Terse POM Syntax - Design Discussion

Background

Recently, I've been noticing that some people (myself included) are having a hard time with the verbosity of the POM syntax. It seems we could do a lot more with the use of XML attributes and some decent ID validators behind the scenes. Some of this is as simple as reworking the Modello model to use attributes, others may require revisions to the underlying Plexus-based configuration mechanism. At any rate, I want to open the discussion for a new syntax here.

Current syntax sample

Below is a sample of what a just-barely-customized POM might look like. Note that it only has three dependencies. Also note the extra lines occupied by simple list delimiters, and the 'implementation' attribute used to specify a String list element (seems reasonable to assume String can be a default list item type)...

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.myco.somegroup</groupId>
    <artifactId>my-parent</artifactId>
    <version>1.0-SNAPSHOT</version>
  </parent>
  <artifactId>my-artifact</artifactId>
  <version>1.0-SNAPSHOT</version>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>servletapi</groupId>
      <artifactId>servletapi</artifactId>
      <version>2.3</version>
      <scope>provided</scope>
    </dependency>
    <dependency>
      <groupId>plexus</groupId>
      <artifactId>plexus-utils</artifactId>
      <version>1.0.2</version>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>

        <configuration>
          <something>some value</something>
        </configuration>

        <executions>
```

```
        <execution>
          <id>first</id>

          <configuration>
            <excludes>
              <exclude implementation="java.lang.String">**/BadTest.java</exclude>
            </excludes>
          </configuration>

          <goals>
            <goal>test</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
  </plugins>
```

```
  </build>
</project>
```

First proposed alternative

I think we can make use of two things that will dramatically reduce the verbosity of the above POM: XML attributes, and good ID validation. The first will make better use of 'container' elements, and the second has the potential to define and enforce a sub-syntax for IDs, like dependency IDs for example. Consider the following revision of the above POM:

```
<project modelVersion="4.0.0" id=":my-artifact:1.0-SNAPSHOT">

  <parent groupId="org.myco.somegroup" artifactId="my-parent" version="1.0"/>

  <dependency id="junit:junit:3.8.1" scope="test"/>
  <dependency id="servletapi:servletapi:2.3" scope="provided"/>
  <dependency id="plexus:plexus-utils:1.0.2"/>

  <build>
    <plugin id="org.apache.maven.plugins:maven-surefire-plugin:">
      <configuration>
        <something>some value</something>
      </configuration>

      <execution id="first">
        <configuration>
          <excludes type="java.lang.String">
            <exclude>**/BadTest.java</exclude>
          </excludes>
        </configuration>

        <goal name="test"/>
      </execution>
    </plugin>
  </build>
</project>
```

Here, IDs are using a terse format that looks like groupId:artifactId:version, where missing information would be expressed through blank values. For example, the project's groupId is inherited from the parent, so its ID looks like ":my-artifact:1.0-SNAPSHOT". Also, a plugin section that doesn't want to tie the build to a particular plugin version would look like "org.apache.maven.plugins:maven-surefire-plugin:".

Also, XML elements used solely to delimit lists of other elements are eliminated. We can depend on XSD validation for the positioning and grouping of list elements. For things like managed dependencies and managed plugins, these should probably be grouped under a <managedInformation> section or something. But list delimiter elements are useless, since the presence of multiple of the same element implies a list.

Finally, the 'implementation' attribute on all configuration list elements has been moved to the list delimiter for that configuration - the only place where I see it making sense to have a list delimiter XML element. This removes the need to replicate this attribute again and again, and adds the benefit of templatizing the list handling for configurations. In other words, it has semantics similar to:

```
java.util.List<String>
```

This is just a first stab at revising the POM syntax...possibly other options could include a maven-handled namspace convention, where plugin configuration can be directly embedded in the build section to improve coherence.