch ~/.bashrc_mdarte
```

Editar o arquivo `/home/<usuario>/.bashrc_mdarte` adicionando os seguintes valores.

```
#MDArte Configurations

export MAVEN_OPTS=-Xmx1024m

export JAVA_HOME=/usr/lib/jvm/java-6-oracle/

if [ -d ~/Work/programs/maven ] ; then
  export MAVEN_HOME=~/Work/programs/maven
fi

if [ -d ~/Work/programs/maven ] ; then
 export PATH=\$PATH:~/Work/programs/maven/bin
fi

if [ -d ~/Work/programs/jboss ] ; then
  export JBOSS_HOME=~/Work/programs/jboss
fi
```

Após será necessário reiniciar a sessão do usuário para essas variáveis estarem no sistema ou utilizar o seguinte comando abaixo.

```
source ~/.bashrc
```

## 1.5 MDArte

O MDArte, na verdade, não é um aplicativo, mas sim um conjunto de bibliotecas de classes. Em nosso processo de desenvolvimento, utilizaremos o MDArte como um plugin do Maven. O Maven, por sua vez, possui um mecanismo próprio para obtenção de plugins. Através de parâmetros na linha de comando podemos especificar ao Maven qual plugin queremos instalar e ele se encarrega de buscar este plugin no(s) repositório(s) para o(s) qual(is) estiver configurado.

No caso do plugin do MDArte, o seguinte comando deve ser executado para a instalação (ao copiar o comando, verificar se foi copiado corretamente, inclusive os hifens):

```
maven plugin:download -DgroupId=andromda -DartifactId=maven-andromdapp-plugin-
```

Após a execução desse comando o Maven terá instalado o plugin do AndroMDA no cache local do usuário e tarefas referentes ao MDArte poderão ser executadas através do Maven.

Eventualmente, dependendo das tarefas executadas, o Maven poderá buscar outros artefatos nos repositórios, contudo isso será feito de forma transparente e automática.

## 1.6 MagicDraw

O download do MagicDraw pode ser feito em http://www.magicdraw.com.

O MagicDraw é uma ferramenta para modelagem em UML e é recomendada para uso com o MDArte devido a seu suporte a diagramas de atividade, utilizados pelo cartucho BPM4Struts. Ainda, para que os modelos sejam corretamente utilizados pelo MDArte eles deverão conter estereótipos específicos, disponíveis através de um profile fornecido com o MDArte, que será mostrado com mais detalhes na seção “Iniciando o projeto no MagicDraw”.

## 1.7 Eclipse

O download do Eclipse pode ser feito em http://www.eclipse.org/.

Durante a geração de um projeto, o MDArte gerará automaticamente os arquivos de configuração `.project` e `.classpath` de um projeto Eclipse. Esses arquivos podem ser usados diretamente para importação do projeto ao Eclipse. O `.classpath` é o arquivo onde será indicado as bibliotecas para o eclipse que serão utilizados pelo projeto. Assim, o eclipse saberá completar as informações automaticamente. Já o `.project` é uma descrição das opções do projeto.

Citation of Einstein paper [1].

*Chapter 2*

# Desenvolvimento de Projetos com o AndroMDA

Nesta seção veremos os passos necessários ao desenvolvimento de projetos com o MDArte, utilizando os cartuchos EJB, Hibernate e BPM4Struts.

## 2.1 Criação de um Novo Projeto e Configuração do ambiente

O plugin do MDArte para o Maven já possui um procedimento parametrizado para criação de projetos, que funciona como um wizard, onde o usuário deve responder a perguntas. Através das respostas fornecidas, o MDArte direcionará a criação da estrutura básica e dos artefatos básicos de configuração de projetos. O procedimento para criação de um novo projeto é:

1. Abra o terminal (command prompt) e vá para o diretório onde se deseja criar o projeto. Na verdade, o projeto será gerado em um subdiretório do diretório escolhido. No Windows, não se pode ter espaços em branco no caminho desse diretório. Exemplo de diretório inválido: C:\Documents and Settings\MDArte.

2. Digite o comando: maven andromdapp:generate

3. Responda as perguntas de acordo com o seu projeto. Abaixo um exemplo com respostas típicas (perguntas em negrito):

   **Please enter your first and last name (i.e. Rodrigo Salvador):**
   MDArte
   **Please enter the name of your J2EE project (i.e. Sistema Academico):**
   Sistema Academico
   **Please enter the id for your J2EE project (i.e. sistemaacademico):**
   sistemaacademico
   **Please enter a version for your project (i.e. 1.0):**
   1.0
   **Please enter the base package name for your J2EE project (i.e. br.mdarte.exemplo.academico):**
   br.mdarte.exemplo.academico

**Would you like to enable security? (enter 'yes' or 'no')?**
yes
**Would you like to use oAuth (enter 'yes' or 'no') ?**
no
**Would you like to use MDArte's default Controle Acesso (enter 'yes' or 'no') ?**
yes
**Would you like to use modules (enter 'yes' or 'no')?**
yes
**Please enter the EJB version number (enter '2' or '3'):**
3
**Please enter the Struts version number (enter '1' or '2'):**
2
**Would you like to enable the JUnit support for general testing? (enter 'yes' or 'no')?**
no
**Please enter the database backend for the persistence layer: (enter 'hypersonic' or 'mysql' or 'oracle' or 'postgres')**
postgres

4. Após receber as respostas, o MDArte criará um subdiretório onde será gerada a estrutura inicial do projeto. A partir desse momento chamaremos esse diretório de <DiretorioProjeto>.
5. Ainda no console, vá para o diretório onde está seu projeto: <DiretorioProjeto>.
6. Digite maven. Isto obrigará o Maven a obter todos os artefatos (por exemplo, bibliotecas) de que o projeto dependerá.

## 2.2 Configuração do Banco

Para se configurar o Banco de Dados é necessário modificar o arquivo project.properties da raiz do projeto, onde se encontram as propriedades que devem ser alteradas. Abaixo estão as propriedades do arquivo de configuração para cada um dos Bancos de Dados

Oracle

- dataSource.driver.jar=${env.JBOSS_HOME}/server/default/lib/hsqldb.jar
- dataSource.driver.class=oracle.jdbc.driver.OracleDriver
- sql.mappings=Oracle9i
- hibernate.db.dialect=org.hibernate.dialect.Oracle9Dialect

SQLServer

- dataSource.driver.jar=${env.JBOSS_HOME}/server/default/lib/hsqldb.jar
- dataSource.driver.class=oracle.jdbc.driver.OracleDriver
- sql.mappings=Oracle9i

- hibernate.db.dialect=org.hibernate.dialect.Oracle9Dialect

Postgres

- dataSource.driver.jar=${env.JBOSS_HOME}/server/default/lib/hsqldb.jar
- dataSource.driver.class=oracle.jdbc.driver.OracleDriver
- sql.mappings=Oracle9i
- hibernate.db.dialect=org.hibernate.dialect.Oracle9Dialect

MySQL

- dataSource.driver.jar=${env.JBOSS_HOME}/server/default/lib/hsqldb.jar
- dataSource.driver.class=oracle.jdbc.driver.OracleDriver
- sql.mappings=Oracle9i
- hibernate.db.dialect=org.hibernate.dialect.Oracle9Dialect

Outro arquivo que deve ser alterado ou criado é o arquivo de configurações do Banco de Dados do JBoss, localizado no diretório JBOSS_HOME/server/default/deploy/, com formação do nome terminando com -ds.xml (ex.: aplicacoes-ds.xml), que deve ter a tag <local-tx-datasource> preenchida de acordo com as informações fornecidas no arquivo <projeto>/project.properties. Exemplo (usando banco Postgres):

```
<local-tx-datasource>
        <jndi-name>sistemaacademicoDS</jndi-name>
        <use-java-context>true</use-java-context>
        <connection-url>jdbc:postgresql://127.0.0.1:5432/sistemaacademico</con
        <driver-class>org.postgresql.Driver</driver-class>
                <user-name>usuario</user-name>
                <password>senha</password>
        <!--<exception-sorter-class-name>org.jboss.resource.adapter.jdbc.vendo
</local-tx-datasource>
```

Repare que no exemplo anterior, o nome do Data Source é sistemaacademicoDS, que deve ser o mesmo nome informado no arquivo project.properties da raiz do projetom ou seja ‘sistemaacademicoDS‘.

Além disso, é necessário alterar o arquivo login-config.xml, localizado no diretório JBOSS_HOME/server/default/co deverá ser modificado, adicionando o seguinte trecho:

```
<!--
SistemaAcademico Policy
-->
<application-policy name="sistemaacademico">
        <authentication>
                <login-module code="org.jboss.security.ClientLoginModule"
                        flag="required">
                        <module-option name="multi-threaded">true</module-opti
                </login-module>
```

```
            <login-module code="accessControl.LoginModuleImpl" flag="required">
                    <module-option name="dsJndiName">java:/controleacessoDS</m
                    <module-option name="unauthenticatedIdentity">guest</module
                    <module-option
                            name="principalClass">accessControl.PrincipalImpl<
                    <module-option name="hashEncoding">hex</module-option>
                    <module-option name="hashAlgorithm">md5</module-option>
                    <module-option name="principalsQuery">
                            select SENHA from OP_C_A where LOGIN=?</module-opti
                    <module-option name="rolesQuery">
                            select OP_PF.PF_OP_C_A_FK, 'Roles'
                            12from OP_C_A, OP_C_A_PF_OP_C_A OP_PF
                            where LOGIN=? AND OP_C_A.ID = OP_PF.OP_C_A_FK
                    </module-option>
            </login-module>
    </authentication>
</application-policy>
```

As informações presentem nesse arquivo permitirão a aplicação do Sistema Acadêmico se conectar a base de dados e validar o usuário no momento de login.

## 2.3 Controle de Acesso

Neste tutorial estaremos utilizando funcionalidades de controle de acesso, porém não é nosso propósito explorar suas funcionalidades. Assim, estaremos utilizando um projeto de controle de acesso desenvolvido pela comunidade do MDArte. O projeto pode ser obtido a partir do repositório Git do MDArte, pelo endereço https://github.com/MDArte/controleacesso.git . Por fim, edite também o arquivo project.properties do ControleAcesso para configurar o tipo de Banco de Dados a ser utilziado, conforme realizado com o projeto SistemaAcademico. Note que a propriedade dataSource.name está definida como ControleAcessoDS. Novamente, precisaremos criar um arquivo de configuração do Banco de Dados, localizado no diretório JBOSS_HOME/server/default/deploy/ . O nome do arquivo deve seguir a mesma formatação mencionada, terminando em -ds.xml (ex.: aplicacoes-ds.xml), podendo estar no mesmo arquivo com as configurações do projeto SistemaAcademico.

Exemplo:

```
<local-tx-datasource>
        <jndi-name>controleAcessoDS</jndi-name>
        <use-java-context>true</use-java-context>
        <connection-url>jdbc:postgresql://127.0.0.1:5432/controleacesso</connection
        <driver-class>org.postgresql.Driver</driver-class>
                <user-name>usuario</user-name>
                <password>senha</password>
        <!--<exception-sorter-class-name>org.jboss.resource.adapter.jdbc.vendor.Or
</local-tx-datasource>
```

Note que no exemplo anterior o ControleAcesso estará utilizando a mesma base de dados do projeto SistemaAcademico, definida pela tag <connection-url>. Agora, execute os seguinte comandos, na raiz do projeto ControleAcesso, para gerar, compilar e copiar os pacotes para o diretório JBOSS_HOME/server/default/deploy/:

```
maven mda -Dprojeto=ca-core
cd common
maven jar:install deploy
cd ../core/cd
maven jar:install deploy
```

# Bibliography

[1] Albert Einstein. Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]. *Annalen der Physik*, 322(10):891–921, 1905.