DataModel

The Modello Data Model.

Staging site giving full modello's model (under construction) can be found here: http://dcabasson.developpez.com/maven/modello-metamodel.html

Here's an example of a modello data model.

```
<model>
  <id>myproject</id>
  <name>MyProject</name>
  <description>My Project's Data
Model</description>
  <defaults>
    <default>
      <key>package</key>

<value>org.codehaus.myproject.model</value>
    </default>
  </defaults>
  <classes>
    <class rootElement="true"
xml.tagName="myproject">
      <name>Model</name>
      <version>3.0.0+</version>
      <fields>
        <field>
          <name>myfield</name>
          <version>1.0.0</version>
          <description>The example of
field.</description>
          <type>String</type>
        </field>
      </fields>
    </class>
  </classes>
</model>
```

Top level <model>

The top level <model> must contain an <id> <name> and at least 1 <classes><class> element.

The optional <defaults> tag is commonly used to define properties that the various modello generators use.

The <classes> section.

You can have 1 or more <class> elements within this section.

Each <class> represents a container for a set of <field>s.

Every <field> is a component in the container.

Notes on the <field> section

XML Generator Notes about <field>

1. By default each `<field>` starts out life as an `<element>` with text content.
2. A `<field>` can be set to be an attribute on the `<class>` element by using the `<field xml.attribute="true">` syntax.
3. `field/name` becomes the name of the element (or attribute key).
    a. `field#xml.tagName` can be used to change the name of the element (or attribute key)
4. `field/version` is unused.
5. `field/description` is unused.
6. `field/type` is one of the following
    - boolean
    - char
    - double
    - int
    - long
    - short
    - date
    - String
    - Boolean
    - DOM (currently only supported by xpp3 generators)
    - Another `<class>` **this can only be achieved through an association tag**
7. `field/association` allows linking to other classes of the model
    a. generated readers and writers do not enforce the multiplicity rules (yet)
    b. `field/association` must have at a minimum the <type> element defined.
    c. `field/association/multiplicity` dictates the multiplicity of a field (1 or `*` : 0 to many so far)
    d. `field/association/type` only refers to other <class> id's, no java primitives allowed here.
        i. `field/association/type` will be used to determine the name of the nested elements.
        ii. singular form of the field tagname will be used for the nested element name.
            - `s/(.*)ies$/\1y/g` - `"Properies"` becomes `"Property"`
            - `s/(.*(ch)?)es$/\1/g` - `"Branches"` becomes `"Branch"`
            - `s/(.*)s$/\1/g` - `"Reports"` becomes `"Report"`
        iii. `field/association#xml.associationTagName` can be used to specify an overridden nested element name.
        iv. To learn more about tagName resolution, see [TagName resolution]
    e. `field#xml.listStyle` defaults to "wrapped".

i. Example of `field#xml.listStyle="wrapped"`

```
<!-- Model -->
<class>
  <name>example</name>
  <fields>
    <field xml.listStyle="wrapped">
      <name>components</name>
      <version>4.0.0</version>
      <association>
        <type>Component</type>
        <multiplicity>*</multiplicity>
      </association>
    </field>
  </fields>
</class>
<class>
  <name>Component</name>
  <version>4.0.0</version>
  <fields>
    <field>
      <name>name</name>
      <type>String</type>
    </field>
  </fields>
</class>
```

```
<!-- xml -->
<example>
  <components>
    <component>
      <name>foo</name>
    </component>
    <component>
      <name>bar</name>
    </component>
  </components>
</example>
```

ii. Example of `field#xml.listStyle="flat"`

```
<!-- Model -->
<class>
  <name>example</name>
  <fields>
    <field xml.listStyle="flat">
      <name>components</name>
      <version>4.0.0</version>
      <association>
        <type>Component</type>
        <multiplicity>*</multiplicity>
      </association>
    </field>
  </fields>
</class>
<class>
  <name>Component</name>
  <version>4.0.0</version>
  <fields>
    <field>
      <name>name</name>
      <type>String</type>
    </field>
  </fields>
</class>
```

```
<!-- xml -->
<example>
  <component>
    <name>foo</name>
  </component>
  <component>
    <name>bar</name>
  </component>
</example>
```