Extended Attributes 2

Description

Extension attributes are a method of adding new attributes to existing business objects without having to change the delivered code or database structure. Some minor coding is required, but all changes can be made without altering the files delivered with KFS. This allows upgrades to be performed much easier than if changes are made to core elements.

Extension attributes are implemented by creating a secondary table that shares the same primary key as the table being extended. The KNS has a built-in hook which allows an institution to attach these new attributes without needing to alter the object being extended. Most of the changes required are XML configuration. And, while the instructions below may seem long, most of it is the standard set of modifications which need to be done to create a new business object.

Implementation Instructions

In our example, we are adding an attribute called program code to the account table. This example makes program code a free-form field, rather than associating a reference table that can be used to lookup / validate values. However, it is perfectly feasible to associate a reference table with an extension attribute. It is also possible to use extension attributes in custom properties and existing JavaScript in transactional documents, e.g. to display the value next to the account description on accounting lines. See the `patches/program-code-extension-attribute` in the distribution for a full example demonstrating both of these things. Basically, once you have created an extension table using the steps below, the new attributes can be used just like normal attributes in the delivered system or standalone entities that you add to the system.

1. Create a table to hold the additional attributes (e.g. CA_ACCOUNT_EXT_T).
   The table should have the same primary key fields as the delivered table that you are adding to. See the table creation documentation for more information. For example...

   CA_ACCOUNT_EXT_T.table.ddl

```
CREATE TABLE CA_ACCOUNT_EXT_T(
        FIN_COA_CD VARCHAR2(2) CONSTRAINT CA_ACCOUNT_EXT_TN1 NOT NULL,
        ACCOUNT_NBR VARCHAR2(7) CONSTRAINT CA_ACCOUNT_EXT_TN2 NOT NULL,
        OBJ_ID VARCHAR2(36) CONSTRAINT CA_ACCOUNT_EXT_TN3 NOT NULL,
        VER_NBR NUMBER(8) DEFAULT 1 CONSTRAINT CA_ACCOUNT_EXT_TN4 NOT NULL,
        PGM_CD VARCHAR2(6),
     CONSTRAINT CA_ACCOUNT_EXT_TP1 PRIMARY KEY (
        FIN_COA_CD,
        ACCOUNT_NBR),
     CONSTRAINT CA_ACCOUNT_EXT_TC0 UNIQUE (OBJ_ID)
)
/
```

2. Create a Java object for the extension table (e.g. `edu.sampleu.kuali.module.chart.bo.AccountExtension`).
   It must implement the `org.kuali.core.bo.PersistableBusinessObjectExtension` interface and should extend the `org.kuali.core.bo.PersistableBusinessObjectExtensionBase` class. For example...

edu.sampleu.kuali.module.chart.bo.AccountExtension.java

```
package edu.sample.kuali.module.chart.bo;

import java.util.LinkedHashMap;

import org.kuali.core.bo.PersistableBusinessObjectExtensionBase;
import org.kuali.kfs.context.SpringContext;
import org.kuali.module.chart.bo.Account;
import org.kuali.module.chart.service.AccountService;

public class AccountExtension extends PersistableBusinessObjectExtensionBase {

    private String chartOfAccountsCode;
    private String accountNumber;
    private String programCode;

    public String getChartOfAccountsCode() {
        return chartOfAccountsCode;
    }

    public void setChartOfAccountsCode(String chartOfAccountsCode) {
        this.chartOfAccountsCode = chartOfAccountsCode;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public void setAccountNumber(String accountNumber) {
        this.accountNumber = accountNumber;
    }

    public String getProgramCode() {
        return programCode;
    }

    public void setProgramCode(String programCode) {
        this.programCode = programCode;
    }
}
```

3. Create/modify *your* OJB repository file for the module.
    a. Add a class descriptor for the extension business object.
    b. Copy the repository descriptor for the base business object from the delivered OJB repository file (e.g. `org.kuali.module.chart.OJB-repository-chart.xml`) into your OJB repository file and add a reference to the extension business object. For example...

edu.sampleu.kuali.module.chart.sampleu-repository-chart.xml

```
<descriptor-repository version="1.0">
    <class-descriptor
class="edu.sampleu.kuali.module.chart.bo.AccountExtension"
table="CA_ACCOUNT_EXT_T">
        <field-descriptor name="chartOfAccountsCode" column="FIN_COA_CD"
jdbc-type="VARCHAR" primarykey="true" index="true" />
        <field-descriptor name="accountNumber" column="ACCOUNT_NBR"
jdbc-type="VARCHAR" primarykey="true" index="true" />
        <field-descriptor name="programCode" column="PROGRAM_CD"
jdbc-type="VARCHAR" />
        <field-descriptor name="objectId" column="OBJ_ID"
jdbc-type="VARCHAR" index="true" />
        <field-descriptor name="versionNumber" column="VER_NBR"
jdbc-type="BIGINT" locking="true" />
    </class-descriptor>
    <class-descriptor class="org.kuali.module.chart.bo.Account"
table="CA_ACCOUNT_T">
        <field-descriptor name="chartOfAccountsCode" column="FIN_COA_CD"
jdbc-type="VARCHAR" primarykey="true" index="true" />
        <field-descriptor name="accountNumber" column="ACCOUNT_NBR"
jdbc-type="VARCHAR" primarykey="true" index="true" />
        <field-descriptor name="accountName" column="ACCOUNT_NM"
jdbc-type="VARCHAR" />
        ...
        <reference-descriptor name="extension"
class-ref="edu.sampleu.kuali.module.chart.bo.AccountExtension"
auto-retrieve="true" auto-update="object" auto-delete="object"
proxy="false">
            <foreignkey field-ref="chartOfAccountsCode" />
            <foreignkey field-ref="accountNumber" />
        </reference-descriptor>
    </class-descriptor>
</descriptor-repository>
```

Be sure to retain all of the original field, reference, and collection descriptors from the original OJB class descriptor. If any are missing, the system may not be able to persist to/retrieve from the corresponding attributes of the business object.

4. Update the data dictionary.
    a. Make the business object entry changes.
        i. Add a data dictionary file for the extension business object.
        This would be completed like any other business object except that it does not need an Inquiry or Lookup section. For example...

edu.sampleu.kuali.module.chart.datadictionary.AccountExtension.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dictionaryEntry PUBLIC
    "-//Kuali Project//DTD Data Dictionary 1.0//EN"

"../../../../../../org/kuali/core/datadictionary/dataDictionary-1_0.dt
d">
<dictionaryEntry>
    <businessObject
businessObjectClass="edu.sampleu.kuali.module.chart.bo.AccountExtensio
n">
        <attributes>
     <attributeReference sourceAttributeName="chartOfAccountsCode"
name="chartOfAccountsCode"
sourceClassName="org.kuali.module.chart.bo.Chart" />

     <attributeReference sourceAttributeName="accountNumber"
name="accountNumber"
sourceClassName="org.kuali.module.chart.bo.Account" />

            <attribute name="programCode" forceUppercase="true">
                <label>Program Code</label>
                <shortLabel>Program</shortLabel>
                <maxLength>3</maxLength>
                <validationPattern>
                    <alphaNumeric exactLength="3"
allowWhitespace="false" allowUnderscore="false" />
                </validationPattern>
                <required>true</required>
                <control>
                    <text size="4" />
                </control>
            </attribute>

     <attributeReference sourceAttributeName="programCode"
name="programCode"
sourceClassName="edu.sample.kuali.module.chart.bo.Program" />

            <attributeReference name="versionNumber"
sourceClassName="org.kuali.core.bo.AttributeReferenceDummy"
sourceAttributeName="versionNumber" />
        </attributes>
    </businessObject>
</dictionaryEntry>
```

ii. Copy the data dictionary file for the base object into the *your* data dictionary directory. **IMPORTANT: Keep the file name the same!**
    - Update the copy of the base object's file to include properties from the extension object.
    - Create attributes as <attributeReference> tags and link to the extension business object (or as appropriate).
    - Add new fields to the inquiry and lookup sections as required.
    - Add relationships if the extension object uses lookups. Relationships must be defined from the location of the base object being edited.
    - Link in the `extension.versionNumber` attribute for object persistence to work properly. (`Account`) even if the relationships also exist on the subordinate object (`AccountExtension`).

edu.sampleu.kuali.module.chart.datadictionary.Account.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dictionaryEntry PUBLIC
    "-//Kuali Project//DTD Data Dictionary 1.0//EN"
    "../../../core/datadictionary/dataDictionary-1_0.dtd">
<dictionaryEntry>
    <businessObject
businessObjectClass="org.kuali.module.chart.bo.Account">
        <inquiry>
            <title>Account Inquiry</title>
            <inquirySections>
             <inquirySection title="Account Details">
                 <inquiryFields>
                 ...
                 <field attributeName="extension.programCode" />
                 </inquiryFields>
         </inquirySection>
                ...
        </inquiry>
        <lookup>
            <lookupFields>
                ....
                <lookupField
attributeName="extension.programCode" required="false" />
                ...
            </lookupFields>

            <resultFields>
                 ...
                <field attributeName="extension.programCode" />
                ...
            </resultFields>
        </lookup>
        <attributes>
            ...
            <attributeReference name="extension.programCode"
sourceClassName="edu.sampleu.kuali.module.chart.bo.AccountExtens
ion" sourceAttributeName="programCode" />
            <attributeReference name="extension.versionNumber"
sourceClassName="org.kuali.core.bo.AttributeReferenceDummy"
sourceAttributeName="versionNumber" />
        </attributes>
        ...
    </businessObject>
</dictionaryEntry>
```

b. Make the maintenance document data dictionary changes.
   - Add attributes to the maintainable fields lists.
   - If new rules are needed, create a new business rule and/or pre-rules class by subclassing the base one's referenced here, then change the references to point to the new rules class(es) you created.
   - Update the default existence checks, if necessary.
   - Add new script files, if needed.

5. Override the appropriate module definition to pull in the files that you have added.

We used "extension" as the name of the OJB reference descriptor and in the property paths in the data dictionary file above, because we are taking advantage of the extension attribute of type `PersistableBusinessObjectExtension` on `PersistableBusinessObject` (the super class of `Account`). This is what allows us to avoid modifying the delivered `Account` business object.

Unable to render {include} The included page could not be found.