Event Module

What this module does

This module creates easy handle points for other modules to hook onto. When an event occurs in openmrs it will notify all registered/subscribed listeners.

Downloads

View Source: https://github.com/openmrs/openmrs-module-event
Checkout Source: https://github.com/openmrs/openmrs-module-event
Download: https://addons.openmrs.org/#/show/org.openmrs.module.event

Technical Documentation

Possible Actions

An enum on Event.Action contains the possible actions of CREATED, UPDATED, RETIRED, UNRETIRED, VOIDED, UNVOIDED, PURGED

Registering module domain objects for which to fire events

The module's domain objects must be subclasses of OpenmrsObject.
If you wish for retired/unretired or voided/unvoided events to be fired, they should implement Retireable or Voidable respectively.

To subscribe to an event

In your module activator or anywhere else in your module code:

```
Event.subscribe(Class, String, EventListener); // The String argument can be any of the values in the Event.
Action Enum
or
Event.setSubscription(SubscribableEventListener); // spring callable, see below
```

or in your Spring moduleApplicationContext:

```
<bean class="org.openmrs.eventbus.Event">
   <property name="subscription"><bean class="org.your.module.package.SubscribableEventListenerImpl"></bean><
/property>
</bean>
```

A **SubscribableEventListener** specifies the list of Classes and list of Actions that it supports.

Getting notified of an event.

The module uses apache's ActiveMQ messaging server, your EventListener class should extend EventListener and its **onMessage(Message message)** wi
ll be called

Unsubscribing from an event

```
Event.unsubscribe(Class, Action, EventListener);
or
Event.unsetSubscription(SubscribableEventListener); // just because its cute
```

Example Client code

- The Atom Feed Module takes advantage of this module.
- The below class should be registered to the Event class with Event.subscribe(____, new CountEventListener());

```
// A simple event listener that just keeps a count of all created, updated and purged items
public class CountEventListener implements EventListener {

        private int createdCount = 0;

        private int updatedCount = 0;

        private int purgedCount = 0;

        public int getCreatedCount() {
                return createdCount;
        }

        public void getUpdatedCount() {
                return updatedCount;
        }

        public int getPurgedCount() {
                return purgedCount;
        }

        @Override
        public void onMessage(Message message) {
                try {
                        MapMessage mapMessage = (MapMessage) message;
                        if (Action.CREATED.toString().equals(mapMessage.getString("action")))
                                createdCount++;
                        else if (Action.UPDATED.toString().equals(mapMessage.getString("action")))
                                updatedCount++;
                        else if (Action.PURGED.toString().equals(mapMessage.getString("action")))
                                purgedCount++;


                        //..... Keep counts for more event actions
                }
                catch (JMSException e) {
                        System.out.println("Ooops! some error occurred");
                }
        }

}
```

Release Notes

2.2.1

- Fixes the issue of OpenMRS hanging on startup due to incorrect activemq-data location (EVNT-31)

2.2 (broken for bean subscription, please use 2.2.1)

2.0.1

- Changed module so that it works via Hibernate Interceptors instead of AOP (EVNT-22)

1.2

- Handle saving of subclasses (EVNT-23)
- Added person to list of supported classes (EVNT-24)

1.1

- Handle events on Orders (EVNT-21)

1.0

- To be written after release