CubiEventObserver
Implement Event Observer in Openbiz Cubi (beta)
Phase-Design

Updated Mar 29, 2013 by rockys...@gmail.com

Introduction

In latest Openbiz framework, EventManager is a component designed for the following use cases:

- Implementing simple subject/observer patterns.
- Implementing Aspect-Oriented designs.
- Implementing event-driven architectures.

The basic architecture allows you to attach and detach observers to named events, trigger events, and interrupt execution of observers.

Basic Event Support in Openbiz Framework

There are 3 new interfaces defined in the framework.

```
interface iEventManager
{
        public function trigger($event_key, $target, $params);
        public function attach($event_key, $observer, $priority=null);
}

interface iEvent
{
        public function getName();
        public function getTarget();
        public function getParams();
}

interface iEventObserver
{
        public function observe($event);
}
```

And Openbiz includes very basic implementations of these interfaces.

- class EventManager implements iEventManager
- class Event implements iEvent
- class EventObserver implements iEventObserver

Trigger an Event

An EventManager is really only interesting if it triggers some events. Basic triggering takes three arguments: the event name, which is usually the current function/method name; the “context”, which is usually the current object instance; and the arguments, which are usually the arguments provided to the current function/method.

```
class UserDO
{
    // ... assume events definition from above

    public function deleteRecord($recordId)
    {
        $params = array('Id'=>$recordId);
        $this->getEventManager()->trigger(__FUNCTION__.'post', $this, $params);
    }
}
```

Observe an Event

In turn, triggering events is only interesting if something is observing (listening) for the event. Observer attach to the EventManager, specifying a named event and pass its own object. The observers's observe method will be called with $event object passed in as an argument.

```
$observer = new MyEventObserver();
$user = new UserDO();
$user->getEventManager()->attach('deleteRecord.post', $observer);

class MyEventObserver extends EventObserver
{
    public function observe($event)
    {
        $params = $event->getParams();
        BizSystem::log(LOG_DEBUG, $event->getName(), "Deleted Id is ".$params['Id']);
    }
}
```

Default Triggered Events

DataObject triggers several events including:

- deleteRecord.pre
- deleteRecord.post
- updateRecord.pre
- updateRecord.post
- insertRecord.pre
- insertRecord.post

Event Manager Module in Cubi

Cubi makes hooking in custom event observer much easier by introducing eventmgr module. The Event Manager module manages the mapping between named events and event observers. Cubi sets a class in this module "CubiEventManager" (/eventmgr/lib/CubiEventManager.php) as default system event manager class in cubi/bin/app_init.php

```
define('EVENT_MANAGER', "eventmgr.lib.CubiEventManager");
```

Define Event Observer

An Event Observer is just a normal Openbiz plugin service as long as the implementation class has public method "observe". The eventmgr module comes with a sample Event Observer at eventmgr/sample/EventLogger.xml.

```
<?xml version="1.0" standalone="no"?>
<PluginService Name="EventLogger" Class="EventLogger">
</PluginService>
```

The implementation class is eventmgr/sample/EventLogger.php. It simply invoke Cubi EventLog service to log an event message.

```
class EventLogger
{
    public function observe($event)
    {
        $triggerObj = $event->getTarget();
        $triggerEvent = $event->getName();
        $params = $event->getParams();
        // get eventlog service
        $eventLog = BizSystem::getService(EVENTLOG_SERVICE);
        // log message
        $eventLog->Log($triggerEvent,"triggered by ".$triggerObj->m_Name);
    }
}
```

Link Event Observer to Named Events

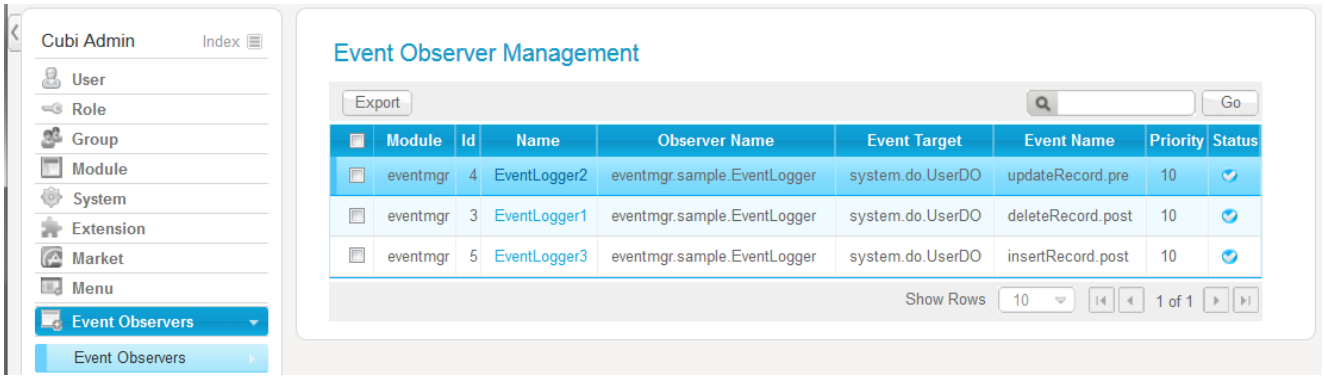
You can link your event observer to any name event triggered in the system (e.g. the "pre" and "post" events triggered from DataObject. Such mapping is defined in mod.xml EventObservers section.

```
<EventObservers>
    <Observer Name="EventLogger1" ObserverName="eventmgr.sample.EventLogger" EventTarget="system.do.UserDO" EventName="del
    <Observer Name="EventLogger2" ObserverName="eventmgr.sample.EventLogger" EventTarget="system.do.UserDO" EventName="upd
    <Observer Name="EventLogger3" ObserverName="eventmgr.sample.EventLogger" EventTarget="system.do.UserDO" EventName="ins
</EventObservers>
```

Once the module is loaded into Cubi, CubiEventManager will call the right event observers' observe method.

View the Cubi Event Observers

Application administrator can find all event observers in "Event Observer" view.



You can disable and enable an event observer by toggling the status icon.