uScaffold

What is it?

uScaffold is a simple Magento extension that was created for personal use to assist with rapid prototyping of custom Magento extensions.

It will attempt to generate a standard Magento configuration based on simple hints from custom extension declaration XML file.

This extension is geared towards developers, who are learning Magento or just wish to speed up their extension development.

How do I get it?

Download the archive from this link and unpack into Magento root folder: http://download.unirgy.com/Unirgy_Scaffold-latest.zip

Disable or refresh cache as needed.

How does it work? (by example)

Initialize extension with uScaffold support

After installation of the extension, create a new file as you would normally do for a custom extension:

app/etc/modules/Custom_Extension.xml

```
<?xml version="1.0"?>
<config>
  <modules>
    <Custom_Extension>
      <active>true</active>
      <codePool>local</codePool>

      <!-- THIS PART IS CUSTOM FOR SCAFFOLDING -->
      <scaffold code="customext"/>
      <!-- END SCAFFOLDING -->

    </Custom_Extension>
  </modules>
</config>
```

After you save, uScaffold will generate basic configuration for the module, which encompasses models, helpers and blocks. Now you can just create PHP files as you usually would do for magento, without worrying about the configuration for them. Take note: You DO NOT need to create `app/code/local/Custom/Extension/etc/config.xml`.

app/code/local/Custom/Extension/Model/Test.php

```
<?php

class Custom_Extension_Model_Test extends Varien_Object
{
}
```

The models, helpers and blocks are referenced by the `code` attribute, in this case `customext`.

```
<?php

$model = Mage::getModel('customext/test');
$helper = Mage::helper('customext');
```

```
<layout>
  <default>
    <reference name="content">
      <block type="customext/your_block"/>
    </reference>
  </default>
</layout>
```

If you do not have any special logic for helper, you do not need to create Helper/Data.php file, and still use <?php `echo $this→__('…')` ?> in your blocks.

Working with DB tables

Your module needs DB tables support? No problem.

app/etc/modules/Custom_Extension.xml

```
<?xml version="1.0"?>
...
      <!-- THIS PART IS CUSTOM FOR SCAFFOLDING -->
      <scaffold code="customext">
        <tables>
          <test1/>
          <test2 table="custom_test"/>
        </tables>
      </scaffold>
      <!-- END SCAFFOLDING -->
```

```
...
```

All the configuration for setup, read and write connections are made automatically, using core DB connections. If `table` attribute is not specified, the table name will be the same as table entity code (`test1`).

Event observers

app/etc/modules/Custom_Extension.xml

```
...
      <!-- THIS PART IS CUSTOM FOR SCAFFOLDING -->
      <scaffold code="customext">
        <observers>
          <customer_load_after/>
          <sales_order_save_before/>
        </observers>
      </scaffold>
      <!-- END SCAFFOLDING -->
...
```

Doing the above will generate configuration to trigger an observer from your PHP model. All you have to do is to add this file:

app/code/local/Custom/Extension/Model/Observer.php

```
<?php
class Custom_Extension_Model_Observer
{
  public function customer_load_after($observer)
  {
  //...
  }

  public function sales_order_save_before($observer)
  {
  //...
  }
}
```

Frontend development

app/etc/modules/Custom_Extension.xml

```
...
      <!-- THIS PART IS CUSTOM FOR SCAFFOLDING -->
      <scaffold code="customext">
        <frontend/>
      </scaffold>
      <!-- END SCAFFOLDING -->
...
```

BAM! layout updates are loaded from `layouts/customext.xml` and translations are loaded from `Custom_Extension.csv`

Do you need to run controllers for your module?

app/etc/modules/Custom_Extension.xml

```
...
      <!-- THIS PART IS CUSTOM FOR SCAFFOLDING -->
      <scaffold code="customext">
        <frontend>
          <controllers/>
        </frontend>
      </scaffold>
      <!-- END SCAFFOLDING -->
...
```

Frontend only events?

app/etc/modules/Custom_Extension.xml

```
...
      <!-- THIS PART IS CUSTOM FOR SCAFFOLDING -->
      <scaffold code="customext">
        <frontend>
          <observers>
            <customer_load_after/>
            <sales_order_save_before/>
          </observers>
        </frontend>
      </scaffold>
      <!-- END SCAFFOLDING -->
...
```

NOTE: frontend specific observer method should start with FRONTEND_:

```
public function FRONTEND_customer_load_after($observer)
{
//...
}
```

Admin development

Similar to frontend:

app/etc/modules/Custom_Extension.xml

```
...
      <!-- THIS PART IS CUSTOM FOR SCAFFOLDING -->
      <scaffold code="customext">
        <adminhtml>
          <controllers/>
          <observers>
            <customer_load_after/>
            <sales_order_save_before/>
          </observers>
        </adminhtml>
      </scaffold>
      <!-- END SCAFFOLDING -->
...
```

NOTE: admin specific observer method should start with `ADMIN_`:

```
public function ADMIN_customer_load_after($observer)
{
//...
}
```

But wait, there's more!

app/etc/modules/Custom_Extension.xml

```
...
      <!-- THIS PART IS CUSTOM FOR SCAFFOLDING -->
      <scaffold code="customext">
        <adminhtml>
          <nav>
            <system>
              <test title="Test" sort_order="10"
action="sometestadmin/adminhtml_test/index"/>
            </system>
          </nav>
          <config>
            <test title="Test"/>
          </config>
        </adminhtml>
      </scaffold>
      <!-- END SCAFFOLDING -->
...
```

The code above will create admin menu entry and related ACL entry.

Moving from scaffolding to production

The uScaffold is not made for the production deployment, and as such is not recommended to be bundled with your extensions. To retrieve automatically generated configuration, add `output` attribute to your scaffold declaration:

app/etc/modules/Custom_Extension.xml

```
...
      <!-- THIS PART IS CUSTOM FOR SCAFFOLDING -->
      <scaffold code="customext" output="true">
        <!-- ... -->
      </scaffold>
      <!-- END SCAFFOLDING -->
...
```

Refresh the page and you'll find `var/uscaffold/Custom_Extension/config.xml` file, containing the full Magento config file.

Just copy it to `app/code/local/Custom/Extension/etc/config.xml`, and you're ready for production!

Wishlist

- etc/system.xml scaffolding
- Automatic emulation of basic model classes (Model/*, Model/Mysql4/*, Model/Mysql4/Collection/*)

FAQ

Why the method names for event observers are not in Zend convention?

In my personal experience it is much easier to maintain event observers and method names when they're the same. Usually they're in 1-to-1 relationship and making them different names only adds to complexity

Why not to use admin routers as additional modules?

There are still many stores that run Magento 1.3.x, where admin router modules are not available. I was trying to make the scaffold as compatible as possible. Most probably, in future versions it will be replaced with admin router modules or added as a configuration option.

Example uScaffold configuration vs full Magento configuration

uScaffold configuration

app/etc/modules/Custom_Extension.xml

```
<?xml version="1.0"?>
<config>
    <modules>
        <Custom_Extension>
            <active>true</active>
            <codePool>local</codePool>

            <scaffold code="customext" output="true">
                <tables>
                    <test table="sometest_test"/>
                </tables>
                <frontend>
                    <controllers/>
                    <observers>
                        <checkout_cart_add/>
                    </observers>
                </frontend>
                <adminhtml>
                    <controllers/>
                    <nav>
                        <system>
                            <test title="Test" sort_order="10"
action="sometestadmin/adminhtml_test/index"/>
                        </system>
                    </nav>
                    <config>
                        <test title="Test"/>
                    </config>
                </adminhtml>
                <observers>
                    <customer_load_after/>
                    <sales_order_save_before/>
                </observers>
            </scaffold>
        </Custom_Extension>
    </modules>
</config>
```

Automatically generated full Magento configuration

var/uscaffold/Custom_Extension/config.xml

```
<?xml version="1.0"?>
<config>
   <global>
      <models>
         <customext>
            <class>Custom_Extension_Model</class>
            <resourceModel>customext_mysql4</resourceModel>
         </customext>
         <customext_mysql4>
            <class>Custom_Extension_Model_Mysql4</class>
            <entities>
               <test>
                  <table>sometest_test</table>
               </test>
            </entities>
         </customext_mysql4>
      </models>
      <helpers>
         <customext>
            <class>Custom_Extension_Helper</class>
            <rewrite>
               <data>Mage_Core_Helper_Data</data>
            </rewrite>
         </customext>
      </helpers>
      <blocks>
         <customext>
            <class>Custom_Extension_Block</class>
         </customext>
      </blocks>
      <resources>
         <customext_setup>
            <setup>
               <module>Custom_Extension</module>
            </setup>
            <connection>
               <use>core_setup</use>
            </connection>
         </customext_setup>
         <customext_write>
            <connection>
               <use>core_write</use>
            </connection>
         </customext_write>
         <customext_read>
            <connection>
               <use>core_read</use>
            </connection>
         </customext_read>
      </resources>
      <events>
         <customer_load_after>
            <observers>
               <customext>
                  <type>singleton</type>
                  <model>customext/observer</model>
                  <method>customer_load_after</method>
               </customext>
            </observers>
         </customer_load_after>
         <sales_order_save_before>
            <observers>
               <customext>
                  <type>singleton</type>
                  <model>customext/observer</model>
                  <method>sales_order_save_before</method>
               </customext>
            </observers>
         </sales_order_save_before>
      </events>
   </global>
   <frontend>
      <layout>
         <updates>
            <customext>
               <file>customext.xml</ file>
            </customext>
         </updates>
      </layout>
      <translate>
         <modules>
            <Custom_Extension>
               <files>
                  <default>Custom_Extension.csv</default>
               </files>
            </Custom_Extension>
         </modules>
      </translate>
      <routers>
         <customext>
```

```
            <use>standard</use>
            <args>
               <module>Custom_Extension</module>
               <frontName>customext</frontName>
            </args>
         </customext>
      </routers>
      <events>
         <checkout_cart_add>
            <observers>
               <customext>
                  <type>singleton</type>
                  <model>customext/observer</model>
                  <method>checkout_cart_add</method>
               </customext>
            </observers>
         </checkout_cart_add>
      </events>
   </frontend>
   <adminhtml>
      <layout>
         <updates>
            <customext>
               <file>customext.xml</ file>
            </customext>
         </updates>
      </layout>
      <translate>
         <modules>
            <Custom_Extension>
               <files>
                  <default>Custom_Extension.csv</default>
               </files>
            </Custom_Extension>
         </modules>
      </translate>
      <menu>
         <system>
            <children>
               <test>
                  <title>Test</title>
                  <sort_order>10</sort_order>
                  <action>sometestadmin/adminhtml_test/index</action>
                  <children/>
               </test>
            </children>
         </system>
      </menu>
      <acl>
         <resources>
            <admin>
```

```
                <children>
                    <system>
                        <children>
                            <test>
                                <title>Test</title>
                                <sort_order>10</sort_order>
                                <children/>
                            </test>
                            <config>
                                <children>
                                    <test>
                                        <title>Test</title>
                                    </test>
                                </children>
                            </config>
                        </children>
                    </system>
                </children>
            </admin>
        </resources>
    </acl>
</adminhtml>
<admin>
    <routers>
        <customextadmin>
            <use>admin</use>
            <args>
                <module>Custom_Extension</module>
                <frontName>customextadmin</frontName>
            </args>
        </customextadmin>
    </routers>
</admin>
</config>
```

CHANGELOG

0.6.0

- Fixed copy/pasta mistake in declaring admin routers
- Fixed declaring mysql4 models
- Fixed typo in helpers declaration
- Added configuration ACL entries

0.5.0

- Initial release