

Eclipse plugin refactored for extensibility

During 2.x development, the Maven-eclipse-plugin became more and more complex with addition of support for many variants of Eclipse (RAD, MyEclipse, ...) and support for various users request. The 2.5.x series introduced nice integration with eclipse with detection of projects present in workspace, with the cost of yet more code. In the same time, many users are still not happy with the way this plugin configures the workspace, with request to support many tools configuration, that exist both as maven and eclipse plugins.

This proposal goal is to define a new architecture for the maven-eclipse-plugin 3.x , based on a limited and extensible core, completed by loosely coupled components to analyse the workspace and setup eclipse to match user environment. Custom extension can be added to the plugin simply by adding a dependency to the plugin configuration. This will encourage other plugin developpers to create maven-eclipse-plugin extension that can be used with no delay, and could be later included in the plugin itself if supported by the community.

Please note this proposal is early draft

new extensible architecture

The maven-eclipse-plugin will define a public contributor API, as a dedicated artifact that any extension will depend on. This API consist of :

An interface for contributors that will analyse the workspace and detect some features to be available

The contributor will receive the workspace location path and can extract from there any relevant information (eclipse variant, installed features, ...)

examples :

- detect WTP version
- detect m2eclipse plugin
- detect RAD6 / RAD7 / myEclipse
- ...

An interface for contributors that will analyse the maven project and extract some configuration

The contributor will receive the MavenProject model object to be analyzed ant extract maven configuration that may be translated to eclipse

examples :

- maven compiler plugin target (for JRE)
- checkstyle configuration
- servlet API (from dependencies)

An interface for contributors that will create or extend the set of configuration files

The contributor will get invoked after workspace and project analysis, and can then add / contribute configuration objets in the context

A context (Map-style) for contributors to share datas

The context will contain informations extracted from the environment (eclipse workspace + maven project) and

configuration objects

A base abstract class for any configuration file to be generated

At contributor-API level, this is not related to files being written but only to abstract configuration objects. Only the core plugin will convert those classes to configuration files. The plugin offers a default set of Configuration objects for standard eclipse configuration files (.classpath, .project, .settings ...)

A contributor can then detect if the feature it supports is enabled/supported (and share the info with other contributors), contribute configuration objects and define new ones.

Pre-packaged contributors

The maven-eclipse-plugin itself will package many contributors for the features supported by the 2.5.x series. The code from this branch will be refactored to the new architecture.

Contributors configuration

to be investigated

one option is to declare all contributors as nested <extension> elements in the plugin configuration, but this may create huge/ugly configuration blocs.

another option is for the plugin to share a global Tree-style configuration object, with some default entries ("workspace" for the eclipse workspace location) and some contributor dedicated ones ("spring.context.includes" = ***/applicationContext.xml**) that will be used only by the contributors.