Incremental Builds

Status	WIP
Version	
Issue(s)	~~MCOMPILER-21~~ CLOSED
Sources	trunk
Developer(s)	Mark Struberg

Currently (3.0.4) Apache Maven doesn't support incremental builds very well. Because of that fact, most people use `mvn clean compile` instead of `mvn compile` for example. This is pretty time consuming and can be improved a lot.

The goal is to make `mvn compile`, `mvn verify`, `mvn install`, ... (all stuff without `clean`) usable for almost all situations.

In general it's better we unnecessarily force a bit more work than to not detect a change as in the later case we would end up with broken artifacts and people will use the `clean` goal again.

Rational - aka what is broken?

check out the sample project

```
git clone git://github.com/struberg/maventest.git
```

First please set the maven-compiler-plugin version back to 2.5 (the latest release). Then build the whole project with

```
mvn clean install
```

Now change the code of BeanA and rename `getI()` to `getI_doesntexistanymore()`. This means that BeanA2.java as well as BeanB.java should fail to compile!

But if we try to build the project with

```
mvn install
```

without any clean lifecycle, then we see 2 bugs

1. the maven build still succeeds to compile the project
2. maven even generates a jar which contains broken classes
3. moduleB does not get recompiled and is thus broken as well.

And this is just the tip of the iceberg.

Solution

In maven this needs 2 parts to get tweaked.

A.) Incremental Module Builds

The following parts should be implemented in the maven-core reactor code (or what remained from it). If any of those tests indicate a change then we force a 'clean' on the module and on all depending downstream modules.

1. If an artifact in the maven repository changed. The change of an artifact might get detected by it's md5.

1. If an activated Profile or an evaluated property changed. We might store away the hash code (md5) of the evaluated profiles and properties in a file in target/. We do **not** store the evalutated properties in a file due to security reasons (passwords, etc)

B.) Plugin support for Incremental Builds

Each plugin need to check on it's own whether it should perform it's tasks or not. Not every plugin needs the full dependency graph. And other plugins (e.g. maven-ear-plugin) even have 'manual' dependencies which are not reflected in the dependency graph. Thus all plugins need to check for themselfs whether they need to do something or not. The first plugin which kicks in detecting a change will create a result. And this result might trigger work in another plugin/phase.

Strategies for change detection

A plugin has a few ways to detect that it needs to do some work

1. if the input file is of same date or newer than the output file
2. if the input file is is newer than the timestamp when the build got started
3. hash Codes. A plugin might store the md5 of it's dependencies or input files to detect a change.
4. additional pid or status files. E.g. the maven-shade-plugin could store the work state + md5 of the generated result in such a file.

It's suggested that all those additional information will get stored in `./target/maven.status/${plugin-name}.status`