OPEN SOURCE APPLICATIONS FOUNDATION

XML Format for Chandler's Data Model Schemas

Goals

- Capture chandler data model in one canonical source, checked into CVS
- Transform to the XML format the Chandler repository understands, or use this format directly for loading schema items into the repository.
- Transform to RDF
- Generate documentation about data model

Example

```
<DomainSchema id="book:BookSchema"
              xmlns="http://osafoundation.org/2003/08/CoreSchema"
              xmlns:core="http://osafoundation.org/2003/08/CoreSchema"

xmlns:util="http://osafoundation.org/2003/08/UtilitySchema"

xmlns:people="http://osafoundation.org/2003/08/PeopleSchema"
              xmlns:book="http://osafoundation.org/2003/08/BookSchema"
              xml:base="http://osafoundation.org/2003/08/BookSchema">

  <label>Book Schema</label>
  <comment>Kinds, AttributeDefinitions, and Items for a book
parcel.</comment>

  <!-- @@@ may be temporary, once we figure out parcel loading -->
  <!--     right now, very handy for the xslt transform to the other
format -->
  <containmentPath prefix="book" path="OSAF/book"/>
  <containmentPath prefix="people" path="OSAF/people"/>
  <containmentPath prefix="core" path="OSAF/people"/>
  <containmentPath prefix="util" path="OSAF/util"/>

  <version>0.2</version>
  <dependsOn itemref="people:PeopleSchema"/>
  <dependsOn itemref="core:CoreSchema"/>
  <dependsOn itemref="util:UtilitySchema"/>

  <AttributeDefinition id="book:isbn">
    <label>ISBN</label>
    <cardinality>single</cardinality>
    <type itemref="String"/>
    <comment>The ISBN number associated with a book</comment>
    <required/>
```

```
</AttributeDefinition>

<AttributeDefinition id="book:title">
  <label>Book Title</label>
  <cardinality>single</cardinality>
  <type itemref="String"/>
  <comment>The title of a book.</comment>
  <required/>
</AttributeDefinition>

<AttributeDefinition id="book:author">
  <label>Book Author</label>
  <cardinality>single</cardinality>
  <type itemref="book:Author"/>
  <comment>The author of a book.</comment>
  <inverseAttribute itemref="authored"/>
</AttributeDefinition>

<AttributeDefinition id="book:authored">
  <label>Authored</label>
  <cardinality>set</cardinality>
  <type itemref="book:Book"/>
  <comment>Books a given person has authored.</comment>
  <inverseAttribute itemref="book:author"/>
</AttributeDefinition>

<Kind id="book:Book">
  <label>Book</label>
  <comment>A kind representing a book</comment>
  <attribute itemref="book:isbn"/>
  <attribute itemref="book:title"/>
  <attribute itemref="book:author"/>
</Kind>

<Kind id="book:Author">
  <label>Book Author</label>
  <comment>A kind representing the author of a book.</comment>
  <superKind itemref="people:Person"/>
  <attribute itemref="book:authored"/>
  <displayAttribute itemref="people:name"/>
</Kind>

<Alias id="book:LibraryInfo">
  <label>Library Information</label>
  <comment>An alias for any book related item.</comment>
  <aliasFor itemref="book:Book"/>
  <aliasFor itemref="book:Author"/>
</Alias>

<book:Author id="book:YM">
  <people:name>Yann Martel</people:name>
</book:Author>

<book:Book id="book:LP">
  <book:title>Life of Pi</book:title>
  <book:author itemref="YM"/>
  <book:isbn>0-15-602732-1</book:isbn>
```

```
  </book:Book>

</DomainSchema>
```

Elements

- **<DomainSchema>**: The root element for the document, the schema being defined.
    - **id** xml attribute: Uniquely identifies this schema. Not the UUID. Used to define the URI for this Item, also used in the repository containment path.
    - **<label>**: String that can be used as a display name for this Schema.
    - **<comment>**: Comment about this item, for documentation.
    - **<example>**: Example for this item, for documentation.
    - **<issue>**: Open issue related to this item, for documentation.
    - **<containmentPath>**: Defines the mapping between a namespace prefix and the containment path of the data parcel, used to generate itemrefs. Yes, we should be able to derive this somehow, and not need to list the paths. Useful for practical reasons in the short term, may go away.
        - **prefix** xml attribute: The namespace prefix used for this containment path. The prefix is used by itemrefs.
        - **path** xml attribute: The containment path.
    - **<version>**: Capture version information about the schema. Semantics are still fuzzy, a placeholder.
    - **<dependsOn>**: Include a tag for every schema this schema depends on.
        - **itemref** xml attribute: gives a URI to reference the DomainSchema

- **<Kind>**: Top level element, will generate a Kind item in the repository.
    - **id** xml attribute: Uniquely identifies this Item among all other Kinds, Types and Aliases in this schema. Not the UUID. Used to define the URI for this Item, also used in the repository containment path.
    - **<label>**: String that can be used as a display name for this Kind.
    - **<comment>**: Comment about this item, for documentation.
    - **<example>**: Example for this item, for documentation.
    - **<issue>**: Open issue related to this item, for documentation.
    - **<superKind>**: Optional empty element if this Kind has a superKind.
        - **itemref** xml attribute: gives a URI to reference the superKind.
    - **<attribute>**: One empty element for each implied attribute on this Kind.
        - **itemref** xml attribute: gives a URI to reference the AttributeDefinition.
    - **<displayAttribute>**: Optional empty element if a particular AttributeDefinition is to be used as the display value for an instance.
        - **itemref** xml attribute: gives a URI to reference the AttributeDefinition.
    - **<equivalentKind>**: Include an empty tag for each equivalent Kind.
        - **itemref** xml attribute: gives a URI to reference the Kind
    - **<hidden>**: Optional empty element if this Kind is hidden.
    - **<abstract>**: Optional empty element if this Kind is abstract.

- **<AttributeDefinition>**: Top level element, will generate an AttributeDefinition item in the repository. The AttributeDefinition will be globally accessible.
    - **id** xml attribute: Uniquely identifies this Item among all other AttributeDefinitions. Not the UUID. Used to define the URI for this Item, also used in the repository containment path.
    - **<label>**: String that can be used as a display name for this AttributeDefinition.
    - **<comment>**: Comment about this item, for documentation.
    - **<example>**: Example for this item, for documentation.
    - **<issue>**: Open issue related to this item, for documentation.
    - **<type>**: One empty element for each Type, Kind or Alias allowed for values of this AttributeDefinition.
        - **itemref** xml attribute: gives a URI to reference the Type, Kind or Alias
    - **<cardinality>**: Gives the cardinality of the attribute, one of these:
        - single: only one value
        - set: collection, no duplicates
        - list: ordered collection
    - **<superAttribute>**: Optional empty element if this attribute has a superAttribute.
        - **itemref** xml attribute gives a URI to reference the superAttribute.
    - **<default>**: Optional default value for this AttributeDefinition.
    - **<required>**: Include an empty tag if this is a required attribute.
    - **<derivation>**: String describing the derivation rule. Likely to change once we figure out how derivation rules will work.
    - **<hidden>**: Include an empty tag if this is a hidden attribute.
    - **<equivalentAttribute>**: Include an empty tag for each equivalent AttributeDefinition
        - **itemref** xml attribute: gives a URI to reference the AttributeDefinition
    - **<inverseAttribute>**: Include an empty tag for the AttributeDefinition to be used as the inverse. Only applies to bidirectional references.
        - **itemref** xml attribute: gives a URI to reference the AttributeDefinition
    - **<unidirectional>**: We might want to make this attribute a unidirectional reference.
    - **<relationshipType>**: Only applies to attributes that are references to items. Describes the type of relationship that this item has to the item refered to by this attribute. The type of relationship can be one of these:
        - peer:
            - This is the default value, if the attribute has no relationshipType element.
            - The 'inverseAttribute' must have relationshipType 'peer'.
            - The delete policy will be set to 'strong reference' (aka 'cascade:count').
            - The copy policy will be set to 'leave null if the inverseAttribute is cardinality single, otherwise do a shallow copy'.

        - parent: The item that uses this attribute is part of a hierarchy of items. This attribute points to an item which is the parent of this item.
            - The 'inverseAttribute' must have relationshipType 'children'.
            - The delete policy will be set to 'strong reference' (aka 'cascade:count').
            - The copy policy will be set to 'shallow'.
            - In RDF/OWL, the attribute will be marked as 'transitive'.
        - children: The item that uses this attribute is part of a hierarchy of items. This attribute points to the items which are the children of this item.
            - The 'inverseAttribute' must have relationshipType 'parent'.
            - The delete policy will be set to 'strong reference' (aka 'cascade:count').
            - The copy policy will be set to 'leave null if the inverseAttribute is cardinality single, otherwise do a shallow copy'.
            - In RDF/OWL, the attribute will be marked as 'transitive'.
        - owns:
            - The 'inverseAttribute' must have relationshipType 'ownedBy'.
            - The delete policy will be set to 'cascade'.
            - The copy policy will be set to 'deep'.
        - ownedBy:
            - The 'inverseAttribute' must have relationshipType 'owns'.
            - The 'cardinality' must be 'single'.
            - The delete policy will be set to 'weak reference' (aka 'remove').
            - The copy policy will be set to 'leave null if the inverseAttribute is cardinality single, otherwise do a shallow copy'.
        - equivalent:
            - The 'inverseAttribute' must be this attribute.
            - In RDF/OWL, the attribute will be marked as 'transitive'.
            - In RDF/OWL, the attribute will be marked as 'symmetric'.

- **<Alias>**: Top level element, will generate an Alias item in the repository. The Alias will be globally accessible.
    - **id** xml attribute: Uniquely identifies this item among all other kinds, types and aliases in this schema. Not the UUID. Used in the repository containment path. Used to reference this item from other schema declarations.
    - **<label>**: String that can be used as a display name for this Alias.
    - **<comment>**: Comment about this item, for documentation.
    - **<example>**: Example for this item, for documentation.
    - **<issue>**: Open issue related to this item, for documentation.
    - **<aliasFor>**: One empty element for each Type or Kind this alias can represent.

        - **itemref** xml attribute: gives a URI to reference a Type or Kind item.

- **<Type>**: Top level element, will generate a Type item in the repository. The Type will be globally accessible.
    - **id** xml attribute: Uniquely identifies this item among all other kinds, types and aliases in this schema. Not the UUID. Used in the repository containment path. Used to reference this item from other schema declarations.
    - **<label>**: String that can be used as a display name
    - **<comment>**: Comment about this item, for documentation.
    - **<example>**: Example for this item, for documentation.
    - **<issue>**: Open issue related to this item, for documentation.
    - **<superType>**: Include an empty element if this Type has a superType.
        - **itemref** xml attribute: gives a URI to reference a Type item.
    - **<field>**: Empty attribute for each field in the type
        - **name** xml attribute: give the name of the field
        - **type** xml attribute: give the type of the field

Schemas Describing XML Format

- DataModelSchema-DTD (@@@ mildly bogus, need to fix to match current thinking)
- DataModelSchema-XMLSchema?
- DataModelSchema-RelaxNG?

XSLT Transforms

- DataModelSchemaRepositoryTransform (now just an exercise, we're going to parse the format directly)
- DataModelSchemaXHTMLTransform Micah Dubinko's XSLT
- @@@ To RDF/XML.

Notes/Assumptions/Decisions

- Every item has a URI. The URI is a separate concept from the uuid.
- Wherever the xml attribute "itemref" is used, the xml attribute "uuid" could be used instead. Instead of having a URI value, "uuid" xml attribute would have a uuid value identifying the item.

Open Issues/To Do

- Terminology.
    - It would be nice to have one word for Kind/Type/Alias. Current proposal is Taxon.
    - DomainSchema vs schema vs data model schema, etc.

- Does every CalendarEvent Kind item have the same URI? Across different repositories? Across replicated repositories? Is the repository an item is located in part of its URI?

    - One proposal: URIs are unique for items, just like uuids.

- We're still a bit fuzzy on some of the problems we know we need to solve with the data model. Some elements are subject to change once we're less fuzzy:
    - version
    - derivationRules
    - unidirectional
    - relationshipType

- We might want an "include" feature, either our own tag or use XMLInclude. At the moment, we're able to avoid it in our PIM schema and core schemas.

- We'll want various attributes to have defaults. For example, the default of cardinality might be 'single'. One goal is for the common case to be short and sweet. This document doesn't specify the defaults yet, we'll add them as we gain experience.

- We'll want to clean up the containmentPath/dependsOn/namespace tags. We want the containment path for each DomainSchema specified in at most one document. Perhaps it will be generated based on the parcel directory, or some other reasonable solution.

Discussion

- BrianDouglasSkinner's comments about relationship types: http://lists.osafoundation.org/pipermail/dev/2003-August/000756.html
- Feedback from MicahDubinko: http://lists.osafoundation.org/pipermail/dev/2003-August/000760.html

-- KatieCappsParlante - 13 Aug 2003

XSL Transform from Data Model XML format to Repository XML format

Transform

```
<?xml version="1.0"?>

<xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

xmlns:schema="http://osafoundation.org/2003/08/schema_definition">

  <xsl:output method="xml" indent="yes"/>

  <xsl:key name="prefix"
match="schema:DomainSchema/schema:containmentPath" use="@prefix"/>

  <xsl:template match="/">
    <items>
      <xsl:apply-templates
select="schema:DomainSchema/schema:AttributeDefinition"/>
      <xsl:apply-templates select="schema:DomainSchema/schema:Kind"/>
    </items>
  </xsl:template>

  <xsl:template match="schema:AttributeDefinition">
    <item>
      <name><xsl:value-of select="substring-after(./@id, ':')"/></name>
      <kind type="path">//Schema/Model/AttrDef</kind>
      <ref name="Type" type="path"><xsl:apply-templates
select="schema:type/@itemref"/></ref>
      <attribute name="Cardinality"><xsl:value-of
select="schema:cardinality"/></attribute>
    </item>
  </xsl:template>

  <xsl:template match="schema:Kind">
    <item>
      <name><xsl:value-of select="substring-after(./@id, ':')"/></name>
      <kind type="path">//Schema/Model/Kind</kind>
      <ref name="AttrDefs">
        <xsl:apply-templates select="schema:attribute"/>
      </ref>
      <ref name="SuperKind">
        <ref type="path">//Schema/Model/Item</ref>
      </ref>
      <attribute name="Class"
type="model.schema.Types.Class"><xsl:value-of
select="schema:pythonClass"/></attribute>
    </item>
  </xsl:template>

  <xsl:template match="schema:attribute">
```

```
    <ref type="path">
      <xsl:attribute name="name"><xsl:value-of select="substring-
after(./@itemref, ':')"/></xsl:attribute>
      <xsl:apply-templates select="./@itemref"/>
    </ref>
  </xsl:template>

  <xsl:template match="@itemref">
    <xsl:value-of select="key('prefix', substring-before(.,
':'))"/><xsl:value-of select="substring-after(., ':')"/>
  </xsl:template>

</xsl:stylesheet>
```

Calendar Schema 'canonical' XML (input to the transform)

```
<DomainSchema id="cal:CalendarSchema"

xmlns="http://osafoundation.org/2003/08/schema_definition">

  <label>Calendar Schema</label>
  <comment></comment>
  <version></version>

  <containmentPath
prefix="types">//Schema/Model/AttrDef/</containmentPath>
  <containmentPath
prefix="cal">//Schema/OSAF/CalendarSchema/</containmentPath>

  <AttributeDefinition id="cal:startTime">
    <label>Start Time</label>
    <cardinality>single</cardinality>
    <type itemref="types:DateTime"/>
  </AttributeDefinition>

  <AttributeDefinition id="cal:endTime">
    <label>End Time</label>
    <cardinality>single</cardinality>
    <type itemref="types:DateTime"/>
  </AttributeDefinition>

  <AttributeDefinition id="cal:headline">
    <label>Headline</label>
    <cardinality>single</cardinality>
    <type itemref="types:String"/>
  </AttributeDefinition>

  <AttributeDefinition id="cal:location">
    <label>Location</label>
    <cardinality>single</cardinality>
    <type itemref="cal:Location"/>
  </AttributeDefinition>

  <AttributeDefinition id="cal:calendar">
    <label></label>
```

```
    <cardinality>set</cardinality>
    <type itemref="cal:Calendar"/>
  </AttributeDefinition>

  <AttributeDefinition id="cal:name">
    <label>Name</label>
    <cardinality>single</cardinality>
    <type itemref="String"/>
  </AttributeDefinition>

  <Kind id="cal:CalendarEvent">
    <label>Calendar Event</label>
    <attribute itemref="cal:headline"/>
    <attribute itemref="cal:startTime"/>
    <attribute itemref="cal:endTime"/>
    <attribute itemref="cal:location"/>

<pythonClass>OSAF.calendar.model.CalendarEvent.CalendarEvent</pythonCla
ss>
  </Kind>

  <Kind id="cal:Location">
    <label>Location</label>
    <attribute itemref="cal:name"/>
    <pythonClass>OSAF.calendar.model.Location.Location</pythonClass>
  </Kind>

  <Kind id="cal:Calendar">
    <label>Calendar</label>
    <attribute itemref="cal:name"/>
    <pythonClass>OSAF.calendar.model.Calendar.Calendar</pythonClass>
  </Kind>

</DomainSchema>
```

Calendar Schema 'repository' XML (output of the transform)

```
<?xml version="1.0" encoding="UTF-8"?>
<items
xmlns:schema="http://osafoundation.org/2003/08/schema_definition">
  <item>
    <name>startTime</name>
    <kind type="path">//Schema/Model/AttrDef</kind>
    <ref type="path" name="Type">//Schema/Model/AttrDef/DateTime</ref>
    <attribute name="Cardinality">single</attribute>
  </item>
  <item>
    <name>endTime</name>
    <kind type="path">//Schema/Model/AttrDef</kind>
    <ref type="path" name="Type">//Schema/Model/AttrDef/DateTime</ref>
    <attribute name="Cardinality">single</attribute>
  </item>
  <item>
    <name>headline</name>
    <kind type="path">//Schema/Model/AttrDef</kind>
    <ref type="path" name="Type">//Schema/Model/AttrDef/String</ref>
```

```
    <attribute name="Cardinality">single</attribute>
  </item>
  <item>
    <name>location</name>
    <kind type="path">//Schema/Model/AttrDef</kind>
    <ref type="path"
name="Type">//Schema/OSAF/CalendarSchema/Location</ref>
    <attribute name="Cardinality">single</attribute>
  </item>
  <item>
    <name>calendar</name>
    <kind type="path">//Schema/Model/AttrDef</kind>
    <ref type="path"
name="Type">//Schema/OSAF/CalendarSchema/Calendar</ref>
    <attribute name="Cardinality">set</attribute>
  </item>
  <item>
    <name>name</name>
    <kind type="path">//Schema/Model/AttrDef</kind>
    <ref type="path" name="Type"/>
    <attribute name="Cardinality">single</attribute>
  </item>
  <item>
    <name>CalendarEvent</name>
    <kind type="path">//Schema/Model/Kind</kind>
    <ref name="AttrDefs">
      <ref type="path"
name="headline">//Schema/OSAF/CalendarSchema/headline</ref>
      <ref type="path"
name="startTime">//Schema/OSAF/CalendarSchema/startTime</ref>
      <ref type="path"
name="endTime">//Schema/OSAF/CalendarSchema/endTime</ref>
      <ref type="path"
name="location">//Schema/OSAF/CalendarSchema/location</ref>
    </ref>
    <ref name="SuperKind">
      <ref type="path">//Schema/Model/Item</ref>
    </ref>
    <attribute type="model.schema.Types.Class" name="Class"/>
  </item>
  <item>
    <name>Location</name>
    <kind type="path">//Schema/Model/Kind</kind>
    <ref name="AttrDefs">
      <ref type="path"
name="name">//Schema/OSAF/CalendarSchema/name</ref>
    </ref>
    <ref name="SuperKind">
      <ref type="path">//Schema/Model/Item</ref>
    </ref>
    <attribute type="model.schema.Types.Class"
name="Class">OSAF.calendar.model.Location.Location</attribute>
  </item>
  <item>
    <name>Calendar</name>
    <kind type="path">//Schema/Model/Kind</kind>
    <ref name="AttrDefs">
```

```
      <ref type="path"
name="name">//Schema/OSAF/CalendarSchema/name</ref>
    </ref>
    <ref name="SuperKind">
      <ref type="path">//Schema/Model/Item</ref>
    </ref>
    <attribute type="model.schema.Types.Class"
name="Class">OSAF.calendar.model.CalendarEvent.CalendarEvent</attribute
>
  </item>
</items>
```

-- KatieCappsParlante - 16 Aug 2003

XSL Transform to XHTML

```
<xsl:stylesheet version="1.0"
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
     xmlns:chan="http://osafoundation.org/2003/08/CoreSchema">
   <xsl:output method="html" encoding="ISO-8859-1"/>
   <xsl:key name="attr" match="chan:AttributeDefinition" use="@id"/>

   <!-- note on namespaces: since the source format uses namespaces,
ALL LocationPaths must use a prefix here
        The default namespace doesn't make a difference. -->

   <!-- starting point -->
   <xsl:template match="chan:DomainSchema">
     <html>
       <head>
         <title>Chander Schema Documentation</title>
       </head>
       <body>
         <h1>Chandler Schema Documentation</h1>
         <xsl:apply-templates select="chan:Kind"/>
       </body>
     </html>
   </xsl:template>

   <!-- match once here for each Kind -->
   <xsl:template match="chan:Kind">
     <h2 class="kind"><a name="{translate(chan:label, '
','_')}"><xsl:value-of select="chan:label"/></a></h2>
     <p class="kinddesc"><xsl:value-of select="chan:comment"/></p>
     <table border="1">
       <tr>
         <th>Attribute</th>
         <th>Description</th>
       </tr>
       <xsl:for-each select="chan:attribute">
         <tr>
           <td><xsl:value-of select="key('attr',
@itemref)/chan:label"/></td>
           <td><xsl:value-of select="key('attr',
@itemref)/chan:comment"/></td>
         </tr>
       </xsl:for-each>
     </table>

   </xsl:template>

 </xsl:stylesheet>
```

Contributed by MicahDubinko

-- KatieCappsParlante - 22 Aug 2003