MAGENTO ERKLÄRT

in 30 Minuten

WAS ZU SAGEN?

AUF TWITTER MIT #MHRH

VON UND MIT RIBIAN BLECHTZEL

RICO NEITZEL *und* FABIAN BLECHSCHMIDT

PRÄSENTATION ONLINE

Die Präsentation findet ihr unter:
http://diepraesentationistonline.de/

DER INHALT:

VIELES WAS EIN MAGENTO PROGRAMMIERER WISSEN SOLLTE

INDEX.PHP

DEVMODE

MULTISITE

CODEPOOLS

NAMESPACES

EIGENE MODULE

CONFIG.XML

FACTORYNAMES

ROUTINGWORKFLOW

FRONTNAME

CONTROLLER

ACTION

LAYOUTWORKFLOW

LOCAL.XML

LAYOUTXML

BLÖCKE

TEMPLATES

CACHING

LAYOUTHANDLES

PAGETEMPLATES

FALLBACK

OBSERVER

SYSTEMKONFIGURATION

SYSTEM.XML

ACL

DEFAULTS

SETUPSKRIPTE

DATASKRIPTE

HELPER

REWRITES

MODELS

RESOURCEMODELS

COLLECTIONS


EAV

SQL-ABSTRAKTION

WEITERLEITUNGEN

UMLEITUNGEN

EXCEPTIONS

OH VERDAMMTE 
DAS SCHAFFEN WIR NIE

UND LOS

INDEX.PHP

- Apache – Konfiguration (nie möglich)
- .htaccess – Anpassung (selten möglich)
- index.php – Gepfusche (der Standard)

„Anpassungen in der index.php vermeiden.“

– Ribian Blechtzel

DEVMODE

FEHLERAUSGABEN ANSCHALTEN

Mehr bzw. überhaupt erstmal Fehler im Frontend,
das AUS für fehlende Klassen und Methoden

`ini_set('display_errors', 1);` einkommentieren

"WENIGER" ÜBERSETZUNG

Block und Übersetzung müssen im gleichen Modul liegen!

Es wird nur übersetzt, wenn die Übersetzungsdatei im Modul ist (`app/locale/de_DE/My_Module.csv`),

Übersetzung in Modulen kann den Standard-Data-Helper erfordern `MyCompany_MyModule_Helper_Data`

BLOCKÜBERSCHREIBUNG VERHINDERN

Wenn zwei Blöcke den selben Namen haben eigentlich auch, aber:

```
# …/core/Mage/Core/Model/Layout.php:450
…
} elseif (isset($this->_blocks[$name]) && Mage::getIsDeveloperMode()) {
    //Mage::throwException(Mage::helper('core')->__('Block with name "%s" already exists', $name));
}
…
```

daher passiert hier nichts, auch nicht im DevMode

DEVMODE AN/AUS

VHOST KONFIGURATION ODER .HTACCESS

```
SetEnv MAGE_IS_DEVELOPER_MODE
```

muss keinen Wert haben, Setzen reicht

DEVMODE AN/AUS

Die hässlichste, aber häufigste Lösung

INDEX.PHP:

```
if (true || isset($_SERVER['MAGE_IS_DEVELOPER_MODE'])) {
```

UNTERSCHIEDLICHE

WEBSITES/STORES DIREKT STARTEN

VHOST KONFIGURATION /.HTACCESS

```
SetEnv MAGE_RUN_CODE {storecode}
```

oder

```
SetEnvIf Host www\.domain2\.com MAGE_RUN_CODE={storecode}
```

oder

```
SetEnv MAGE_RUN_CODE {websitecode}
SetEnv MAGE_RUN_TYPE website
```

Geht super und non-destruktiv, weil:

```
# index.php:82
$mageRunCode = isset($_SERVER['MAGE_RUN_CODE']) ? _SERVER['MAGE_RUN_CODE'] : '';
$mageRunType = isset($_SERVER['MAGE_RUN_TYPE']) ? _SERVER['MAGE_RUN_TYPE'] : 'store';
```

INDEX.PHP

Auch hier wieder:

Die hässlichste, aber häufigste Lösung vor den eben gezeigten Code:

```
switch($_SERVER['SERVER_NAME']):
    case 'domain1.de':
    case 'www.domain1.de':
        $_SERVER['MAGE_RUN_CODE'] = '{storeviewcode}';
        break;
    case 'domain2.de':
    case 'www.domain2.de':
        $_SERVER['MAGE_RUN_CODE'] = '{websitecode}';
        $_SERVER['MAGE_RUN_TYPE'] = 'website';
        break;
endswitch;

...
```

EIGENE MODULE IN MAGENTO

naja, wo auch sonst

MODULE WERDEN IN NAMESPACES GRUPPIERT

NAMESPACES

sind nur Ordner

keine Magie

dienen der Struktur

alles von einem schön zusammen

sorgen für weniger Konflikte

gleiche Modulnamen, unterschiedliche Namespaces

FÜR EIN EIGENES MODUL SIND 3 DINGE NÖTIG

1. Eine gute Idee
2. eine Deklarationsdatei (/app/etc/modules/)
3. eine Konfigurationsdatei (./etc/config.xml)

EIGENES MODUL DEKLARIEREN

Eine Datei anlegen in `app/etc/modules/`

{Namespace}_{Modul}.xml

Achtung: Die beiden Teile (Namespace, Modul) müssen mit einem Großbuchstaben anfangen.

INHALT DER {NAMESPACE}_{MODUL}.XML

```
<?xml version="1.0"?>
<config>
    <modules>
        <{Namespace}_{Modul}>
            <active>{true|false}</active>
            <codePool>{local|community}</codePool>
            <depends>
                <Mage_Catalog />
                <Mage_Customer />
            </depends>
        </{Namespace}_{Modul}>
    </modules>
</config>
```

DER KLASSIKER

`<codePool>` wird mit einem großen P geschrieben!
(siehe später auch `<frontName>`)

DER CODE MUSS IRGENDWO HIN

Anschließend landet der ganze Modulcode in

`app/code/{codePool}/{Namespace}/{Modul}/`

INAKTIV

Bei der Prüfung, ob ein Modul aktiviert ist,
erwartet Magento die Zeichenfolge 'true'

Daher ist `<active>1</active>`
oder `<active>TRUE</active>`
für Magento ein "nein, ist inaktiv".

```
# …/core/Mage/Core/Model/Config.php:807
```

MODULJUNKIES

Wenn andere Module vorher geladen sein müssen:

```
<depends>
    <Mage_Catalog />
</depends>
```

KONFIGURATION MUSS SEIN

CONFIG.XML

Die config.xml im Modulverzeichnis ist für die Konfiguration des gesamten Moduls zuständig. Hier werden PHP-Klassen-Namen vorkonfiguriert und Dateien 'verlinkt'.

Dokumentiertes Beispiel einer einfachen, ausreichenden config.xml

```xml
<?xml version="1.0"?>
<config>
    <!-- Versionierung: Ist wichtig für Installscripte (Setup und Data) -->
    <modules>
        <Namespace_Modul>
            <version>0.1.0</version>
        </Namespace_Modul>
    </modules>

    <!-- Knoten für die übergreifende Konfiguration -->
    <global>
        <models>
            <!-- dieser Knoten heißt: class-group (wird später bei Mage::getModel() verwendet)
            <namespace_modul>
                <!-- die class-group 'namespace_modul' löst Magento zu 'Namespace_Modul_Model' auf -->
                <class>Namespace_Modul_Model</class>
                <resourceModel>namespace_modul_resource</resourceModel>
            </namespace_modul>
            <!-- dieser Knoten heißt auch: class-group (wird später indirekt in Mage::getResourceModel() verwendet)
            <namespace_modul_resource>
                <class>Namespace_Modul_Model_Resource</class>
                <entities>
                    <!-- dieser Knoten heißt: entity (damit kann Magento später den Tabellennamen finden) -->
                    <developer>
                        <!-- der eigentliche Tabellenname -->
                        <table>developer</table>
                    </developer>
                </entities>
            </namespace_modul_resource>
        </models>
        <blocks>
            <!-- dieser Knoten heißt: class-group (wird später bei Mage::getBlock() verwendet)
            <namespace_modul>
                <!-- die class-group 'namespace_modul' löst Magento zu 'Namespace_Modul_Block' auf -->
                <class>Namespace_Modul_Block</class>
            </namespace_modul>
        </blocks>
        <helpers>
            <!-- dieser Knoten heißt: class-group (wird später bei Mage::helper() verwendet)
            <namespace_modul>
                <!-- die class-group 'namespace_modul' löst Magento zu 'Namespace_Modul_Helper' auf -->
                <class>Namespace_Modul_Helper</class>
            </namespace_modul>
        </helpers>
        <resources>
            <!-- dieser Knoten bestimmt den Unterordner in ./sql und ./data für die Installscripte -->
            <namespace_modul_setup>
                <setup>
                    <module>Namespace_Modul</module>
                </setup>
            </namespace_modul_setup>
        </resources>
    </global>
</config>
```

UNTERKNOTEN IN DER CONFIG.XML

MODULES

```
<modules>
    <{Namespace_Modul}>
        <version>
```

GLOBAL /1

```
<models>
    <{namespace_modul}>
        <class>
        <resourceModel>
    <{namespace_modul}_resource>
        <deprecatedNode>
        <entities>
            <{entity}>
                <table>
        <rewrite>
            <{model}>
```

GLOBAL /2

```
<helpers>
    <{namespace_modul}>
        <class>
        <rewrite>
            <{helper}>

<blocks>
    <{namespace_modul}>
        <class>
        <rewrite>
            <{block}>
```

GLOBAL /3

```
<index>
    <indexer>
        <{indexer_code}>
            <model>
            <depends>
                <{indexer_code}/>

<cache>
    <types>
        <{cache_code}>
            <label>
            <description>
            <tags>
```

GLOBAL /4

```
<resources>
    <{namespace_modul}_setup>
        <connection>
            <use>
        <setup>
            <module>
            <class>

<template>
    <email>
        <{email_template_code}>
            <label>
            <file>
            <type>

<page>
    <layouts>
        <{page_template_code}>
            <label>
            <template>
            <layout_handle>
            <is_default>
```

GLOBAL /5

```
<events>
    <{event_name}>
        <observers>
            <{namespace_modul}>
                <type>
                    <model>
                    <class>
                    <method>
```

GLOBAL /6

```
<sales>
    <quote>
        <item>
            <product_attributes>
                <sku/>
                <type_id/>
                <name/>
                <status/>
                <visibility/>
                <price/>
                <weight/>
                <url_path/>
                <url_key/>
                …
```

INSTALL

```
<events>
    <{event_name}>
        <observers>
            <{namespace_modul}>
                <type>
                    <model>
                    <class>
                    <method>

<translate>
    <modules>
        <{Namespace_Modul}>
            <file>
                <default>
                <others>

<layout>
    <updates>
        <{namespace_modul}>
            <file>
```

ADMIN

```
<routers>
    <{routername}>
        <use>
        <args>
            <module>
            <frontName>
            <modules>
                <{Namespace_Modul}>
```

ADMINHTML /1

```
<events>
    <{event_name}>
        <observers>
            <{namespace_modul}>
                <type>
                <model>
                <class>
                <method>

<translate>
    <modules>
        <{Namespace_Modul}>
            <file>
                <default>
```

ADMINHTML /2

```
<layout>
    <updates>
        <{namespace_modul}>
            <file>

<global_search>
    <{entity}>
        <class>
```

FRONTEND /1

```
<category>
    <collection>
        <attributes>
            <{category_attributecode_to_load}>

<product>
    <collection>
        <attributes>
            <{product_attributecode_to_load}>

<routers>
    <{routername}>
        <use>
        <args>
            <module>
            <frontName>
            <modules>
                <{Namespace_Modul}>
```

FRONTEND /2

```
<events>
    <{event_name}>
        <observers>
            <{namespace_modul}>
                <type>
                    <model>
                    <class>
                    <method>

<translate>
    <modules>
        <{Namespace_Modul}>
            <file>
                <default>
                <others>

<layout>
    <updates>
        <{namespace_modul}>
            <file>
```

CRONTAB

```
<crontab>
    <jobs>
        <{cronjob_identifier}>
            <schedule>
                <cron_expr>
            <run>
                <model>
```

FACTORYNAMES

In Magento gibt's kein

`new Klassenname()`

Stattdessen Factories mit FactoryNames via XML-Konfiguration

und ja: das macht's auch so blöde kompliziert

AM BEISPIEL EINES BLOCKS

catalog/product_view

wird übersetzt in

Mage_Catalog_Block_Product_View

und liegt in der Datei

/app/code/core/Mage/Catalog/Block/Product/View.php

Die Auflösung des Teils vor dem / wird später erklärt.
Der Teil nach dem / wird durch folgende Methode geschickt:

```
str_replace(" ", DS, ucfirst(str_replace("_", " ", $zweiterTeil))).'.php'
```

AUF DEUTSCH

1. alle _ durch Leerzeichen ersetzen
2. alle Wörter in 'Anfangsbuchstabe groß' ändern
3. alle Leerzeichen durch die Konstante DS ersetzen
4. am Ende .php anfügen

DS steht hier bei für DirectorySeparator und enthält je nach Betriebssystem einen / oder \ ... \o/ YAY!

GROSSBUCHSTABENSALAT

my_module/grOSSbuChstaBenSaLAT

wird übersetzt in

My_Module_Block_GrOSSbuChstaBenSaLAT

und liegt in der Datei

/app/code/local/My/Module/Block/GrOSSbuChstaBenSaLAT.php

ROUTING-WORKFLOW

VON DER INDEX.PHP BIS ZUM CONTROLLER

EIGENER FRONTNAME (ROUTE)

IN DER CONFIG.XML

```
<config>
    <frontend>
        <routers>
            <{namespace_module}>
                <use>standard</use>
                <args>
                    <module>{Namespace_Module}</module>
                    <frontName>{frontname}</frontName>
                </args>
            </{namespace_module}>
        </routers>
    </frontend>
</config>
```

ACHTUNG

Der Tag frontName wird mit einem großen N geschrieben!
(siehe auch `<codePool>`)

Damit steht im Frontend

`http://domain.tld/{frontname)/`

zur Verfügung.

Beispiel:

`http://domain.tld/ribian/`

`<frontName>ribian</frontName>`

EIGENER CONTROLLER

Im eigenen Modul muss im Ordner controllers
(Achtung: kleingeschrieben und Plural)
eine IndexController.php angelegt werden.

Der Klassenname entspricht teilweise dem Verzeichnispfad: Namespace_Module_IndexControllers. Diese Klasse muss von `Mage_Core_Controller_Front_Action` ableiten.

Damit steht im Frontend `http://domain.tld/{frontname)/index/` zur Verfügung.

DER 2. TEIL DER URL

```
http://domain.tld/{frontname)/{controller}/
```

(hier {controller}) entspricht dem Beginn des Dateinamens im controllers Verzeichnis: {Controller}Controller.php.

Achtung: Das erste Zeichen muss großgeschrieben werden, da Magento diese Klasse per Autoloader laden wird.

Beispiel:

```
http://domain.tld/ribian/blechtzel/

# sucht nach
./controllers/BlechtzelController.php
```

EIGENE ACTION

```
http://domain.tld/{frontname)/{controller}/{action}/
```

Actions sind **public Methoden** innerhalb der eigenen Controller Klasse.

Der Name dieser Methoden setzt sich immer aus dem URL-Teil {action} und "Action" zusammen.

```
http://domain.tld/ribian/blechtzel/mergeface
```

```
public function mergefaceAction()
```

Achtung: Das erste Zeichen muss kleingeschrieben werden.

http://domain.tld/ribian/blechtzel/mergeface

ribian `<frontName>ribian</frontName>`

blechtzel `./controllers/BlechtzelController.php`

mergeface `public function mergefaceAction()`

FRONTNAME/CONTROLLER/ACTION ZUSAMMENFASSUNG

EIGENE ACTION

Layout Handle nimmt den NODE namen!

Route: frontend/<controllerName>/<actionName>
Layout-Handle: companyModule_frontend (NODE NAME!)/
<controllerName>/<actionName>

EIGENE PARAMETER PER GET

Eigene GET-Parameter können einfach
mit /key/value an die URL angehängt werden.

```
http://domain.tld/ribian/blechtzel/mergeface/key/value
```

```
array("key" => "value");
```

ROUTING WORKFLOW IM DETAIL

Index.php ruft Mage::run() auf

Mage::run() ruft Mage_Core_Model_App::run() auf

Mage_Core_Model_App::run() lädt
Mage_Core_Controller_Varien_Front ('FrontController');

FrontController holt alle verfügbaren Routes der Module ab

anschließend ruft run() dispatch() auf dem FrontController auf

dispatch() match()'t die angeforderte URL gegen alle Routes

match() prüft auf verfügbare Controller-Klassen

match() prüft auf verfügbare Action-Methode
innerhalb der gefundenen Controller-Klasse

anschließend wird dispatch()
auf der gefundenen Controller-Klasse aufgerufen

(kommt von Mage_Core_Controller_Varien_Action)

dispatch() führt dann schlussendlich die Action im Controller aus.

LAYOUT-WORKFLOW

VON DER LAYOUT-XML ZUM HTML

LOCAL.XML

In das eigene Theme einfach eine local.xml speichern:

```
<?xml version="1.0"?>
<layout>
    <default>
    </default>
</layout>
```

LOCAL.XML

Dann kann man das Layout in einer eigenen Datei beeinflussen.
Sie wird von Magento automatisch ganz zum Schluss geladen.

Vorteil: Die local.xml wird bei einem Update nicht überschrieben.

CSS DATEI EINBINDEN UND ENTFERNEN

```
<!-- Block mit dem Namen 'head' laden -->
<reference name="head">
    <!-- auf dem Blockobjekt die Methode 'addItem' aufrufen -->
    <action method="addItem">
        <!-- der Methode folgende 4 Argumente übergeben -->
        <!-- HTML-Tag-Typ und Pfad-Ursprung für die Datei festlegen -->
        <type>skin_css</type>
        <!-- relativen Pfad zur Datei angeben -->
        <filename>{relative/path/in/theme/file.css}</filename>
        <!-- HTML-Attribute mit Werten festlegen, wenn nötig, sonst <params/> -->
        <!-- wenn kein <if> benötigt wird, kann man die Argumente hier auch weglasse n-->
        <params>{attribute="value"}</params>
        <!-- Anweisung für den Conditional Comment festlegen -->
        <if>{lt IE 7}</if>
    </action>
</reference>
```

JAVASCRIPT DATEI EINBINDEN UND ENTFERNEN

```
<reference name="head">
    <action method="addItem">
        <type>skin_js</type>
        <filename>{relative/path/in/theme/file.js}</filename>
        <params>{attribute="value"}</params>
        <if>{lt IE 7}</if>
    </action>
</reference>
```

ES SIND VERSCHIEDENE WERTE FÜR `<type>` MÖGLICH

- skin_css – CSS Datei relativ zum Theme-Ordner (CSS für's Theme)
- skin_js – JavaScript Datei relativ zum Theme-Ordner (js)
- js_css – CSS Datei relativ zum js-Ordner (CSS für js Libraries)
- js – JS Datei relativ zum js-Ordner (js-Libraries)

UNÜBLICHE WERTE FÜR `<type>`

- rss – RSS-URL
- link_rel – <link> Url

EIGENER BLOCK

Für einen eigenen Block braucht/muss man die config.xml

und eine Klassendatei mit Vererbung:
Mage_Core_Block_Abstract
oder Mage_Core_Block_Template

ggf. Template vorbelegen

CONFIG.XML

```
<config>
    <global>
        <blocks>
            <!-- dieser Knoten heißt: class-group
            (wird später bei Mage::getBlock() verwendet) -->
            <{namespace_modul}>
                <!-- die class-group 'namespace_modul'
                löst Magento zu 'Namespace_Modul_Block' auf -->
                <class>{Namespace_Modul}_Block</class>
            </{namespace_modul}>
        </blocks>
    </global>
</config>
```

```
<!-- LayoutXML -->
<block type="namespace_modul/ribian" …>`

<!-- config.xml -->
<global>
    <blocks>
        <{namespace_modul}>
            <!-- in <class> darf kein Zeilenumbruch rein!!! -->
            <class>Namespace_Modul_Block</class>
```

dann den 2. Teil des FactoryNames ribian dazusetzen: _Ribian

```
Namespace_Modul_Block_Ribian
```

KLASSENDATEI

Die Klassendatei für Namespace_Modul_Block_Ribian muss gemäß ihres Namens in app/code/local/Namespace/Modul/Block/ liegen und Ribian.php heißen.

VERERBUNG

Die meisten Blockklassen erben, wenn sie ein phtml-Template verwenden, die Klasse Mage_Core_Block_Template.

Für Blockklassen ohne Template reicht Mage_Core_Block_Abstract.

TEMPLATE VORBELEGEN

Block mit Template? Dann phtml-Datei vorbelegen,
falls der Frontendler das vergessen sollte.

Kann man im _construct()'or machen.

```
protected function _construct()
{
    parent::_construct();
    $this->setTemplate('{modul}/pfad/zur/datei.phtml');
}
```

BLOCK-CACHING

Dafür ist nötig:

- Lifetime in Sekunden
- Key, um verschiedene Varianten zu unterscheiden
- CacheTag, um sie zu gruppieren und ggf. zu löschen

BLOCK-CACHING

```
// app/code/core/Mage/Core/Block/Abstract.php:904 - gekürzt

final public function toHtml()
{
    $html = $this->_loadCache();
    if ($html === false) {
        $this->_beforeToHtml();
        $html = $this->_toHtml();
        $this->_saveCache($html);
    }
    $html = $this->_afterToHtml($html);
```

ÜBERRASCHENDE ERKENNTNIS

wird nur ungecacht ausgeführt:

```
$this->_beforeToHtml();
```

wird auch gecacht ausgeführt:

```
$html = $this->_afterToHtml($html);
```

BLOCK-CACHING

```
# …/core/Mage/Core/Block/Abstract.php:1400

protected function _saveCache($data)
{
    if (is_null($this->getCacheLifetime())
            || !Mage::app()->useCache(self::CACHE_GROUP)) {
        return false;
    }
    $cacheKey = $this->getCacheKey();
    // SessionID ersetzen
    Mage::app()->saveCache(
      $data, $cacheKey, $this->getCacheTags(),
      $this->getCacheLifetime()
    );
    return $this;
}
```

BLOCK-CACHING

```
# …/core/Mage/Core/Block/Abstract.php:1362

public function getCacheLifetime()
{
    if (!$this->hasData('cache_lifetime')) {
        return null;
    }
    return $this->getData('cache_lifetime');
}
```

getCacheLifetime **überschreiben** oder
$this->setCacheLifetime(120)

BLOCK-CACHING

```
# …/core/Mage/Core/Block/Abstract.php:1289

public function getCacheKey()
{
    if ($this->hasData('cache_key')) {
        return $this->getData('cache_key');
    }
    $key = $this->getCacheKeyInfo();
    $key = array_values($key);  // ignore array keys
    $key = implode('|', $key);
    $key = sha1($key);
    return $key;
}
```

BLOCK-CACHING

```
protected function getCacheKeyInfo() {}
```

überschreiben oder

```
$this->setCacheKey(
    $storeId.
    $customerId.
    $productId
);
```

BLOCK-CACHING

```
# …/core/Mage/Core/Block/Abstract.php:1311

public function getCacheTags()
{
    if (!$this->hasData('cache_tags')) {
        $tags = array();
    } else {
        $tags = $this->getData('cache_tags');
    }
    $tags[] = self::CACHE_GROUP;
    return $tags;
}
```

BLOCK-CACHING

```
$this->setCacheTags(array(
    Mage_Catalog_Model_Product::CACHE_TAG,
    'STORE_'.$storeId.'-PRODUCTS',
));
```

Ein Cache-Eintrag kann mehrere CacheTags gleichzeitig haben.

EIGENE LAYOUT.XML

EIGENE LAYOUT.XML

CONFIG.XML

```
<frontend>
    <layout>
        <updates>
            <namespace_modul>
                <file>namespace/modul.xml</file>
```

kommt dann in /app/design/frontend/base/default/layout/

PHTANDARD LAYOUT-HANDLE


PHTEVEN?
IS THAT YOU?...


PHTEVEN?
PHTANDARD LAYOUT-HANDLE
IS THAT YOU?...

Die wichtigsten Layout-Handle, die man nicht von der URL direkt ablesen kann sind:

- <default> (alle seiten)
- <customer_logged_(in|out)> (Kundenstatus)
- <catalog_category_default> (Kat. ohne LN)
- <catalog_category_layered> (Kat. mit LN)
- <catalog_category_view> (Alle Kategorien)
- <catalog_product_view> (Artikeldetailseite)

BESONDERE LAYOUT-HANDLE

<PRODUCT_TYPE_(simple|configurable|grouped|virtual|downloadable|bundle{|giftcard})

EIGENES LAYOUT-HANDLE

Am einfachsten über eigenen Observer

```
public function addMyOwnLayoutHandle(){
    Mage::app()->getLayout()->addHandle('my_own_handle');
}
```

dann in der LayoutXML

```
<layout>
    <my_own_handle>
```

EIGENES PAGE-TEMPLATE

WIR NENNEN SIE 4COLUMNS.PHTML

config.xml für Page-Template und fourcolumns.xml

```
<frontend>
    <layout>
        <updates>
            <file>fourcolumns.xml</file>
<global>
    <page>
        <layouts>
            <four_columns module="page" translate="label">
                <label>4 columns</label>
                <template>page/4columns.phtml</template>
                <layout_handle>page_four_columns</layout_handle>
            </four_columns>
```

fourcolumns.xml anlegen

```
<layout>
    <page_four_columns>
        <reference name="root">
            <block type="core/text_list" name="four" />
            <action method="setTemplate">
                <file>page/4columns.phtml</file>
```

4columns.phtml anlegen

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="<?php echo $this->getLang() ?>" lang="<?php echo $this->getLang() ?>">
<head>
<?php echo $this->getChildHtml('head') ?>
</head>
<body<?php echo $this->getBodyClass()?' class="'.$this->getBodyClass().'"':'' ?>>
<?php echo $this->getChildHtml('after_body_start') ?>
<div class="wrapper">
    <?php echo $this->getChildHtml('global_notices') ?>
    <div class="page">
        <?php echo $this->getChildHtml('header') ?>
        <div class="main-container col3-layout">
            <div class="main">
                <?php echo $this->getChildHtml('breadcrumbs') ?>
                <div class="col-wrapper">
                    <div class="col-main">
                        <?php echo $this->getChildHtml('global_messages') ?>
                        <?php echo $this->getChildHtml('content') ?>
                    </div>
                    <div class="col-left sidebar"><?php echo $this->getChildHtml('left') ?></div>
                </div>
                <div class="col-right sidebar"><?php echo $this->getChildHtml('right') ?></div>
            </div>
            <div class="col-right sidebar"><?php echo $this->getChildHtml('four') ?></div>
        </div>
        <?php echo $this->getChildHtml('footer') ?>
        <?php echo $this->getChildHtml('global_cookie_notice') ?>
        <?php echo $this->getChildHtml('before_body_end') ?>
    </div>
</div>
<?php echo $this->getAbsoluteFooter() ?>
</body>
</html>
```

EIGENES BLOCK-TEMPLATE

Templates sind PHP+HTML Dateien (.phtml)

EIGENES BLOCK-TEMPLATE

Template setzen in der Blockklasse:

```
$block->setTemplate('meinTemplate.phtml');
```

Layout XML

```
<block type="core/template" name="…" template="folder/filename.phtml" />
<!-- oder -->
<reference name="blockname">
    <action method="setTemplate">
        <val>folder/filename.phtml</val>
    </action>
</reference>
```

THEME FALLBACK

1. *eigenes* Theme für Dateiart im *eigenen* Package
2. *eigenes* Standard-Theme im *eigenen* Package
3. default Theme im *eigenen* Package
4. default Theme im base Package

THEME FALLBACK

Custom Fallback gewünscht?

.../core/Mage/Core/Model/Design/Package.php:392

und bitte ...

OBSERVER IMPLEMENTIEREN

EVENT FINDEN

MODELS

Vor und nach jedem Laden

```
# …/core/Mage/Core/Model/Abstract.php:255
Mage::dispatchEvent($this->_eventPrefix.'_load_before', $params);

# …/core/Mage/Core/Model/Abstract.php:267
Mage::dispatchEvent($this->_eventPrefix.'_load_after', $this->_getEventData());
```

Vor und nach jedem Speichern

```
# …/core/Mage/Core/Model/Abstract.php:391
Mage::dispatchEvent($this->_eventPrefix.'_save_before', $this->_getEventData());

# …/core/Mage/Core/Model/Abstract.php:466
Mage::dispatchEvent($this->_eventPrefix.'_save_after', $this->_getEventData());
```

Vor, während und nach jedem Löschen

```
# …/core/Mage/Core/Model/Abstract.php:501
Mage::dispatchEvent($this->_eventPrefix.'_delete_before', $this->_getEventData());

// Das findet innerhalb der Transaktion statt!
# …/core/Mage/Core/Model/Abstract.php:529
Mage::dispatchEvent($this->_eventPrefix.'_delete_after', $this->_getEventData());

// nach der Transaktion
# …/core/Mage/Core/Model/Abstract.php:541
Mage::dispatchEvent($this->_eventPrefix.'_delete_commit_after', $this->_getEventData());
```

BLÖCKE

vor und nach _prepareLayout()

```
# …/core/Mage/Core/Block/Abstract.php:237
Mage::dispatchEvent('core_block_abstract_prepare_layout_before',
array('block' => $this));

$this->_prepareLayout();

Mage::dispatchEvent('core_block_abstract_prepare_layout_after',
        array('block' => $this));
```

Am Anfang von `_toHtml()`

```
# …/core/Mage/Core/Block/Abstract.php:850
Mage::dispatchEvent('core_block_abstract_to_html_before',
array('block' => $this));
```

Am Ende von `_toHtml()`

```
# …/core/Mage/Core/Block/Abstract.php:886
Mage::dispatchEvent('core_block_abstract_to_html_after',
array('block' => $this, 'transport' => self::$_transportObject));
```

Wird beides auch mit aktivem Blockcache gefeuert

BEISPIELE

```
# …/core/Mage/Checkout/Model/Cart.php:296
Mage::dispatchEvent('checkout_cart_product_add_after',
array('quote_item' => $result, 'product' => $product));

# …/core/Mage/Checkout/Model/Cart.php:396
Mage::dispatchEvent('checkout_cart_update_items_before',
array('cart'=>$this, 'info'=>$data));

// Preis vom Quote-Item ändern -> Produkt anpassen
Mage::dispatchEvent('sales_quote_item_set_product', array(
    'product' => $product,
    'quote_item'=>$this
));
```

SUCHEN, SUCHEN, SUCHEN

Es gibt fast immer ein Event
(außer beim Mailversand)

CONFIG.XML

```
<events>
    <sales_order_save_before>
        <observers>
            <namespace_modul>
                <type>singleton</type>
                <class>Namespace_Modul_Model_Observer</class>
                <method>salesOrderSaveBefore</method>
```

<type>

- singleton
- model oder object
- disabled

```
Mage::getModel(…)
Mage::getSingleton(…)
```

<class>

Parameter für getModel/getSingleton
Best pratice: My_Class statt namespace/model

EVENT ÄNDERN

- alten Observer auf disabled
- neuen Observer implementieren

<method>

Methodenname - *ja wirklich!*

OBSERVER.PHP

```
public function salesOrderSaveBefore(Varien_Event_Observer $observer)
{
    $order = $observer->getOrder();
    $order->setOldState($order->getState());
}
```

SYSTEM KONFIGURATION

Stellt Eingabefelder unter System/Konfiguration bereit.

SYSTEM.XML

```
<?xml version="1.0"?>
<config>
  <tabs />
  <sections>
    <groups>
      <fields />
    </groups>
  </sections>
</config>
```

TABS

Tabs stehen links in der Leiste

```
  <tabs>
    <tabname translate="label" module="classgroup">
      <label>Label für das Tab</label>
      <sort_order>100</sort_order>
```

^Hinweis auf Helper wg. der Übersetzung!

Sections

```xml
  <sections>
    <sectionname translate="label" module="classgroup">
      <tab>tabname</tab>
      <label>Label für die Section</label>
      <sort_order>10</sort_order>
      <show_in_default>1</show_in_default>
      <show_in_website>1</show_in_website>
      <show_in_store>1</show_in_store>
      <groups />
```

Sections stehen auch links in der Leiste und sind Tabs zugeordnet!

GROUPS

```
<sections>
  <sectionname translate="label" module="classgroup">
    <groups>
      <groupname translate="label" module="classgroup">
        <label>Label für Gruppe</label>
        <sort_order>10</sort_order>
        <show_in_default>1</show_in_default>
        <show_in_website>1</show_in_website>
        <show_in_store>1</show_in_store>
        <fields />
```

Sind die auf- und zuklappbaren Gruppen rechts.

FIELDS

```
<sections>
  <sectioname translate="label" module="classgroup">
    <groups>
      <groupname translate="label comment" module="classgroup">
        <fields>
          <fieldname translate="label">
          <label>Label für Feld</label>
          <comment><![CDATA[Text für den Kommentar]]></comment>
          <frontend_type>text</frontend_type>
          <source_model>adminhtml/system_config_source_yesno</source_model>
          <sort_order>10</sort_order>
          <show_in_default>1</show_in_default>
          <show_in_website>1</show_in_website>
          <show_in_store>1</show_in_store>
```

Sind die Eingabefelder rechts.

HÄUFIGE FRONTEND TYPEN

TEXT, TEXTAREA, PASSWORD, SELECT, MULTISELECT

Achtung: Bei select und multiselect benötigt man noch ein Source Model, das die Daten für das DropDown-Feld bereitstellt.

Lust auf Mehr? Da: /lib/Varien/Data/Form/Element

Die Tagnamen der Section/Group/Fields ergeben
die XML-Pfadnamen in der System Konfiguration!

ADMIN ACL - ADMINHTML.XML

ACCESS CONTROL LIST

ohne Logout/Login gibt es einen 404
weil die ACL in der Session steht
und nicht mehr aktuell ist

oder Cooles Snippet vom Schrank (ACL Refresh):

```php
$session = $adminuser = Mage::getSingleton('admin/session');
/* @var $adminuser Mage_Admin_Model_User */
$adminuser = $session->getUser();
$adminuser->setReloadAclFlag(true);
$session->refreshAcl();
```

ADMINHTML.XML

```
<acl>
  <resources>
   <admin>
    <children>
     <system>
      <children>
       <config>
        <children>
         <sectionname translate="title" module="classgroup">
          <title>Titel in der ACL</title>
```

Damit die Zugriffe auf die Section Sytemkonfiguration gesteuert werden können.

SYSTEM KONFIGURATION

NICHT IN DER DATENBANK

In der config.xml ist genug Platz für alle,
auch die Systemkonfiguration.

Super, wenn die System-Konfiguration
über XML statt Backend laufen soll

Versionierung, Produktionssicherheit

DEFAULT

```
<default>
    <general>
        <locale>
            <firstday>1</firstday>
        </locale>
    </general>
</default>
```

Wichtig: Falls der Wert auch in `core_config_data` steht, nutzt Magento den Datenbankwert!

DEFAULTS FÜR

WEBSITES

Das gleiche, was für `<default>` gilt,
gilt auch für die einzelnen Websites.

Dazu legt man den Konfigurationspfad einfach in

```
<websites>
    <{website_code}>
```

DEFAULTS FÜR

STORES

Das gleiche, was für `<websites>` gilt,
gilt auch für die einzelnen Stores.

Dazu legt man den Konfigurationspfad einfach in

```
<stores>
    <{store_code}>
```

EIGENE SETUP- UND DATA-SKRIPTE

landen im Ordner ./sql oder ./data

CONFIG.XML

```
<resources>
    <namespace_modul_setup>
        <setup>
            <class>Mage_Catalog_Model_Resource_Setup</class>
            <module>Namespace_Modul</module>
        </setup>
    </namespace_modul_setup>
</resource>
```

Achtung: Sobald dieses XML angelegt ist, speichert Magento in der Tabelle core_resource die Versionsnummer des Moduls. Ggf. werden dann keine Setupskripte mehr ausgeführt, weil die Versionsnummer bereits in der Datenbank steht.

VERSIONSNUMMER

Magento entscheidet basierend auf der Versionsnummer von Modul und Datenbank, ob Installscripte ausgeführt werden müssen.

Namenskonventionen für die Dateien:

- install-<version>.php
- upgrade-<version_von>-<version_nach>.php

DATA-SCRIPTE

Analog dazu verhält es sich mit den Data-Scripten:

Namenskonventionen für die Dateien:

- data-install-<version>.php
- data-upgrade-<version_von>-<version_nach>.php

HELPER

Standardhelper heißt immer: Data

Dieser Name entspricht teilweise dem Dateinamen.

CONFIG.XML

```
<global>
    <helpers>
        <namespace_modul>
            <class>Namespace_Modul_Helper</class>
```

ERBEN VON MAGE_CORE_HELPER_ABSTRACT

```
class Namespace_Modul_Helper_Data
    extends Mage_Core_Helper_Abstract
{

    public function hilfMirMal(){
        return true;
    }

}
```

WOZU BRAUCHT MAN DAS?!

nur für Hilfsmethoden gedacht:

- createCamelCase()
- escapeHtml()

REWRITES

Einfach und relativ schmerzfrei
bestehende Funktionen ändern oder
neue Funktionen hinzufügen.

REWRITES

KLASSEN DIE NICHT REWRITTEN WERDEN KÖNNEN

- `Mage`
- `Mage_Core_Model_App`
- `Mage_Core_Model_Config`
- **geladen via** `new Mage_*`

MODEL

Proxy für die ORM-Schicht und Businesslogik-Container

Auf Deutsch

Leitet Speichern, Löschen, Ändern weiter und kann die Daten voher prüfen oder verändern.

Ruft `save` auf dem RessourceModel auf und übergibt sich selbst. 

– Ribian Blechtzel

CONFIG.XML

```
<global>
    <models>
        <namespace_modul>
            <class>Namespace_Modul_Model</class>
```

ERBEN VON MAGE_CORE_MODEL_ABSTRACT

```
class Namespace_Modul_Model_Hotel
    extends Mage_Core_Model_Abstract
{

    public function _construct(){
        $this->_init('namespace_model/hotel');
    }

}
```

RESOURCE MODEL

ORM-Schicht für Entitäten aus der Datenbank.

Auf Deutsch

Stellt Funktionen wie Speichern, Erstellen, Löschen zur Verfügung für Einträge aus einer Datenbanktabelle.

In leichtem Deutsch

Damit kann man was in einer Datenbank speichern oder löschen.

CONFIG.XML

```
<global>
    <models>
        <namespace_modul>
            <class>Namespace_Modul_Model</class>
            <resourceModel>namespace_modul_resource</resourceModel>
        <namespace_modul>
        <namespace_modul_resource>
            <class>Namespace_Modul_Model_Resource</class>
```

ERBEN VON MAGE_CORE_MODEL_RESOURCE_DB_ABSTRACT

```
class Namespace_Modul_Model_Resource_Hotel
    extends Mage_Core_Model_Resource_Db_Abstract
{

    public function _construct(){
        $this->_init('namespace_model/hotel', 'id');
    }

}
```

Hier muss der Spaltenname des PK als 2. Argument angegeben werden!

RESOURCE COLLECTION AKA COLLECTION

Bildet einen Container,
der mehrere Models gleichzeitig
aus der Datenbank laden und bereitstellen kann.

CONFIG.XML

brauchen wir nicht, ergibt sich automatisch.

ERBEN VON MAGE_CORE_MODEL_RESOURCE_DB_COLLECTION_ABSTRACT

```
class Namespace_Modul_Model_Resource_Hotel_Collection
    extends Mage_Core_Model_Resource_Db_Collection_Abstract
{

    public function _construct(){
        $this->_init('namespace_model/hotel');
    }

}
```

EAV VS. NORMALE COLLECTIONS

EAV Collections laden nicht automatisch alle Attribute mit.

E A V

ENTITITY ATTRIBUTE VALUE

EAV speichert Attributwerte in unterschiedlichen Tabellen.

Damit höchst flexibel bei der Datenspeicherung.

Damit höchst lahmarschig bei der Abfrage und Speicherung der Daten.

```
- entity
  +- entity-datetime
  +- entity-decimal
  +- entity-int
  +- entity-text
  +- entity-varchar
```

Models ohne EAV speichern alle Daten in einer Zeile einer Tabelle.

Damit nennt man alle Attribute static.

Auch EAV basierte Models haben static Attribute.

Diese stehen dann direkt als Spalte in der entity-Tabelle.

SQL ABSTRAKTION

SQL ABSTRAKTION

FILTERFUNKTIONEN

```
$collection->addFieldToFilter($field, array($operator => $ggf_filter_wert);
$collection->addAttributerToFilter($field, array($operator => $ggf_filter_wert);
```

WHERE OPERANDEN

Magento-Option	SQL
eq	=
neq	!=
like	LIKE
nlike	NOT LIKE
in	IN
nin	NOT IN
is	IS
notnull	IS NOT NULL
null	IS NULL
moreq	>=
gt	>
lt	<
gteq	>=
lteq	<=

WOHER KOMMT EIGENTLICH DER PREIS

- auf Kategorieseiten
- auf Produktdetailseite

KATEGORIESEITE

```
CREATE TABLE `catalog_product_index_price` (
  `entity_id` int(10) unsigned NOT NULL COMMENT 'Entity ID',
  `customer_group_id` smallint(5) unsigned NOT NULL COMMENT 'Customer Group ID',
  `website_id` smallint(5) unsigned NOT NULL COMMENT 'Website ID',
  `tax_class_id` smallint(5) unsigned DEFAULT '0' COMMENT 'Tax Class ID',
  `price` decimal(12,4) DEFAULT NULL COMMENT 'Price',
  `final_price` decimal(12,4) DEFAULT NULL COMMENT 'Final Price',
  `min_price` decimal(12,4) DEFAULT NULL COMMENT 'Min Price',
  `max_price` decimal(12,4) DEFAULT NULL COMMENT 'Max Price',
  `tier_price` decimal(12,4) DEFAULT NULL COMMENT 'Tier Price',
  ...
```

PRODUKTDETAILSEITE

```
// Mage_Catalog_Model_Product_Type_Price::getFinalPrice

1. $product->getPrice()
    // Preis Attribute
2. $this->_applyTierPrice($product, $qty, $finalPrice);
    // Staffelpreise
3. $this->_applySpecialPrice($product, $finalPrice);
    // Special Preis (von/bis Datum)
4. Mage::dispatchEvent('catalog_product_get_final_price',
        array('product'=>$product, 'qty' => $qty));
     // Möglichkeit den Preis via Event zu beeinflussen
     4a. Mage_CatalogRule_Model_Observer::processFrontFinalPrice
         // Katalogpreisregeln
5. $this->_applyOptionsPrice($product, $qty, $finalPrice);
   // Preise der Optionen werden addiert
6. return max(0, $finalPrice);
   // Der Preis ist mindestens Null
```

WEITERLEITUNGEN

- via Exception
- via Request

VIA EXCEPTION

```
// stark gekürzt
public function dispatch($action) {
    try {
        $this->preDispatch();
        if ($this->getRequest()->isDispatched()) {
            if (!$this->getFlag('', self::FLAG_NO_DISPATCH)) {
                $this->$actionMethodName();
                $this->postDispatch();
            }
        }
    }
    catch (Mage_Core_Controller_Varien_Exception $e) {
        // ...
    }
}

# class Mage_Core_Controller_Varien_Exception extends Exception
```

BEST PRACTICE

`try ... catch` **prüft**
auf `Mage_Core_Exception`, **nicht** `Exception`!

sonst wird z.B.
`Mage_Core_Controller_Varien_Exception`
fälschlicherweise gefangen.

VIA EXCEPTION - 2

```
catch (Mage_Core_Controller_Varien_Exception $e) {
    list($method, $parameters) = $e->getResultCallback();
    switch ($method) {
        case Mage_Core_Controller_Varien_Exception::RESULT_REDIRECT:
            list($path, $arguments) = $parameters;
            $this->_redirect($path, $arguments);
            break;
        case Mage_Core_Controller_Varien_Exception::RESULT_FORWARD:
            list($action, $controller, $module, $params) = $parameters;
            $this->_forward($action, $controller, $module, $params);
            break;
        default:
            $actionMethodName = $this->getActionMethodName($method);
            $this->getRequest()->setActionName($method);
            $this->$actionMethodName($method);
            break;
    }
}
```

FORWARD IN PRACTICE

```
$exception = new Mage_Core_Controller_Varien_Exception();
// $exception->prepareForward(
//     $actionName, $controllerName, $moduleName, $params
// );
$exception->prepareForward(
    'view', 'category', 'catalog', array('id' => $id)
);
throw $exception;
```

REDIRECT IN PRACTICE

Bug in Mage_Core_Controller_Varien_Exception::prepareRedirect()

-> Exception extenden
-> Methode überschreiben

```
public function prepareRedirect($path, $arguments = array())
{
    $this->_resultCallback = self::RESULT_REDIRECT;
    $this->_resultCallbackParams($path, $arguments); // ORIGINAL
    $this->_resultCallbackParams = array($path, $arguments);// RICHTIG
    return $this;
}
```

REDIRECT IN PRACTICE - 2

```
$exception = new My_Extended_Exception_Because_Of_The_Bug();
// $exception->prepareRedirect($path, $arguments);
$exception->prepareRedirect(
    'catalog/category/view', array('id' => $id)
);
throw $exception;
```

NAMENSAUFLÖSUNG
FÜR BLÖCKE, MODELS, HELPER

BEISPIEL MODEL

```
$product = Mage::getModel('catalog/product');
```

getModel() 👀

MODEL

Mage::getModel()

```
public static function getModel($modelClass = '', $arguments = array())
{
    return self::getConfig()->getModelInstance($modelClass, $arguments);
}
```

getModelInstance() 👀

MODEL

Mage_Core_Model_Config::getModelInstance()

```
public function getModelInstance($modelClass='', $constructArguments=array())
{
    $className = $this->getModelClassName($modelClass);
    if (class_exists($className)) {
        $obj = new $className($constructArguments);
        return $obj;
    } else {
        return false;
    }
}
```

getModelClassName() 👀

Mage_Core_Model_Config::getModelClassName()

```
public function getModelClassName($modelClass)
{
    $modelClass = trim($modelClass);
    if (strpos($modelClass, '/')===false) {
        return $modelClass;
    }
    return $this->getGroupedClassName('model', $modelClass);
}
```

getGroupedClassName() 👀

Mage_Core_Model_Config::getGroupedClassName()-1

```php
public function getGroupedClassName($groupType, $classId, $groupRootNode=null)
{
    if (empty($groupRootNode)) {
        $groupRootNode = 'global/'.$groupType.'s';
    }

    $classArr = explode('/', trim($classId));
    $group = $classArr[0];
    $class = !empty($classArr[1]) ? $classArr[1] : null;

    if (isset($this->_classNameCache[$groupRootNode][$group][$class])) {
        return $this->_classNameCache[$groupRootNode][$group][$class];
    }

    $config = $this->_xml->global->{$groupType.'s'}->{$group};
}
```

getGroupedClassName() 👀

Mage_Core_Model_Config::getGroupedClassName()-2

```
// First - check maybe the entity class was rewritten
$className = null;
if (isset($config->rewrite->$class)) {
    $className = (string)$config->rewrite->$class;
} else {
    if ($config->deprecatedNode) {
        $deprecatedNode = $config->deprecatedNode;
        $configOld = $this->_xml->global->{$groupType.'s'}->$deprecatedNode;
        if (isset($configOld->rewrite->$class)) {
            $className = (string) $configOld->rewrite->$class;
        }
    }
}
```

getGroupedClassName() 👀

Mage_Core_Model_Config::getGroupedClassName() - 3

```
// Second - if entity is not rewritten then use class prefix to form class name
if (empty($className)) {
    if (!empty($config)) {
        $className = $config->getClassName();
    }
    if (empty($className)) {
        $className = 'mage_'.$group.'_'.$groupType;
    }
    if (!empty($class)) {
        $className .= '_'.$class;
    }
    $className = uc_words($className);
}

$this->_classNameCache[$groupRootNode][$group][$class] = $className;
return $className;
```

BONUSTRACK

URBAN LEGEND

Der CronJob triggert keine ModulUpdates

```
// cron.php
Mage::app(); //ruft nicht applyAllUpdates() auf
```

nicht wie hier:

```
// index.php
Mage::run(); //ruft applyAllUpdates() auf
```

HIDDENTRACK


WAT!?

Bei Layout XML haben Blöcke

NAME (*name="..."*)

und ALIAS (*as="..."*)

fine

Block durch Ausschneiden nicht mehr ausgeben

<action method="_unsetChild_"> nutzt Alias

auch fine

Block durch Einfügen doch wieder ausgeben

<action method="_insert_"> nutzt Name

WAT!?

NEWSLETTER MODULE

```
# …/core/Mage/Adminhtml/controllers/Newsletter/QueueController.php
# Zeile 173

// Todo: put it somewhere in config!
$countOfQueue  = 3;
$countOfSubscritions = 20;
```

WAT!?

Wir definieren ein Array

```
# …/core/Mage/Core/Controller/Varien/Exception.php
# Zeile 37

protected $_resultCallbackParams = array();
```

fine

Später casten wir das Array dann aber einfach mal zu einer Methode

```
$this->_resultCallbackParams($path, $arguments);
```

WAT!?

WAT!?

- template_filter (war das vllt email filter?)
  - worum gings da nochmal?!

ARTIKELIMPORT – BILDDATEN

- $productData['_media_lable']
  - lable statt label


WAT!?

ARTIKELIMPORT – LOGGING

```
# …/core/Mage/ImportExport/Model/Abstract.php
# Zeile 89

$dirPath = Mage::getBaseDir('var') . DS . Mage_ImportExport_Model_Scheduled_Operation::LOG_DIRECTORY
```

Die Datei `Scheduled/Operation.php`
fehlt komplett; die gabs wohl auch noch nie!


I CAN'T EVEN ...

ATTRIBUTE ZU GRUPPEN HINZUFÜGEN

function addAttributeToGroup()

```
# …/core/Mage/Eav/Model/Entity/Setup.php
# Zeile 1081

…

$data = array(
    'entity_type_id'        => $entityType,
    'attribute_set_id'      => $setId,
    'attribute_group_id'    => $groupId,
    'attribute_id'          => $attributeId,
#   $data['sort_order']     =  $sortOrder; // kommt aus Zeile 1128
);

…

$this->getConnection()->insert(…, $data);
```

ATTRIBUTE ZU GRUPPEN HINZUFÜGEN

function addAttributeToSet()

```
# …/core/Mage/Eav/Model/Entity/Setup.php
# Zeile 1032

…

$data = array(
    'entity_type_id'        => $entityTypeId,
    'attribute_set_id'      => $setId,
    'attribute_group_id'    => $groupId,
    'attribute_id'          => $attributeId,
    'sort_order'            => $this->getAttributeSortOrder(…),
);

…

$this->_conn->insert(…, $data);
```

OH YEAH ...
I MISSED THAT!

my bad!

FACTORY METHODEN

MODELS

Mage::getModel(); Mage::getResourceModel();

SINGLETONS

Mage::getSingleton(); Mage::getResourceSingleton()

FACTORY METHODEN

HELPER

Mage::helper(); Mage::getResourceHelper();

WAT!?


I MEAN
DOUBLEWAT!?

UNGLEICHE TAGS FÜR GLEICHE FUNKTION

observer nutzt <class> und <method>

```
<catalog_wysiwyg>
    <class>catalog/observer</class>
    <method>catalogCheckIsUsingStaticUrlsAllowed</method>
</catalog_wysiwyg>
```

UNGLEICHE TAGS FÜR GLEICHE FUNKTION

cron run nutzt <model> mit group/model::method

ist aber nicht static, *auch wenn es so aussieht*

```
<run>
    <model>catalog/observer::reindexProductPrices</model>
</run>
```

THAT'S ACTUALLY ...

WO SCHAUEN DIE EIGENTLICH HIN???

ATTRIBUTE

ANLEGEN UND ÄNDERN

addAtribute verwendet visible = ...
updateAttribute verwendet is_visible = ...


WAT IS _ LOS!?

EIN STORE IST EIN STORE

IST EIN STORE IST EIN STORE

im Backend StoreView heißt Store in PHP
im Backend Store heißt StoreGroup in PHP

STORE !== STORE !?!??!

WWWWT!?

QUELLEN

- forward vs redirect
- http://www.giantgeek.com/blog/?p=109
- http://www.javapractices.com/topic/TopicAction.do?Id=181
- Collection filtern
- http://www.webguys.de/magento/turchen-11-eine-collection-filtern/

DANKE