Price Models

This document will show you how to create a new Pricing Model. The chosen example is a custom Pricing Model that lets you set a percentage value in which you would like the price for an item to be incremented depending on the country. If you create an order for a customer in the US, you have the ability to set the price to be incremented by 10%, Canada by 15%. You also have the option to create default values.

In order to create this custom Pricing Model, let’s review the 3 things that need to be done to create your own Pricing Strategy:

1. Create your Pricing Strategy class that will contain the logic.
2. Add your new Price Model to the Pricing Strategies enum.
3. Create a template.

# Create your Pricing Strategy class with the required logic

To do this you have to create a new class that extends from *AbstractPricingStrategy*. The name of your class should be in the form of ***{Name of your strategy}****PricingStrategy*. In our case it will be *CountryPricingStrategy*.

In this class you can do 3 things at the beginning in the constructor:

* Set attributes
* Set the chain position
* Set If the usage is required

Setting attributes

Every attribute that is configured here is going to appear in the Pricing Model template so that you can set a value to it.

To do this you need to add this line inside the constructor:

setAttributeDefinitions**();**

It receives an *AttributeDefinition* object as a parameter. This object has 3 parameters. The first is the name of the attribute, the second is the type, and the third is a boolean stating whether it’s required or not.

For our example I created the following:

private static final String DEFAULT\_PERCENTAGE **=** "default"**;**

public CountryPricingStrategy**()** **{**

setAttributeDefinitions**(**

**new** AttributeDefinition**(**DEFAULT\_PERCENTAGE**,** STRING**,** **true)** // percentage to other countries that are not set in the price model configuration

**);**

**}**

You can add as many attributes as you need.

Setting chain position

This means that you can actually define which position this Pricing Model is allowed to be configured in a Product. The available values are START, MIDDLE and END. To do this you call this method:

setChainPositions**();**

Inside, you put a list of *ChainPosition* elements with the positions you desire. In our case:

setChainPositions**(**

ChainPosition**.**START**,**

ChainPosition**.**MIDDLE**,**

ChainPosition**.**END

**);**

Setting the usage

You have the ability to set if the usage data is required or not. This can save some time. The reason for this is that before the logic is processed in the Pricing Model, the system needs to gather all the usage information. To do this, execute this method:

setRequiresUsage**();**

It receives a boolean as a parameter. You will need to send true or false. For us:

setRequiresUsage**(false);**

Therefore, our constructor looks like:

public CountryPricingStrategy**()** **{**

setAttributeDefinitions**(**

**new** AttributeDefinition**(**DEFAULT\_PERCENTAGE**,** STRING**,** **true)** // percentage to other countries that are not set in the price model configuration

**);**

setChainPositions**(**

ChainPosition**.**START**,**

ChainPosition**.**MIDDLE**,**

ChainPosition**.**END

**);**

setRequiresUsage**(false);**

**}**

What we are saying with this, is that we want an attribute called, default, of type String, and it’s required. Then we say that our Price Strategy can be added in any position of the chain. Finally, we set the usage to false because we don’t require the usage data for our logic.

In order to complete the implementation of our custom Pricing Model we need to implement the *applyTo* method and put our logic in there:

public void applyTo**(**OrderDTO pricingOrder**,** PricingResult result**,** List**<**PricingField**>** fields**,**

PriceModelDTO planPrice**,** BigDecimal quantity**,** Usage usage**,** boolean singlePurchase**)** **{**

**...**

**}**

For our example attach the java class to review the full code.

# Add an entry in the enum

The next step is to add an entry to the *PriceModelStrategy* enum so that jBilling knows that this Pricing Model exists. In our case it looks like:

/\*\* Pricing strategy that increases the price depending on the country.\*/

COUNTRY **(new** CountryPricingStrategy**());**

# Create the template

Finally, to complete our implementation we need to add a template for the view part of our new Pricing Model. The name has to be the camel case for the entry that we added in the enum. If we have for example, NEW\_PRICE\_MODEL, the name of the template should be *\_newPriceModel.gsp*. In our case it’s, *\_country.gsp*. Once you create it put it in this folder: */grails-app/views/priceModel/strategy/*.

Inside you just need to copy and paste the contents of the other template. If your strategy doesn’t need the rate and currency fields, then use the content of the flat Pricing Model. Otherwise, you can use any of the other templates.

The file will also be attached.

# i18n

The last thing you need to do is add one line in the messages.properties file so that the name of the Pricing Model appears nicely in the drop down when you need to select the desired strategy. Look for this text in the file *price.strategy.FLAT=Flat*  and add it at the bottom after all the internationalizations for the other strategies. In our case it’s, *price.strategy.COUNTRY=Country*

This is how it will looks when we use the GUI:



Summary

Here’s a summary of all the files that were modified/created in order to complete this exercise.

|  |  |  |
| --- | --- | --- |
| File | Updated or Created? | Description |
| *src/java/com/sapienter/jbilling/server/pricing/strategy/ CountryPricingStrategy.java* | Created | Class with all the logic for calculating the price for this new Price Model. |
| *grails-app/views/priceModel/strategy/ \_country.gsp* | Created | Template for the Pricing Strategy. |
| *src/java/com/sapienter/jbilling/server/pricing/db/ PriceModelStrategy.java* | Updated | Added one entry so that the system knows about our new Price Model. |
| *grails-app/i18n/ messages.properties* | Updated | Added one line with the i18n for the Price Models drop down. |