XNAT Customizations

Jenny Gurney (gurneyj@wustl.edu)

Audience: XNAT site administrators and developers

Goals: Demonstrate the XNAT customization process, discussing necessary languages and best practices

http://xnat.wikispaces.com/XNAT+2010+Workshop+-+Customizations

3 Common Reasons to Customize XNAT

- Extend the types of data you can store in XNAT
- Modify the graphical user interface
- Customize data listings

Extending the XNAT Data Model

- Step-by-step
  1. Model data in XML schema definition (XSD)
     projects/xnat/src/schemas/<schema>/<schema>.xsd
  2. If new schema, add to config file
     `projects/xnat/InstanceSettings.xml`
  3. Run XNAT update script (after Tomcat shutdown)
     `bin/update.sh –Ddeploy=true`
  4. Run the SQL update script to create database tables
     `psql –f deployments/sql/update-xnat.sql <db>`
  5. Verify scripts' expected end products exist

  (All file paths relative to `projects/xnat/src`)
  - Database tables <schema>_<datatype>
  - Edit and report templates
    `base-templates/screens/`
    `XDATScreen_*_<schema>_<datatype>.vm`
  - Java screen object
    `java/org/nrg/xdat/turbine/modules/screens/`
    `XDATScreen_*_<schema>_<datatype>.java`
  - Display document
    `schemas/<schema>/display/<schema>_<datatype>.xml`

- XNAT XSD best practices
  - Keep names short
  - Limit unbounded elements
  - Put unbounded type in element of maxoccurs = 1
  - Use annotation tags to specify unique field(s)
  - Limit enumerations

- Graphical tools to make writing XSD much easier!
  - XMLSpy[1] (licensed): Professional or Enterprise Edition
  - Oxygen[2] (licensed): XML Editor
  - Eclipse[3] (free): Java or Java EE Developers version

[1] http://www.altova.com
[2] http://www.oxygenxml.com
[3] http://www.eclipse.org/downloads

- Get automatic REST API support and initial edit and reports pages by extending a recommended type
  - subjectAssessorData
  - imageAssessorData

Modifying the User Interface

- Step-by-step

(All file paths relative to `projects/xnat/src`)

1. Copy initial edit and report templates, `XDATScreen_*_<schema>_<datatype>.vm`, from `base-templates/screens` to `templates/screens`
2. Optional: Add/modify *.js file(s) in directory: `scripts`
3. Optional: Add/modify *.css files in directory: `style`
4. Optional: To add to or modify the Context, copy `XDATScreen_*_<schema>_<datatype>.java from java/org/nrg/xdat/turbine/modules/screens` to `java/org/apache/turbine/app/xnat/modules/screens`
5. Optional: To add custom actions, create Java file in `java/org/apache/turbine/app/xnat/modules/actions`
   - No mandatory name pattern
   - Must explicitly set action in .vm file
6. Run update process
   - If no Java updates, then run:
     `bin/quick-deploy-templates.sh`
   - If Java updates and compiling code with (IDE Eclipse or other), then run:
     `bin/quick-deploy.sh -Dclass.dir=<compile path>`
   - If Java updates and no IDE, then run (after Tomcat shutdown): `bin/update.sh –Ddeploy=true`

- XNAT and Velocity Concepts
  - XNAT uses Velocity 1.3.1
  - XNAT uses Velocity to generate HTML from Velocity templates language (VTL) in .vm files
  - The context: container used by Velocity to pass data objects back and forth between Java and .vm files
  - #macro elements, Velocimacros, allow template designers to define a repeated segment of VTL

- XNAT Context Objects
  - $om – object containing getters and setters for all named fields in datatype and parent datatypes

    A corresponding Java class (automatically generated)

    `java/org/nrg/xdat/om/base/auto/Auto<schema><datatype>.java`

    can be customized to provide custom methods
  - $turbineUtils – object containing set of utilities for general use

    The most commonly used method:

    `getTemplateName(String module, String dataType, String project)`

    allows project-specific templates

- XNAT Velocimacros[4]
  - XNAT includes many data input HTML/Velocity code fragments in TurbineMacros.vm
  - Simplify common HTML tasks in XNAT-Velocity environment

Velocity 1.3.1 Cheat Sheet

```
Variable notation  $
Comments  ## (one line), #* *# (multi-line)
Arithmetic Operators +, -, *, /, %
Relational   ==   ## not just equivalency, can be
used to compare objects
Logic Operators    &&, ||, !
Range Operator  [n..m]  (used in loops)
Escape Character  \

Reference a variable $foo
or $!foo ## if value is null, print nothing

Assign a value to a variable
#set( $foo = "Velocity" ) ## string literal

Refer to a hashtable key or a get (Address) method
$customer.Address

Conditional statement
#if( $foo ) <p>Velocity!</p>
#elseif( $foo2 == “cool” ) <p>XNAT!</p> #end

Loop
#foreach($criterion in $criteria) ## loop ArrayList
    Current value:  $!criterion
#end

Incorporate text from another template
#parse( "parsefoo.vm" )
```

[4] http://xnat.wikispaces.com/XNAT+Reports#XNAT%20Codebase:%20Understanding%20XNAT%20Reports-Pre-defined%20Turbine%20Velocity%20Macros

Customizing Data Listings

- Data listings are specified in xml documents called XNAT Display Documents[5]

- Step by step
  1. Modify or create display document `schemas/<schema>/display/<schema>_<datatype>.xml`
  2. Run: `bin/quick-deploy-templates.sh`
  3. To avoid server restart, in GUI, go to Administer, More Options, Refresh Display Documents

- Display document elements
  - <DisplayField> defines a schema field available to be used in listings and searches
  - <DisplayFieldElement> can be used to specify the schema field(s) or view column(s) used in a <DisplayField>
  - <DisplayVersion> specifies a combination of <DisplayField>s and a name for the combination
  - <SchemaLink> includes fields from another Schema Element, whose relation to the Display element is not clear from the XML Schema
  - <Mapping> defines a mapping table
  - <MappingColumn> defines how the root element and the linked element relate to the mapping table in <SchemaLink>
  - <SQLView> allows developer to create custom database views and is only way to include unbounded elements in a listing
  - <ViewLink> defined to join custom views to a displayable element, then MappingColumns are then used to connect the view to the schema element
  - Arcs: to join displayVersions from separate displayable elements, an arc can be defined to specify how the two elements are related. Declaring an Arc takes two steps
    1. Define an <Arc-Definition> defined with a unique id and a collection of <CommonField>s, the two elements are related by another element, the <Bridge-Element>
    2. Define Arc Membership <Arc> included within the member elements to imply that this element is a member of that Arc

[5] http://xnat.wikispaces.com/XNAT+Display+Documents