API Save Handlers

We use save handlers in the OpenMRS API to do a lot of the grunt work of setting values like creator, changed by, date voided, etc.

This saves us from having to rewrite all of the setCreator(Context.getAuthenticatedUser(); setDateCreated (new Date()) in many many places.

How it Works

The RequiredDataAdvice (add link) class is AOP'd around all methods that start with save/(un) void/(un)retire. (create/update are also in there for backwards compatibility to pre 1.5 modules).

If a method starts with that string and takes in an OpenmrsObject as the first parameter, it will be "handled".

A "handled" class will automatically have appropriate values put in like the creator, date changed, etc.

Which attributes are filled in depends on what interfaces the object extends.

In core, there are handlers for Auditable, Voidable, and Retireable.

Recursion on OpenmrsObject child collections is done on an object prior to it being "handled". E.g If a Patient is saved with a call to savePatient(patient), the save handlers will be used. Before calling the save handler for the Patient object though, the patient.patientIdentifiers list is iterated and each identifier gets passed to all save handlers. The changedBy/dateChanged objects are handled differently, see below.

Using the Handlers in your own Service

The handlers are automatically called around your own module's service methods if you follow a few simple steps:

1. Make sure your object extends the right parent class from openmrs core: Auditable, Voidable, or Retireable.
2. Make sure the method name starts with the right phrase. It should be "save__" or "void__", *"unvoid_", "retire__", or "unretire___"*
3. Add the "listener" with id="requiredDataInterceptor" to your service definition in the spring module application context file:
   <property name="preInterceptors">
   <ref bean="serviceInterceptors" />
   </property>

Disabling Handlers

All RequiredDataHandlers are called on all child collections the OpenmrsObject being handled. Therefore, for instance, if you void an Encounter, the VoidHandler is called for all Obs associated with that Encounter. In a one-to-many relationship, this is the functionality we want, but it is not necessarily what we want in a many-to-many case. For instance, in the Provider Management module, there is a "ProviderRole" object that contains the collection of RelationshipTypes that the particular ProviderRole can support. As more than one ProviderRole can support the same relationship type, when retiring a ProviderRole we do NOT want to retire the associated RelationshipTypes.

By annotating a Collection with a @DisableHandlers annotation, you specific that RequiredDataAdvice should NOT apply the specified handler(s) to a child collection. For example:

```
private class ClassWithDisableHandlersAnnotation extends BaseOpenmrsData {
        @DisableHandlers(handlerTypes = {VoidHandler.class, SaveHandler.class})
        private List<Person> persons;
}
```

Note: @DisableHandlers requires OpenMRS 1.9.0+, or 1.8.4+

Creating Your Own Handler

To add in your own handler, simply create a class in your module that extends SaveHandler/Retire Handler/VoidHandler and annotate it with the appropriate Handler annotation for the handled class.

Example Save Handler Code

AuditableSaveHandler