Magento Cheat Sheet extensionprogrammer.com

Event-Observer

```
<config>
 <frontend> [1]
  <events>
   <catalog_product_load_after> [2]
    <observers>
     <yourModule> [3]
      <type>singleton</type> [4]
      <class>module/observer</class> [5]
      <method>catalogProductLoadAfter</method> [6]
     </yourModule>
    </observers>
   </catalog_product_load_after>
  </events>
 </frontend>
</config>
```

[1] Area: „adminhtml“, „frontend“ oder „global“.
[2] Event Code
[3] Must be unique within the event observer.
[4] „model“ or „singleton“
[5] Your_Class_Name oder module/class
[6] The method to be called on the observer instance.

Register Module Controllers

```
<config>
 <frontend> [1]
  <routers>
   <yourModule> [2]
    <use>standard</use> [3]
    <args>
     <module>Namespace_Module</module> [4]
     <frontName>yourmod</frontName> [5]
    </args>
   </yourModule>
  </routers>
 </frontend>
</config>
```

[1] Depending on the controllers area „frontend“ or „admin“
[2] Must be unique
[3] „admin“ for Adminhtml controllers, otherwise „standard“
[4] Namespace and Name of the Module as defined in the <config><modules> node
[5] The URL part to point to the controller directory (e.g. http://example.com/yourmod)
Magento will look in Namespace/Module/controllers/ for the called controller.

Rewrite Controllers

```
<config>
 <frontend> [1]
  <routers>
   <checkout> [2]
    <args>
     <modules>
      <yourModule
       before="Mage_Checkout">Namespace_Module_Overwrite_Checkout</yourModule> [3]
     </modules>
    </args>
   </checkout>
  </routers>
 </frontend>
</config>
```

[1] Depending on the controllers area „frontend“ or „admin“
[2] Router node of the module to be overridden (look it up in the config.xml of that module)
[3] The tag <yourModule> can be anything, it must be unique within the <modules> node.
Your_Module_Overwrite_Checkout (is mapped to directory)
Your/Module/controllers/Overwrite/Checkout/

Rewrite a Model/Block/Helper

```
<config>
 <global>
  <models> [1]
   <sales> [2]
    <rewrite>
     <order_item>Namespace_Module_Model_Sales_Order_Item</order_item> [3]
    </rewrite>
   </sales>
  </models>
 </global>
</config>
```

[1] Depending on object type „models“, „helpers“ or „blocks“
[2] Modul shorthand
[3] The tag is the class to override, the content is the full name of the replacement class.

Define a Model / Block / Helper module shorthand

```
<config>
 <global>
  <models> [1]
   <yourModule> [2]
    <class>Namespace_Module_Model</class> [3]
   </yourModule>
  </models>
 </global>
</config>
```

[1] Depending on object type „models“, „helpers“ or „blocks“
[2] The module shorthand, must be unique for the module
[3] Classname prefix, will replace the shorthand [2] when resolving a class name.

Magento Developer Cheatsheet (v1.2 2010-04-21) Präsentiert von ecommerce-schulungen.de

Magento Cheat Sheet extensionprogrammer.com

The most common layout XML

Update Handles:

`<default>`	Always applied
`<[module]_[controller]_[action]>`	Applied when the matching action is called, e.g.: catalog_product_view when a customer views a product detail page.
`<customer_account>`	Applied on all pages of the customer account.
`<catalog_category_default>`	Applied on all category pages without the layered navigation (Is Anchor = No).
`<catalog_category_layered>`	Applied on all category pages with active layered navigation (Is Anchor = Yes).

Tags:

`<block type="module/class" name="name_in_layout" as="alias" template="path/to/the/template.phtml"/>`	Creates an instance of the specified type. The name_in_layout must be unique. The alias must be unique within the parent block.
`<reference name="name_in_layout">`	Applies the contained layout updates to the block specified in <reference name="...">.
`<update handle="update_handle_name"/>`	Applies the layout updates specified in the update handle „update_handle_name".
`<remove name="name_in_layout/>`	Unsets the block with the name „name_in_layout".
`<action method="nameOfTheMethod"/>`	Using the <action> tag any public method can be called on Block Instances. Strings can be passed as arguments (see next example), or the return value of a helper method can be used to pass complex values (e.g. <url helper="customer/getAccountUrl"/>).
`<action method="unsetChild"> <name>alias</name> </action>`	Removes the block with the specified alias from the list of sub-blocks from the containing block. The parent must be specified using <reference>. The removed block instance still exists and can be assigned to a different block using <insert> (see next example).
`<action method="insert"> <name>name_in_layout</name> </action>`	Inserts the block with the name „name_in_layout" as a sub-block to the current block specified with <reference>.
`<action method="setTemplate"> <template>path/template.phtml</template> </action>`	Set the template for the current block.
`<action method="addLink">`	Only usable on block types page/template_links and customer/account_navigation, but be careful: the parameters for both classes differ.

Questions & Answers

How do I get the Magento Home URL?

```
Mage::getBaseUrl();
```

How do I get the Magento Home Directory?

```
Mage::getBaseDir();
```

How do I get the URL of a file in the skin directory?

```
$this->getSkinUrl('images/myfile.png');              // in a template or Block
Mage::getDesign()->getSkinUrl('images/myfile.png'); // from anywhere
```

How do I format a price value?

```
Mage::helper('core')->formatPrice($amount);
```

How do I get the display value of a (multi-)select product attribute?

```
echo $product->getAttributeText('manufacturer');
```

How do I create an object instance in magento?

```
Mage::getModel('module/class');       // for models
Mage::helper('module/class');        // for helpers
Mage::app()->getLayout()->createBlock('module/class', 'name_in_layout'); // for blocks
```

How do I get at GET/POST parameters?

```
Mage::app()->getRequest()->getParam('param_name');    // single parameter
Mage::app()->getRequest()->getParams();               // all parameters
```

How do I get the Id of the curent store view?

```
Mage::app()->getStore()->getId();
```

How do I get all Store Views?

```
Mage::app()->getStores(); // pass true to include the admin store view, too
```

How do I get the configurable/grouped/bundled product a simple product belongs to?

```
$simpleProduct->loadParentProductIds();
$parentProductIds = $simpleProduct->getParentProductIds();
```

How do I get the simple products assigned to a configurable product?

```
$configProduct->getTypeInstance()->getUsedProductCollection($configProduct);
```

How do I get an instance of an attribute?

```
Mage::getSingleton('eav/config')->getAttribute($entityType, $attributeCode)
```

Magento Developer Cheatsheet (v1.2 2010-04-21) Präsentiert von ecommerce-schulungen.de