Mozilla XUL Templates rule language

Mozilla eXtensible User interface Language XUL (pronounced “zool”) (http://developer.mozilla.org/en/docs/XUL) is an XML-based language for building cross-platform browser-based applications. It is currently limited to running under browsers using the Gecko engine, most notably Mozilla Firefox. In fact, the Mozilla platform is constructed on top of XUL.

XUL provides extensive content creation and layout features where the presentation is composed UI components (XUL widgets). Within the Mozilla development framework, XUL is complemented by related technologies that include eXtensible Bindings Language (XBL) that is used for extending XUL by creating new tags, Overlays that are used for providing external customisations (*branding*) to the already existing interface, Cross Platform Component Object Model (XPCON) and XPConnect are used for writing and accessing to external native code components incorporating application logic that requires additional performance that cannot run in JavaScript, and finally, XPInstall used for packaging and installing XUL application components.

The XUL content is internally represented and merged from W3C Resource Definition Language (RDF) data stores. RDF is the language for representing directed labelled graphs, essentially collections of triples *(subject,predicate,object)*. To reiterate, *all* the UI elements as well as external data is represented by RDF. The rationale behind this decision was due to two main reasons: the need to *aggregate* disparate data (such as browser bookmarks, mail messages, local file references etc.) and to *merge* data from different local and remote stores. The latter functionality is especially powerful in RDF resulting in compelling use cases such as merging of company-wide bookmarks and user bookmarks or specifying local annotations on remotely specified interface (such as Instant Messenger buttons that the user could override locally). The RDF data stores are used both for representing the internal layout of the browser and for integrating the Semantic data available locally or on the web, which makes XUL a powerful multi-purpose content integration framework.

The part of Mozilla XUL that is most interesting to the W3C RIF WG is the XUL Template mechanism that uses the rules technology. The *builder* in the engine is responsible for generating content using XUL markup and referenced data stores. If a content element (for example, a *vbox*) has an associated data stored referenced from the *datasoures* attribute, the builder will look for template rules included with the element to execute queries on the data store and generate other elements.

The rules used by XUL Templates include the *conditions* and *action* parts. The conditions part is used for determining the matches for the information retrieved by querying the data store. The action part is used to create UI content based on the matches. The rules may also include a *bindings* element that is used for generating optional information that is not used in the rules processing.

The conditions part loosely corresponds to navigating the RDF graph beginning with the current node. The conditions may include the *content* tag used for specifying the starting point of the processing, multiple *triple* tags used for specifying atomic queries to the RDF store (with the restriction that one node and predicate are originally instantiated), and member tag used for extracting element of an RDF collection that may be represented by the node returned by the precious query. The graph navigation is performed by stringing one or more *triple* elements together in such a way that each *triple* queries the RDF store for the opposite end of a graph edge given one node. This is directly comparable to Sideway Information Passing (SIP) used in magic set transformations in which the values of the instantiated variables are used to find values for other variables in the predicate (for triples, the other end of the graph edge).

A template may contain more than one rule each executed in sequence. The engine maintains a local RDF store referred to as *network of information* that remains for the lifetime of the template. The method used for maintaining (adding and deleting) information in this network is derived from a RETE algorithm. However, simply referring to the engine as a forward chaining one would not be correct. In fact, during the content building phase, the rules are processed in a top-down fashion, effectively using backward chaining.

The content building starts at the top level using a *seed* that can be compared to the top level query. The seed corresponds to a node in the RDF graph that becomes the current node. In each iteration the rules application results in (1) generation of new matches that correspond to nodes that are reached by graph navigation from the current node (with the corresponding triples added to the network associated with the template) and (2) generation of UI content for the new nodes. All nodes reached by the graph navigation executed by the rule are again recursively examined as starting points for further applications of the rules—not unlike the XSL templates.

We finish this short outlook with two examples taken from the XUL Template Guide (http://developer.mozilla.org/en/docs/XUL:Template_Guide). The first example uses implicit RDF queries (without using the *triple* elements). Consider the following RDF data about the neighbourhood streets and houses.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:r="http://www.xulplanet.com/rdf/"
xmlns:nso="http://www.daml.org/2001/09/countries/country-ont#">

<rdf:Bag rdf:about="http://www.xulplanet.com/rdf/myneighbourhood">
 <rdf:li>
  <rdf:Seq rdf:about="http://www.xulplanet.com/rdf/marion" dc:title="Marion Street">
   <rdf:li rdf:resource="http://www.xulplanet.com/rdf/marion/16"/>
   <rdf:li rdf:resource="http://www.xulplanet.com/rdf/marion/18"/>
  </rdf:Seq>
 </rdf:li>

 <rdf:li>
  <rdf:Seq rdf:about="http://www.xulplanet.com/rdf/garden" dc:title="Garden Avenue">
   <rdf:li rdf:resource="http://www.xulplanet.com/rdf/garden/25"/>
   <rdf:li rdf:resource="http://www.xulplanet.com/rdf/garden/37"/>
  </rdf:Seq>
 </rdf:li>
</rdf:Bag>
```

```
<r:House rdf:about="http://www.xulplanet.com/rdf/marion/16" r:floors="2" r:address="16"/>
<r:House rdf:about="http://www.xulplanet.com/rdf/marion/18" r:floors="2" r:address="18"/>
<r:House rdf:about="http://www.xulplanet.com/rdf/garden/25" r:floors="1" r:address="25"/>
<r:House rdf:about="http://www.xulplanet.com/rdf/garden/37" r:floors="4" r:address="37"/>
</rdf:RDF>
```

The following XUL document will recursively apply two rules to the structure of the neighbourhood and generate Web content based on the type and data of the RDF nodes matched. The first rule only applies to the nodes of type *House* so it is used for leaves in the graph. It generates a *groupbox* with the *dc:title* attribute used for its heading. The second rule applies to streets (and any top level grouping that may be introduced later) and generates two labels based on two literal (data) nodes reachable via relationships *address* and *floors*, respectively.

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

<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<window xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

<label value="Houses in my Neighbourhood"/>

<hbox datasources="template-guide-streets.rdf"
      ref="http://www.xulplanet.com/rdf/myneighbourhood"
      xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <template>
    <rule rdf:type="http://www.xulplanet.com/rdf/House">
      <vbox uri="rdf:*" class="box-padded">
        <label value="Address: rdf:http://www.xulplanet.com/rdf/address"/>
        <label value="Floors: rdf:http://www.xulplanet.com/rdf/floors"/>
      </vbox>
    </rule>
    <rule>
      <groupbox uri="rdf:*" class="box-padded">
        <caption label="rdf:http://purl.org/dc/elements/1.1/title"/>
      </groupbox>
    </rule>
  </template>
</hbox>
</window>
```

The following Figure (http://developer.mozilla.org/samples/xultemp/template-guide-ex25.xul) shows the results of presenting this XUL document in the Firefox browser.

The second example queries a photo album represented by the following RDF data.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:r="http://www.xulplanet.com/rdf/">
```

```
  <rdf:Seq rdf:about="http://www.xulplanet.com/rdf/myphotos">
    <rdf:li rdf:resource="http://www.xulplanet.com/ndeakin/images/t/palace.jpg"/>
    <rdf:li rdf:resource="http://www.xulplanet.com/ndeakin/images/t/canal.jpg"/>
    <rdf:li rdf:resource="http://www.xulplanet.com/ndeakin/images/t/obelisk.jpg"/>
  </rdf:Seq>

  <rdf:Description rdf:about="http://www.xulplanet.com/ndeakin/images/t/palace.jpg"
                   dc:title="Palace from Above">
    <r:country resource="http://www.daml.org/2001/09/countries/iso#IT"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.xulplanet.com/ndeakin/images/t/canal.jpg"
                   dc:title="Canal">
    <r:country resource="http://www.daml.org/2001/09/countries/iso#NL"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://www.xulplanet.com/ndeakin/images/t/obelisk.jpg"
                   dc:title="Obelisk">
    <r:country resource="http://www.daml.org/2001/09/countries/iso#IT"/>
  </rdf:Description>

  <rdf:Description about="http://www.daml.org/2001/09/countries/iso#IT"
                     dc:title="Italy"/>

  <rdf:Description about="http://www.daml.org/2001/09/countries/iso#NL"
                     dc:title="Netherlands"/>
</rdf:RDF>
```

The XUL document that uses explicit RDF queries using the *triple* element is shown below followed by the resulting Web content (http://developer.mozilla.org/samples/xultemp/template-guide-ex15.xul).

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

<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>

<window xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

<label value=""/>

<hbox datasources="template-guide-photos3.rdf"
      ref="http://www.daml.org/2001/09/countries/iso#IT">
  <template>
    <rule>
      <conditions>
        <content uri="?start"/>
        <triple subject="?start"
                predicate="http://purl.org/dc/elements/1.1/title"
                object="?countrytitle"/>

        <triple subject="?photo"
                predicate="http://www.daml.org/2001/09/countries/iso-3166-ont#Country"
                object="?start"/>

        <triple subject="?photo"
                predicate="http://purl.org/dc/elements/1.1/title"
                object="?title"/>
      </conditions>
      <action>
        <vbox class="box-padded" uri="?photo">
          <image src="?photo"/>
          <label value="?title"/>
          <label value="Country: ?countrytitle"/>
        </vbox>
      </action>
    </rule>
  </template>
</hbox>
```

```
</window>
```

Obelisk
Country: Italy

Palace from Above
Country: Italy