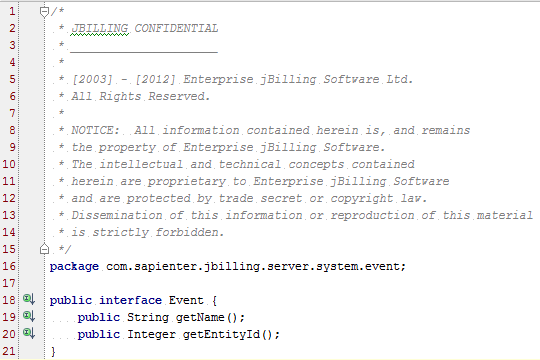
Internal Events

This document will show you how to add Internal Events into jBilling. By following an example from beginning to end, you will learn how to add a new Internal Event. In our example, you are going to create two events; one triggered when a customer is updated, and the other for when a new customer is created. The Listener class will catch these events and create a new order for this customer. This example or use case may not be useful in the future. The idea here is to understand the knowledge, which will enable you to create internal events and then do something with them.

There are 5 things that you need to do in order to be able to create a new event and get it to work:

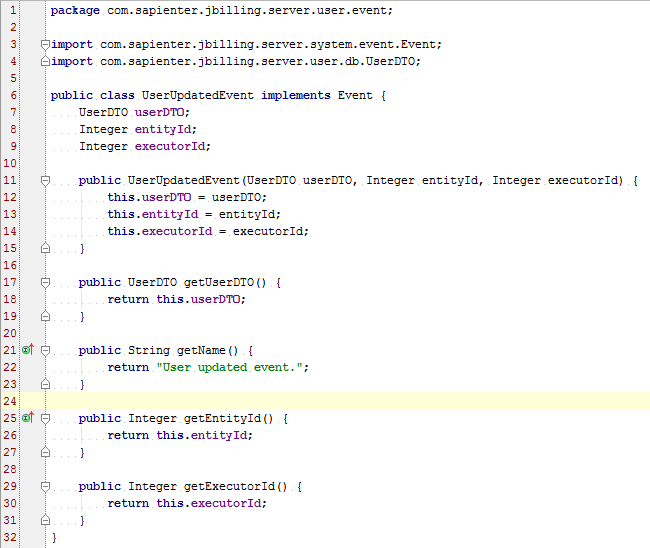
1. Create the event
2. Create the listener class
3. Configure the Listener class in the Plugins section
4. Trigger the event
5. DB queries
6. **Create the event**

To create the event you need a new class that implements the *Event.java* interface:

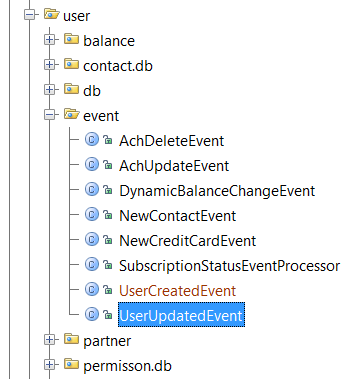


In this new class, besides implementing the required methods through the interface, you can add any other object or data that may be useful once the event is caught by the listener class. For our example, we are creating two events, one for when a customer is updated, and the other for when a new customer is created.

Event for customer update (class *UserUpdatedEvent.java*):



Usually the event classes are saved in a folder named *event* inside the entity’s folder under src.

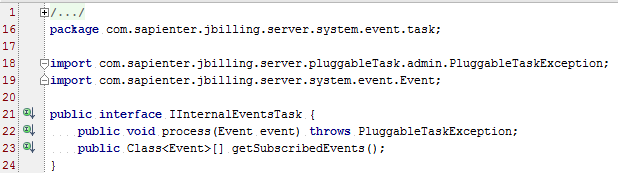


You can see in our class that aside from the interface’s methods we also added a UserDTO object. The reason for this is because we are going to need it in the listener class to create the order. The entity and executor id are also needed for this task. In summary, the event is a simple class that implements the Event interface with attributes, a constructor, and some getters.

1. **Create the listener class**

Once you have your events in place, you need to create the listener class to catch these events when they are triggered, and do some processing. You may already know that in an event-listener scenario, the listener class can listen to one or more events at the same time. In this example, you will complete that task to understand how it’s usually handled (the first scenario is simpler if you already know how to do this).

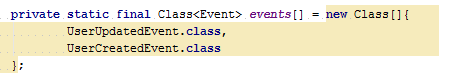
This class needs to extend from *PluggableTask.java* and implement the *IInternalEventsTask.java* interface:



Both methods from the interface are really important to understand. The first one is:

*public Class****<****Event****>[]*** *getSubscribedEvents****()******{}***

It returns a list with the Events that this listener class listens to. Because of this, you may have already guessed that we need an array with the events for this class. To do that you need to add the following code:

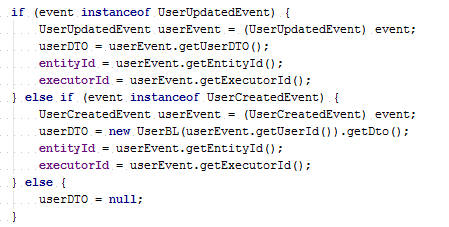


As you can see, here we have the two events that this document has mentioned. It’s easy to understand that for a scenario where we listen to only one event, you would only add one to the array.

The next important step is to implement the process method because it is where all the logic lies:

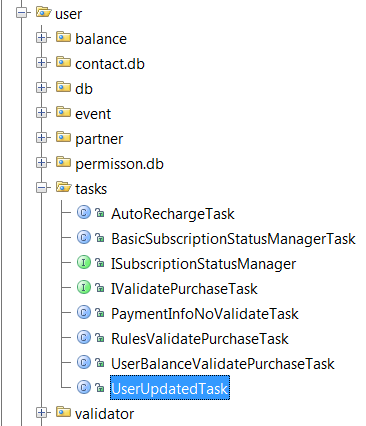
*public void process****(****Event event****)******throws*** *PluggableTaskException* ***{}***

As you can see, it receives an Event as a parameter. This would be the triggered event. However, how do you know which event we are talking about if we listen to more than one? The typical way of handling this is to ask, what type is the Event using the *instanceof* operator? Once you know the type you can cast the Event to it and use the getter to get the necessary data.



What follows next is the logic itself. It may vary from implementation to implementation.

Usually, the Listener task is put in a task folder under the entity’s path in src:

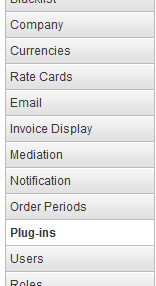


1. **Configure the Listener class in the Plugins section**

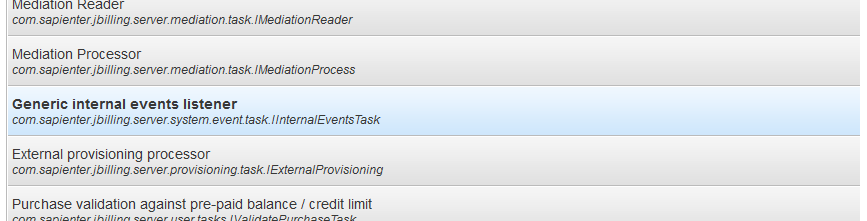
To configure the listener class, in jBilling, you need to go to the Configuration section in the top menu:

C:\Users\jmvidal\Desktop\menu.png

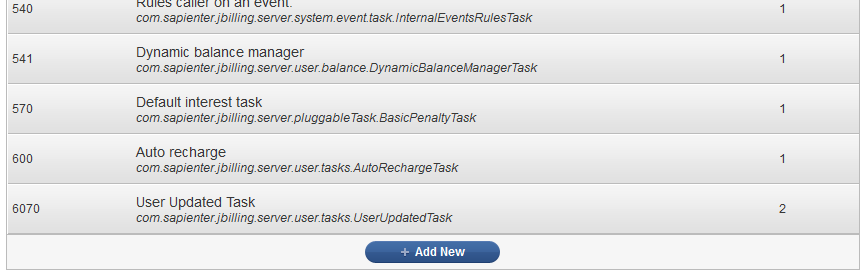
Then click on the Plug-ins link in the left menu to see the list of plug-ins:



Scroll down and select the *Generic internal events listener* category:



Scroll back to the top of the screen, where you can add a new plugin, by clicking on the ‘Add New’ button. Select the plug-in – *UserUpdatedTask* and save it:



1. **Trigger the event**

After creating the event and the listener class, is important to trigger the event by telling it when you want it to occur. To be able to do this, you need to execute one line of code at the point where you need the trigger to be. For our example we have two triggers, one in the *createUser* method in the *WebServicesSessionSpringBean* class, which is where the new customer is created. The other one is in the *updateUser* method in the same class, but here the user is updated. The line of code that you need to run is:

*EventManager.process(****Event****)*

The event that you pass as parameter is the event that you want to trigger. In this example these are the two triggers that we have:

*EventManager****.****process****(new*** *UserCreatedEvent****(****userId****,*** *entityId****,*** *getCallerId****()));***

*EventManager****.****process****(new*** *UserUpdatedEvent****(****bl****.****getDto****(),*** *entityId****,*** *executorId****));***

That is all you need to do. You can see that in the constructor you are passing all the necessary data to instantiate the event. This data will be used later by the listener class.

1. **DB queries**

Finally, there are a couple of queries that you need to run in the DB.

* Add the Listener task to the plug-ins list:

***insert******into*** *pluggable\_task\_type* ***(****id****,*** *category\_id****,*** *class\_name****,*** *min\_parameters****)******values******(****97****,*** *17****,*** *'com.sapienter.jbilling.server.user.tasks.UserUpdatedTask'****,*** *0****);***

* Add i18n for the task’s title and description:

***insert******into*** *international\_description* ***(****table\_id****,*** *foreign\_id****,*** *psudo\_column****,*** *language\_id****,*** *content****)******values******(****24****,*** *97****,*** *'title'****,*** *1****,*** *'User Updated Task'****);***

***insert******into*** *international\_description* ***(****table\_id****,*** *foreign\_id****,*** *psudo\_column****,*** *language\_id****,*** *content****)******values******(****24****,*** *97****,*** *'description'****,*** *1****,*** *'Adds a new item to the last order of the customer when it''s updated..'****);***

Summary

Here is 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/user/event/UserUpdatedEvent.java* | Created | New event triggered when a customer is updated. |
| *src/java/com/sapienter/jbilling/server/user/event/UserCreatedEvent.java* | Created | New event triggered when a customer is created. |
| *src/java/com/sapienter/jbilling/server/user/tasks/UserUpdatedTask.java* | Created | Listener class. |
| *src/java/com/sapienter/jbilling/server/util/WebServicesSessionSpringBean.java* | Updated | Added the two event triggers at the end of the *createUser* and *updateUser* methods |