DEV3 - Major Feature # 10608

Status:	New	**Priority:**	Should have
Author:	Martin Eisengardt	**Category:**	
Created:	2010-11-03	**Assigned To:**	
Updated:	2010-11-03	**Due date:**	
Subject:	FLOW3 module development (FLOW3 module projects)		

Description

- FLOW3 Project

Currently DEV3 is focused on developing a whole web application. There is no clean support or guideline how to develop a single module or library. Developing applications should be devided into several stages and targets. The targets may be:
A) An independent library module
B) An extension containing it's own routings and a web interface
C) A web application.
All of the target may need their own directorty layout. While a web application may need a complete flow3 installation and directory layout the library module and the extension may only need a directorty layout during development time but will be released without so that they can be installed in another project.

Supporting development should devide the following stages:
1) Development of a module (let the module with local directory layout become part of the svn/git)
2) Dependency lookup (lookup the framework and other modules, do not store them as symbolic links)
3) Development installation and development needs
4) releasing

First of all flow3 should support the following directory layout:

```
/wsroot/libproject1
    Configuration....
    Data....
    Packages....
        MyLib....
    Web....
/wsroot/libproject2
    Configuration....
    Data....
    Packages....
        MyLib....
    Web....
/wsroot/webproject
    Configuration....
    Data....
    Packages....
        MyWebApp....
    Web....
/wsroot/fwkproject
    Packages/Framework....
```

The needed patch to FLOW3 is described here: http://forge.typo3.org/issues/10524

The benefit from this directory layout:
- The IDE (f.e. eclipse PDT) does only have the framework classes once and indexing and type lookup would be faster while developing multiple modules.
- Dependencies are not stored in the file system (symbolic links) or the versioning system (git submodules/svn:externals)
- Development local configuration can be stored in the library project; the release will only contain the package directory.

Extension to DEV3:
- A create FLOW3 project wizard that prepares the directory structure

- Support for a common repository system.

The dependency lookup requires some kind of repository system. DEV3 wshould not re-invent the wheel at this point. IMHO maven should be a good starting point. Whenever a module requires dependencies they are not longer stored in the project itself. But they are only stored in the Package.xml file or for maven in the pom.xml file. This xml file describes the module and the dependencies. Maven should be flexible enough that it will support flow3 module and website projects (this point should be cleared out).

Eclipse nicely integrates maven for java already. We will expect the following use cases:
1) A developer starts with a clean workspace and creates a flow3 project
2) A developer starts with a clean workspace and checkouts/clones an existing project
3) A developer has an existing workspace and clones/checkouts an module that is a dependency of another already existing project. For example the framework will first be downloaded from a repository and later on the developer decides to clone the frameworks git repository.

The java integration does the following. Whenever there is a new project the dependencies will be summed up. Foreach single dependency it will do the following lookup:
A) Is the dependency already part of a project in the workspace?
B) Is the dependency present in the local repository (stored somewhere near ~/.m2/repository)? If it is found maven will do a check if this is the newest version.
C) Is the dependency present in configured remote repositories? If it is found the dependency will be downloaded and installed into the local repository.

Having this behaviour would be really nice. There are more benefits later on: FLOW3 and TYPO3 have a clean repository system and know how and where to download the files. But what does it mean for our projects?

Each project should represent a single module. There will be a pom.xml file that defines the module. Eclipse should know how to extract the relevant information from pom.xml and build the files: .htaccess/ Package.xml and others on the fly. The pom.xml will be the leading file. A nice editor will support developers to find the correct modules and versions to build their website.

There must be a parent pom.xml that covers php and flow3 development. However php development is really different from java development...
Each single process declared by maven should be covered by the parent pom.xml file. So that developers and even eclipse are able to support the development stages. Maven declares the following stages or command:
1) Create project
2) Install project
3) Build (See below)
4) Testing
5) Release
6) Install
7) Run

- Additional features

I am currently developing small extensions that provide support for repository/ entity layouts. A small xml file will describe entites and repositories in the module. A generator will be able to generate the php files. Later on there may be an uml editor that is able to manipulate the xml files. Due to the fact I am currently experimenting with FLOW3 I do not use any uml format for the xml file. An eclipse integration should be able to work with open standards.

Eclipse should be able to create the model as soon as the file changes. In other words: As soon as the build stage within maven is reached. The integration could be nicely done by simple providing an alternative pom.xml file that installs an entity/repository builder.

History