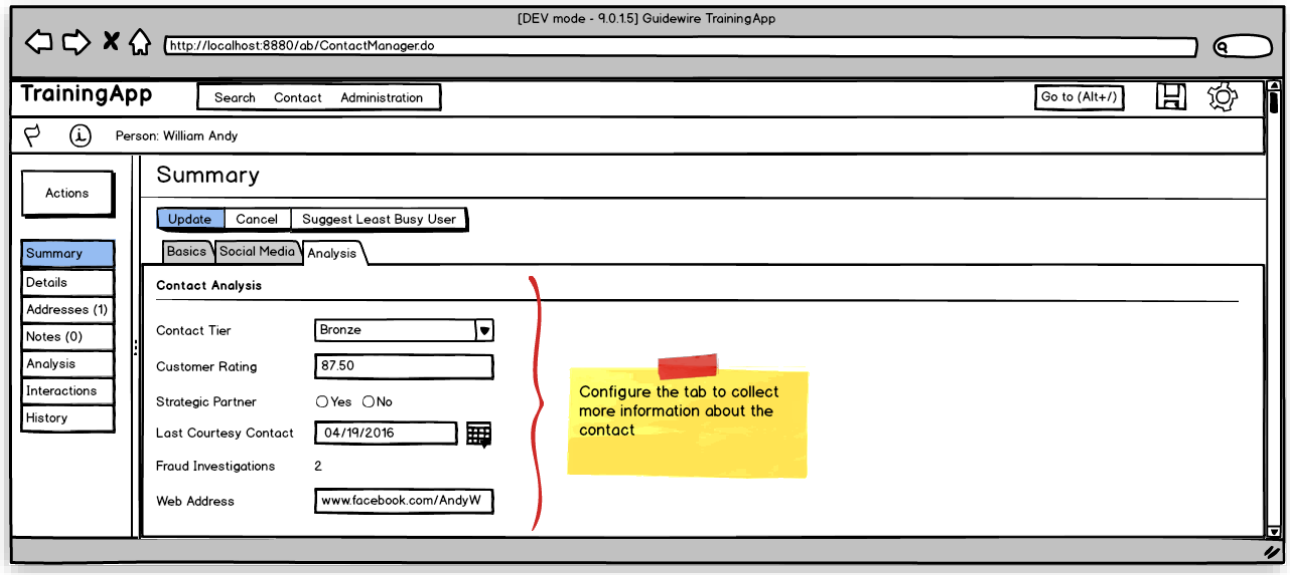
Lesson 3 Extending the Data Model

An insurance company wants to extend TrainingApp functionality by capturing more details about each contact. The complete requirement will be implemented over multiple modules.



In this exercise your job is to make the necessary **data model changes** to meet customer requirements. As a configuration developer, you will use the Entity Editor in Guidewire Studio to modify the TrainingApp data model. You will implement the **user interface changes** later in a later lesson.

3.1 Prerequisites

For this exercise, use TrainingApp, Guidewire Studio, and a supported web browser. `http://localhost:8880/ab/ContactManager.do` is the default URL for TrainingApp. To view, edit, and delete various contacts, log in to TrainingApp as Alice Applegate. The login/password for Alice Applegate is `aapplegate/gw`.

3.2 Lab: Modify an existing Entity Extension

You will use the Entity Editor to modify the existing ABContact.etx Entity Extension and use the column element to add new data fields to the ABContact base application entity.

Review the field definitions received from the business analysts. This table below summarizes the requirements for each field. Note: The Contact Tier field is missing from this list, because it has already been added by one of our fellow configuration developers.

Field name	Datatype	Null ok?
WebAddress_Ext	String of up to 40 characters	true
FraudInvestigationNum_Ext	Integer	true
LastCourtesyContact_Ext	Date (a date and time value)	true
CustomerRating_Ext	A decimal in the format XXX.Y (Values will range from 0.0 to 999.9)	true
IsStrategicPartner_Ext	Bit (a Boolean value)	true

Important!

Read carefully.

For each new entity element, remember to set the nullok attribute to true if not specifically defined to be false. When required, add an element description and specify column parameters.

Best practice

Use _Ext in entity field name

Notice that every field in the table above has an _Ext suffix. For fields that are added to a base application entity, Guidewire recommends that the field name should end with _Ext. This is to prevent potential conflicts during the upgrade to the next version of the Guidewire application.

1. **Open Guidewire Studio**
   a) From the TrainingApp folder, open a command window and start Guidewire Studio for TrainingApp.
   b) Note: Alternatively, use the Start TrainingApp Studio shortcut.
2. **Navigate to ABContact.etx**

3. **Add fields to ABContact.etx to capture additional details**

   a) Add the elements defined in the table below.

Field name	Datatype	Null ok?
WebAddress_Ext	String of up to 40 characters	True
FraudInvestigationNum_Ext	Integer	True
LastCourtesyContact_Ext	Date (a date and time value)	True
CustomerRating_Ext	A decimal in the format XXX.Y (Values will range from 0.0 to 999.9)	True
IsStrategicPartner_Ext	Bit (aBoolean value)	True

4. **Validate your changes in Guidewire Studio and generate the Java class**

   a) Either click on the validate icon in the Entity Editor or use CTRL + S to save your changes

   b) If the code generation was successful (no errors shown in the Codegen tool window) then open the generated Java class and verify you can find your newly added fields

5. **Deploy your changes**

   a) From studio, restart the server using Debug 'Server'.

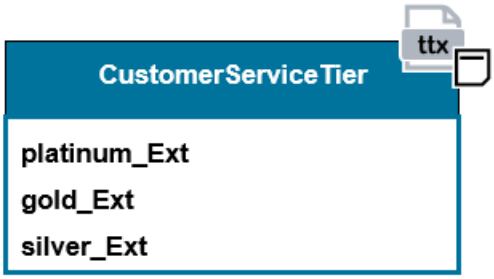
   b) If open, review the Messages Make window for compilation errors.

   c) Review the Debug console for errors.

   d) Verify that the application is running in the Debug console.

3.3 Lab: Extend a base application Typelist

"We also defined new customer service tiers, so we can properly prioritize customer request. Please see the diagram below and configure the Data Model to meet the requirement." – Insurance company business analysts



As a configuration developer, you want to be able to extend base application Typelists so you can add new Typecodes as needed. In this exercise, you will create an extension for the CustomerServiceTier Typelist and use the Typelist Editor to define three new Typecodes.

3.3.1 Configuration

1. **Extend the CustomerServiceTier Typelist**
   a) Create a new TTX file
   b) Define the following Typecodes to display in the order listed using recommended naming conventions:
      - Platinum
      - Gold
      - Silver
2. **Validate your changes in Guidewire Studio and generate the Java class**
   a) Either click on the validate icon in the Typelist Editor or use CTRL + S to save your changes
   b) If the code generation was successful (no errors shown in the Codegen tool window) then open the generated Java class and verify you can find your newly added Typecodes
3. **Deploy your changes**
   a) From studio, restart the server using Debug 'Server'.
   b) If open, review the Messages Make window for compilation errors.
   c) Review the Debug console for errors.
   d) Verify that the application is running in the Debug console.

Best practice

Use intervals when defining priorities

Guidewire recommends using intervals when defining priorities. For example.: 10, 20, 30 This enables you to insert a new value between two existing values without renumbering the other priorities. E.g. you can insert a new Typecode in a later release with the priority 15

Best Practice

Naming typecodes

For typecodes that are added to a base application typelist, Guidewire recommends that the typecode's code should end with _Ext. This is to prevent potential conflicts during the upgrade to the next version of the Guidewire application.

3.3.2 Verification

Activity

Verify the work you have done

Regenerating the data dictionary is not required. Regenerating the data dictionary is a best practice because it creates current documentation for the data model. In addition, the build process for the data dictionary can identify issues beyond schema validation such as referential integrity in the data model.

1. **Build the data dictionary from the command window**
   a) From the bin folder, open a command window.
   b) In the command window, enter the command to build the data dictionary.
2. **Open the data dictionary**
   a) In Windows Explorer, navigate to the data dictionary.
   b) Open the data dictionary using your preferred browser.
3. **View the ABContact entity**
   a) Verify each new field and associated datatype.
4. **View the CustomerServiceTier Typelist**
   a) Verify each new Typecodes.

Best practice

Regenerate the Data Dictionary

Regenerating the data dictionary is not required. Regenerating the data dictionary is a best practice because it creates current documentation for the data model. In addition, the build process for the data dictionary can identify issues beyond schema validation such as referential integrity in the data model.