XNAT Data Type Development

This page describes how to create new custom data types for XNAT. You can find out how to work with data types that are already in XNAT in the section Managing Data Types in XNAT.

Data in XNAT comes in two main varieties: system data that describes XNAT's configurations, settings, and so on, and core XNAT data that represents the research data that is XNAT's primary reason for existing. There are a few steps when creating a custom data type for XNAT:

- Creating a new data type schema
- Configuring your data type for XNAT
- Creating edit and report pages for your data type

Creating new XNAT data types is a complex topic. This section provides a good introduction for working with data types in the context of developing an XNAT plugin.

Creating New Data Type Schema

This core data is maintained in XNAT as a collection of objects grouped by the type of data described by the objects–for example, there are objects that represent MR, PET, or CT imaging sessions, radiological reads of imaging data, FreeSurfer processing, and so on. These objects are stored as XML documents. The format for each type of XML data object is defined by an XML schema document in the XSD format. Creating a new data type in XNAT consists of creating the XSD that defines the data type itself. Creating a new data type in an XNAT plugin consists of putting the schema file into the appropriate place in your built-out plugin jar.

The most important thing about data-type schemas, from the point of view of including them in a plugin, is where to put the schema files in the plugin project structure. Gradle expects a standard directory layout when building a project's source. The root folder for these is named **src**. Under **src**, you can have two folders, **main** and **test**. Although writing test code for your project is a noble and admirable pursuit, we'll leave promoting and explaining that to others and focus solely on the contents of the **main** folder.

Within your **main** development folder, there can be a large number of subfolders depending on the types of data handled by the plugins configured in your **build.gradle** project. In the case of the **java** plugin, the supported subfolders are **java** and **resources**. The purpose of the **java** folder is strictly to contain your Java classes. The purpose of the **resources** folder, on the other hand, is to contain pretty much everything else. XNAT data-type schemas fall under the rubric of "pretty much everything else", so that's where you'll start. You should create your schema file at /src/main/resources/schemas/*schema*/*schema*.xsd , replacing 'schema' with whatever you want to name your schema (for example, /src/main/resources/schemas/*example*/*example*.xsd).

Below is an example of what a schema could look like. After the code block, this code will be explained.

Completing the example XSD

```
<xs:schema targetNamespace="http://nrg.wustl.edu/example" xmlns:example="http://nrg.wustl.edu/example" xmlns:
xnat="http://nrg.wustl.edu/xnat" xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:import namespace="http://nrg.wustl.edu/xnat" schemaLocation="../xnat/xnat.xsd"/>
    <xs:element name="BiosampleCollection" type="example:biosampleCollection"/>
    <xs:complexType name="biosampleCollection">
        <xs:annotation>
            <xs:documentation>Indicates whether the required biosample collections have been acquired per study
protocol.</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="xnat:subjectAssessorData">
                <xs:sequence>
                    <xs:element name="dna">
                        <xs:annotation>
                            <xs:documentation>DNA</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                            <xs:restriction base="xs:boolean"/>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="rna">
                        <xs:annotation>
                            <xs:documentation>RNA</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                            <xs:restriction base="xs:boolean"/>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="plasma">
                        <xs:annotation>
                            <xs:documentation>Plasma</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                            <xs:restriction base="xs:boolean"/>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="serum">
                        <xs:annotation>
                            <xs:documentation>Serum</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                            <xs:restriction base="xs:boolean"/>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="csf">
                        <xs:annotation>
                            <xs:documentation>CSF</xs:documentation>
                        </xs:annotation>
                        <xs:simpleType>
                            <xs:restriction base="xs:boolean"/>
                        </xs:simpleType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

The first few lines of the xsd are mostly standard. Just make sure that you set the **targetNamespace** attribute to the namespace you want to use for your schema. The **xmlns:example** and **xmlns:xnat** attributes define URIs that can be used to uniquely identify schemas that use a particular data type. This lets you change the prefix (in this case, **example** and **xnat**) but still be able to recognize the relationship to the same schema definition. Note that the URI used for this does not need to resolve to a working accessible web address, even though it can.

For the import tag, notice that the **namespace** attribute uses the same URI used by the **xmlns:xnat** attribute in the **<xs:schema>** element. This tells the XML parser that references to elements within the **xnat** namespace should be validated against the schema found at the indicated location. In the case of your plugin, the location doesn't really indicate whether the XNAT schema can be found *now* (although you can copy that file there in order to let any XML editor–including the built-in XML editors in IntelliJ IDEA or Eclipse–properly resolved the **xnat** namespace and provide assistance in locating XML elements, attributes, and the like), but, more importantly, where the XNAT schema *will be found* once your schema is deployed into the XNAT application. Since all data-type schemas in XNAT are located in **schemas/*schema*/*schema*.xsd**, any other schema you want to reference will be located at **../*schema*/*schema*.xsd**.

After the import tag comes the element tag. The element tag in the example above declares the **BiosampleCollection** element and says that it's of type **example:biosampleCollection**. The **example** at the beginning means that the type is declared here in our current namespace, while **biosampleCollection** indicates the actual type.

After the element tag, you define the **complexType**, which is where you tell XNAT what type of data it is. In this example, we're creating a subject assessor data type (which just means a data type that contains data about a research subject).

This is all you need to do to create a data type xsd! However, a data type with just these tags is not very useful because it doesn't extend the base subjectAssessorData type with any new properties. For the **biosampleCollection** data type, the additional properties we want to add are simple boolean yes/no properties. Either DNA was collected from the subject or it was not. After we have added a few boolean properties, we are ready to start using the **biosampleCollection** data type. To do this, we need to add it to a plugin, following the steps from Working with XNAT Plugins.

Configuring Data Types in Plugin Configurations

In older versions of XNAT, data types had to be manually enabled through the Admin UI before your XNAT could support data of that type. As of XNAT 1.7, the XNAT plugin service can automatically detect your data type and enable it for you after your plugin is installed. In order to enable this behavior, you need to ensure that your annotations on the plugin class are properly written. See **XNAT Plugin Configurations: Configuring Data Models** for details.

Data Entry and Reporting for a Data Type

Once you have added your data type to XNAT, you may want to add an instance of that data type. For example, if you added a session data type, you may want to add an image session of that type.

Default data entry and reporting screens

When you add a new data type, default edit and report pages will be created for it. These pages are very basic and have text boxes for fields that you may wish to provide dropdowns, checkboxes, or radio buttons for instead. Here is what a default edit page for a data type looks like:

PROJECT: aa > SUBJECT: Sample_Patient > SESSION: aa > **ww**

ScreeningAssessment Details

Primary Project: aa
Experiment ID in this project: ww

ScreeningAssessment ID XNAT_E00008

imageSession_ID	XNAT_E00009
date	December ▼ 19 ▼ 2016 ▼
time	04:05:06
duration	
delay	
delay/ref_expt_id	
note	
acquisition_site	
visit_id	
visit	
version	
original	
protocol	
label	ww
rater	

And here is what the default report page for that data type looks like:

PROJECT: aa > SUBJECT: Sample_Patient > SESSION: aa > **ww**

ScreeningAssessment Details

imageSession_ID	XNAT_E00009
date	2016-12-19
time	04:05:06
duration	
delay	
delay/ref_expt_id	
note	
acquisition_site	
ID	XNAT_E00008
project	aa
visit_id	
visit	
version	
original	
protocol	
label	ww
rater	
comments	
pass	

Actions

- Edit
- View
- Download XML
- Email
- Manage Files
- Delete

Related Items

scr:screeningAssessment/provenance

scr:screeningAssessment/investigator

title
firstname
lastname
institution
department
email
phone
ID

scr:screeningAssessment/validation

Customized data entry and reporting screens

This example references the Rad Read assessor data type, which is defined in this plugin: **https://bitbucket.org/nrg_customizations/nrg_plugin_radread**

While these default edit and report pages can be very valuable in allowing you to quickly start adding new types of data to XNAT by merely adding a new schema, they often display the data in a way that is unhelpful to users of your site and prompt users for data without indicating how it should be formatted. For example, the time field for screening assessments should be in the hh:mm:ss format, but there is no indication of this on the default page for creating /editing screening assessments. You may also wish to make your edit forms more interactive. For reasons like these, it can be helpful to create customized screens for data types.

One example of a data type with customized edit and report pages is the Radiological Assessment. To develop new template files for edit and report views and have them displayed in XNAT, create Velocity templates in your plugin repo with the following naming:

Edit	/src/main/resources/META-INF/resources/templates/screens/XDAT_Screen_edit_{xsi_type}.vm
Report	/src/main/resources/META-INF/resources/templates/screens/XDAT_Screen_report_{xsi_type}.vm

Where "{xsi_type}" is the URL-friendly translation of your schema's xsiType. For example, the Radiology Read xsiType is **`rad:radiologyReadData`**. The URL-friendly translation of this for template naming is **`rad_radiologyReadData`**.

Here is the custom edit page:

PROJECT: qw > SUBJECT: qe > SESSION: yt > **yt_RAD_1483977113926**

Adult Radiological Assessment

XNAT_E00004 / yt_RAD_1483977113926

Session Id:	yt
Session Date:	2013-01-04
Participant:	qe
Gender:	
Age:	--

Date: January 9 2017

Reader: A. Admin

Exam: Brain without contrast Other

History: qw

Technique: Axial T1w, T2w, and T2*

Comparison: (SELECT)

(SELECT)
qr (:XNAT_E00001)
CUSTOM

Findings

	None Mild Moderate Severe
	None Mild Moderate Severe
Cortical Atrophy	None Mild Moderate Severe
Large Infarct	0 1 2 3+
Small Infarct (<1 cm)	0 1 2 3+
Microbleeds	0 1-4 5-10 >11
Site of Microbleeds	subcortical deep mixed cannot assess
Other Significant Findings	yes no

Additional Findings:

Aging Changes: Normal None Mild Moderate Severe

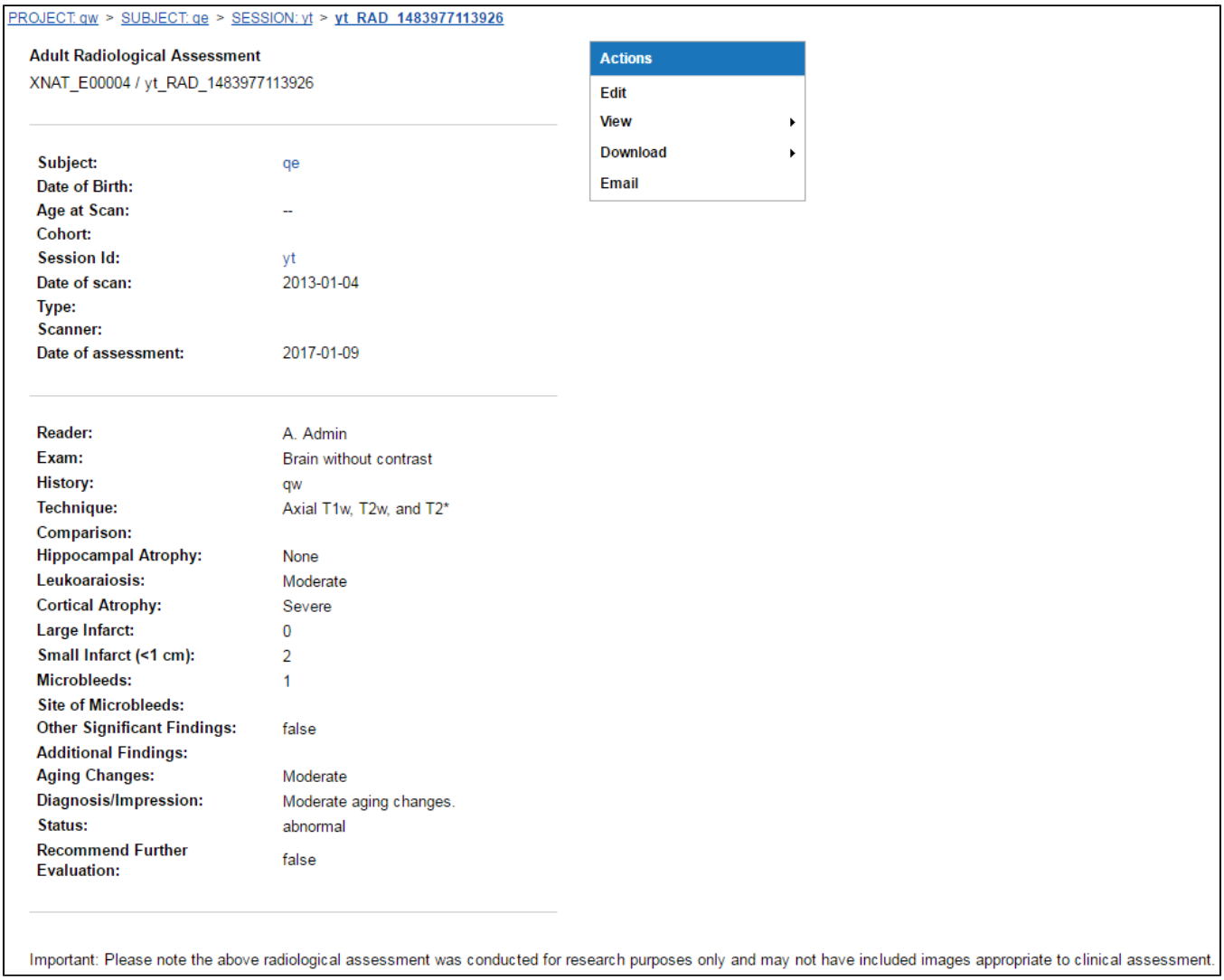
Diagnosis/Impression: Moderate aging changes.

Status: Abnormal

Recommend Further Eval

Here you can see some of the benefit of customizing the edit page. Instead of having a text box for each of the Findings where the user has to guess what value they should enter, this custom page has radio buttons for the possible values and has a gray background every other line for easy readability. You can also see there is a Comparison field from which you can select another session that you want to look at for comparison when creating/editing a radiology read. Another feature of note is the interactivity of aging changes. When the person editing the radiology read selects one of the Aging Changes radio buttons, the Diagnosis/Impression is modified to add information about aging changes. This information is then updated when the user selects a different radio button.

After creating a Radiology Read, you can see that the report page for it has been customized as well:



While much less customized than the edit page, there are some customizations here. The names of the different fields have been bolded, and a note about the clinical applicability of the Radiological Assessment has been added to the end.

Adding Custom Screens to a Plugin

If you wish to customize the edit and report pages for a plugin in XNAT 1.7, the best way to do that is to create a plugin with your customizations, or add them to one of your existing plugins. For general information about plugins, you should consult Working with XNAT Plugins. To start, you should find the XDATScreen_edit and XDATScreen_report .vm files that XNAT has generated for your data type. These should be located in xnat-templates/screens. You should use these as a starting point when creating your custom edit and report screens.

If you are using a new schema that you wrote, you may not find files for your data type located there. In that case, you can either use the edit and report pages of a similar data type as a starting point, or add the schema to a plugin and look for the edit and report pages in your webapp directory. Once you are satisfied with your customizations and are ready to test them, you should add these files to a plugin at **`/src/main/resources/META-INF/resources/templates/screens/`** relative to the top level of your plugin.

If you need to add additional variables to your velocity context because it does not currently contain values you want to have access to when constructing the page, you should create java files with the same names as your velocity files (except ending in .java instead of .vm). You should locate these files at **`/src/main/java/org/nrg/xnat/turbine/modules/screens/`** relative to the top level of your plugin. For example, here is the java file associated with the Radiological Assessment edit page:

XDATScreen_report_rad_radiologyReadData.java

```
package org.nrg.xnat.turbine.modules.screens;
import org.apache.turbine.util.RunData;
import org.apache.velocity.context.Context;
import org.nrg.xdat.turbine.modules.screens.SecureReport;

public class XDATScreen_report_rad_radiologyReadData extends SecureReport {
    public static org.apache.log4j.Logger logger = org.apache.log4j.Logger.getLogger
(XDATScreen_report_rad_radiologyReadData.class);


    public void finalProcessing(RunData data, Context context) {
        try{
            org.nrg.xdat.om.RadRadiologyreaddata om = new org.nrg.xdat.om.RadRadiologyreaddata(item);
            org.nrg.xdat.om.XnatImagesessiondata mr = om.getImageSessionData();
            context.put("om",om);
            System.out.println("Loaded om object (org.nrg.xdat.om.RadRadiologyreaddata) as context parameter
'om'.");
            context.put("mr",mr);
            System.out.println("Loaded mr session object (org.nrg.xdat.om.XnatImagesessiondata) as context
parameter 'mr'.");
            context.put("subject",mr.getSubjectData());
            System.out.println("Loaded subject object (org.nrg.xdat.om.XnatSubjectdata) as context parameter
'subject'.");
        } catch(Exception e){}
    }
}
```

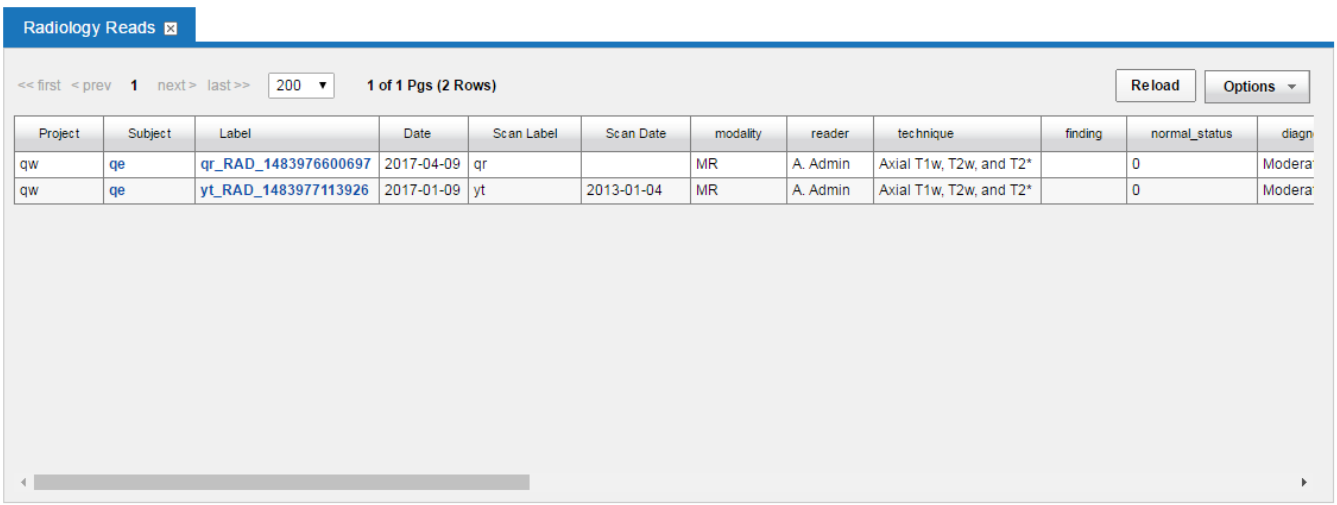
Once you are finished with the java and velocity files for the edit and report pages and have deployed them as part of your plugin, you should make sure that they are linked to from your XNAT site. If the data type is an experiment, you should be able to reach the edit page for the data type by clicking New->Experiment from the top bar, entering the name of your data type, and then clicking the link for it.

Since Radiological Assessments are experiment assessors, more is needed to make the edit page easily accessible to users of your site. To enable users to add Radiological Assessments from MR Session pages, we went to Administer->Data Types, clicked on xnat:mrSessionData, clicked Edit, and then entered a new Available Report Action. We added a new row to that table with Name equal to XDATScreen_edit_rad_radiologyReadData and Display Name equal to Add Rad Read. We then clicked the Submit button at the bottom of the page. After doing that, an Add Rad Read option appeared in the Actions box on the MR Session page. Clicking Add Rad Read then takes you to the edit Rad Read page.

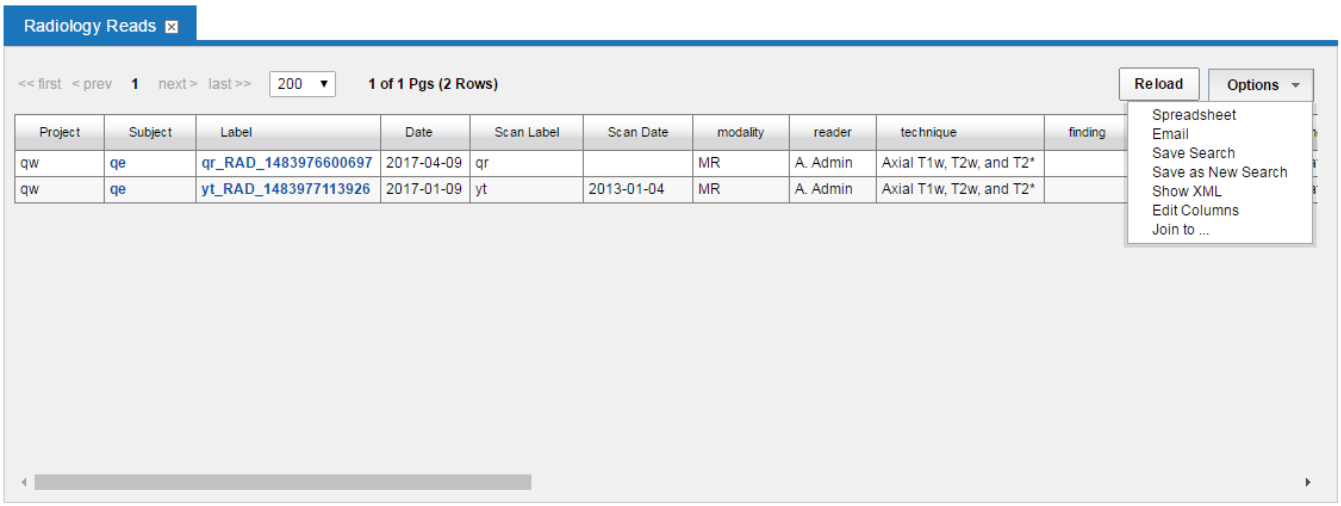
See: **Customizing User Interfaces in XNAT Plugins**

Data tables in XNAT based on data type

Once you have created some instances of your data type, you can view those you have created so far by clicking on Browse-> Data-> Your data type name. You are then shown a table like this:



If this table contains columns that you do not want to have show up, you can click Options-> Edit Columns to add/remove columns to/from the set of Current Fields:


Radiology Reads
<< first < prev 1 next > last >>
200
1 of 1 Pgs (2 Rows)
Reload
Options
Spreadsheet
Email
Save Search
Save as New Search
Show XML
Edit Columns
Join to ...
Project
Subject
Label
Date
Scan Label
Scan Date
modality
reader
technique
finding
qw
qe
qr_RAD_1483976600697
2017-04-09
qr
MR
A. Admin
Axial T1w, T2w, and T2*
qw
qe
yt_RAD_1483977113926
2017-01-09
yt
2013-01-04
MR
A. Admin
Axial T1w, T2w, and T2*