;C_CurrencyConverter_Design.rtfd;;¬ Previous Section
;E_CurrencyConverter_Implement.rtfd;;¬ Next Section

## 2. Currency Converter Tutorial

# Defining the Classes of Currency Converter

Interface Builder is a versatile tool for application developers. It enables you not only to compose the application's graphical user interface, but it gives you a way to define much of the the *programmatic* interface of the application's classes and to connect the objects eventually created from those classes.

You must go to the Classes display of the nib file window to define a class. Once there, the first thing you must do is select the *superclass*, the class your new *subclass* will inherit from. Let's start with the ConverterController class.

**1 Specify a subclass.**

Go to the Classes display of the nib file window

Select NSObject, the superclass of your custom classes.

Choose Subclass from the pull-down Operations menu.

_IB_PickSuperClass.eps ¬

After you choose the Subclass command, ªMyNSObjectº appears under ªNSObjectº highlighted.

Enter the name of the subclass: ªConverterController.º

Press Return.

_IB_PickSuperClass2.eps ¬

**Related Concept**: ;CurrencyConverterConcepts.rtfd;linkMarkername ClassVersusObject;, Class Versus Object

Now your class is established in the hierarchy of classes within the nib file. Next, specify the paths for messages travelling between the ConverterController object and other objects. In Interface Builder you specify these paths as *outlets* and *actions*.

***TableRule.eps ¬Before You Go On***

Here's some basic terminology:

**Outlet** An object held as an instance variable and typed as **id**. Objects in applications often hold outlets as part of their data so they can send messages to the objects referenced by the outlets. An outlet lets you keep track of or manipulate something in the interface.

**id** The generic (or dynamic) type of objects (technically the address of an object).

**Action** Refers both to a message sent to an object when the user clicks a button or manipulates some other control object and to the method that is invoked.

**Control object** A user-interface object (a device) with which users can interact to affect events in the application. Control objects include buttons, text fields, forms, sliders, and browsers. All control objects inherit from NSControl.
853405_TableRule.eps ¬

See the concepts for the "Travel Advisor Tutorial" for more on control objects and their relation to cells and formatters.

**2 Define your class's outlets.**

In the nib file window, click the electrical-outlet icon to the right of the class.

Choose Add Outlet from the Operations pull-down menu
Type the name of the outlet over the highlighted ªmyOutlet.º Name the first outlet rateField.
Press Return.

_IB_Outlets2.eps ¬

Repeat the last three steps to define two other outlets:
**€€€€dollarField**
**€€€€totalField**

IB_Outlets3.tiff ¬

ConverterController has one action method, **convert:**. When the user clicks the Convert button, a **convert:** message is sent to the target object, an instance of ConverterController.

**3 Define your class's actions.**

In the Classes display of the nib file window, click the crosshairs icon.
Choose the Add Action command from the Operations pull-down menu.
Type the name of the action method, convert:.
Press Return.

_IB_Actions1.eps ¬

***787360_TableRule.eps ¬Before You Go On***

**Add an outlet:** ConverterController needs to access the text fields of the interface, so you've just provided outlets for that purpose. But ConverterController must also communicate with the Converter class (yet to be defined). To enable this communication, add an outlet named **converter** to ConverterController.
95032_TableRule.eps ¬
**Related Concept**:

;CurrencyConverterConcepts.rtfd;PathsforObjectCommunication:Outlets,Targets,andActions;¬ Paths for Object Communication: Outlets, Targets, and Actions

## Connecting ConverterController to the Interface

As the final step of defining a class in Interface Builder, you create an instance of your class and connect its outlets and actions.

**4 Generate an instance of the class.**

In the Classes display, select the ConverterController class.

Choose the Instantiate command from the Operations pull-down menu.


**Note:** The Instantiate command does not generate a true instance of ConverterController, but creates a stand-in object used for establishing connections. When the nib file's contents are unarchived, Interface Builder will create true instances of these classes and use the proxy objects to establish the outlet and action connections.

When you instantiate a class (that is, create an instance of it), Interface Builder switches to the Instances display and highlights the new instance, which is named after the class.


Now you can connect this ConverterController object to the user interface. By connecting it to specific objects in the interface, you initialize your outlets. ConverterController will use these outlets to get and set values in the interface.

**5 Connect the custom class to the interface via its outlets.**

In the Instances display of the nib file window, Control-drag a connection line from the ConverterController instance to the first text field.

When the field is outlined in black, release the mouse button.

_IB_ConnectOutlet_Comp.eps ¬

Interface Builder brings up the Connections display of the Inspector panel. This display shows the outlets you have defined for ConverterController.

In the Connections display, select the outlet that corresponds to the first field (**rateField**).

Click the Connect button.

Following the same steps, connect ConverterController's **dollarField** and **totalField** outlets to the appropriate text fields.

_IB_ConnectOutletInspect.eps ¬

To receive action messages from the user interfaceÐto be notified, for example, when users click a buttonÐyou must connect the control objects that emit those messages to CurrencyConverter. The procedure for connecting actions is similar to that for outlets, but with one major difference. When you connect an action, always start the connection line from a *control object* (such as a button, text field, or form) that sends an action message; you usually end the connection at an instance of your custom class. That instance is the *target* outlet of the control object.

**6 Connect the interface's controls to the custom class via its actions.**

Control-drag a connection line from the Convert button to the ConverterController instance in the nib file window.

When the instance is outlned in black, release the mouse button..

_IB_ConnectAction_Comp.eps ¬

The Connections display of the Inspector panel shows the action methods you have specified for ConverterController.

In the Connections display, make sure **target** in the Outlets column is selected.
Select **convert:** in the Actions column.
Click the Connect button.
Save the CurrencyConverter nib file (Document *arrow.eps* ¬ Save).

_IB_ConnectActionInspect.eps ¬

You've finished defining the classes of Currency ConverterÐalmost.

### *2149_TableRule.eps ¬Before You Go On*

**Define the Converter Class:** While connecting ConverterController's outlets, you probably noticed that one outlet remains unconnected: **converter**. This outlet identifies the instance of the Converter class in the Currency Converter application, which doesn't exist yet.

Define the Converter class. This should be pretty easy because Converter, as you might recall, is a model class within the Model-View-Controller paradigm. Since instances of this type of class don't communicate directly with the interface, there is no need for outlets or actions. Here are the steps to be completed:

1. In the Classes display, make Converter a subclass of NSObject.
2. Instantiate the Converter class.
3. Make an outlet connection between ConverterController and Converter.

When you are finished, save **CurrencyConverter.nib**.

325515_TableRule.eps ¬

### *736553_TableRule.eps ¬Optional Exercise*

**Text fields and action messages:** The **NSReturnsign** image that you embedded earlier in the Convert button indicates that users can activate this button by pressing the Return key. In Currency Converter this key event occurs when the cursor is in a text field. Text fields are control

objects just as buttons are; when the user presses the Return key and the cursor is in a text field, an action message is sent to a target object if the action is defined and the proper connection is made.

Connect the second text field (that is, the one with the ªDollars to Convertº label) to the **convert:** action method of ConverterController. You won't be disconnecting the prior action connection because multiple control objects in an interface can invoke the same action method. 731580_TableRule.eps ¬