










Welcome to the
Hitchhiker's Guide to FLOW3



Hitchhiker's Guide to FLOW3

Inspiring people to
share

Overview

-  Hello FLOW3!
-  Configuration
-  Bootstrap
-  Packages
-  Components
-  Caching
-  More



Getting Started



Hello FLOW3!

The Hello World example based on FLOW3's
Model-View-Controller Framework.

Hitchhiker's Guide to FLOW3

Inspiring people to
share

5 Easy Steps

1. Download FLOW3
2. Adjust write permissions
3. Create a new package
4. Create a default controller
5. Create a default action

1. Download FLOW3

Just checkout the FLOW3 distribution via Subversion:

```
svn co http://svn.typo3.org/FLOW3/dist/trunk/
```

```
robsmac:/ robert$ _
```

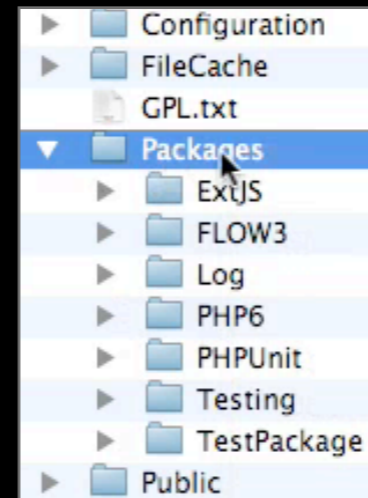
2. Adjust write permissions

Make sure that the public folder is writeable for the webserver's user:

```
sudo chown -R robert:www public/  
sudo chmod -R 770 public/
```

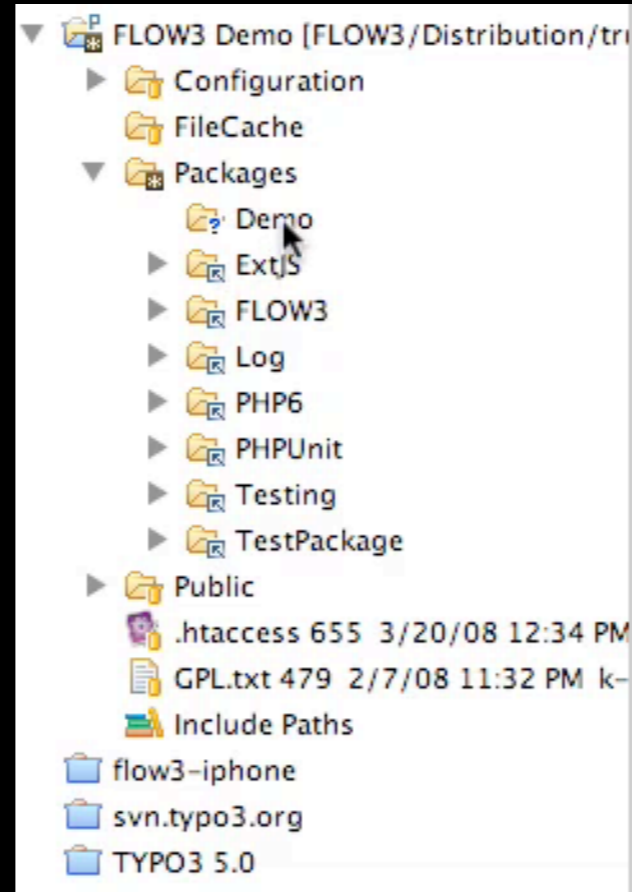
3. Create a package

In order to create a new package, just create a new folder within the Packages directory.



4. Create a Default Controller

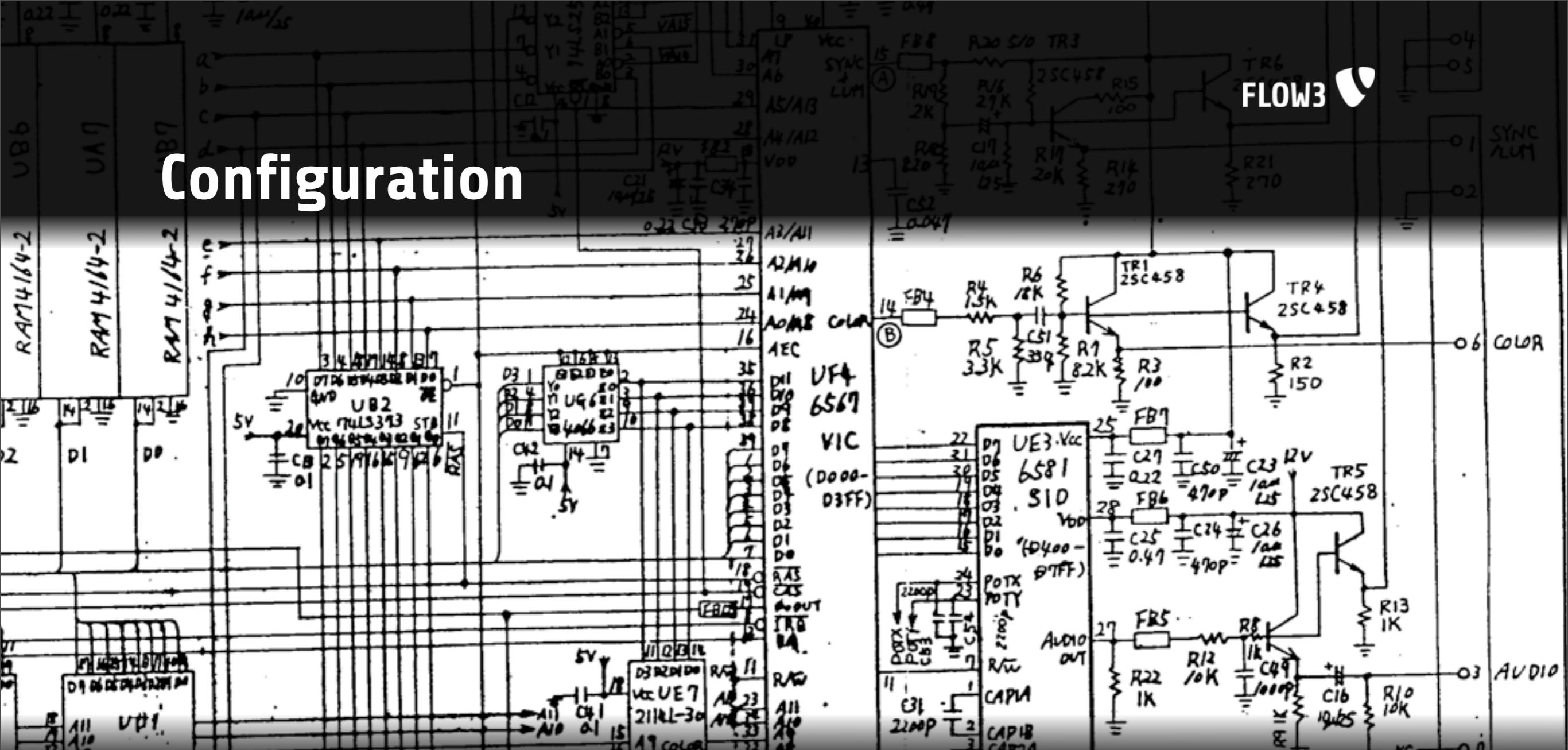
1. Create a "Classes" directory
2. Create a "Controller" directory
3. Create a class file
4. Extend FLOW3's action controller



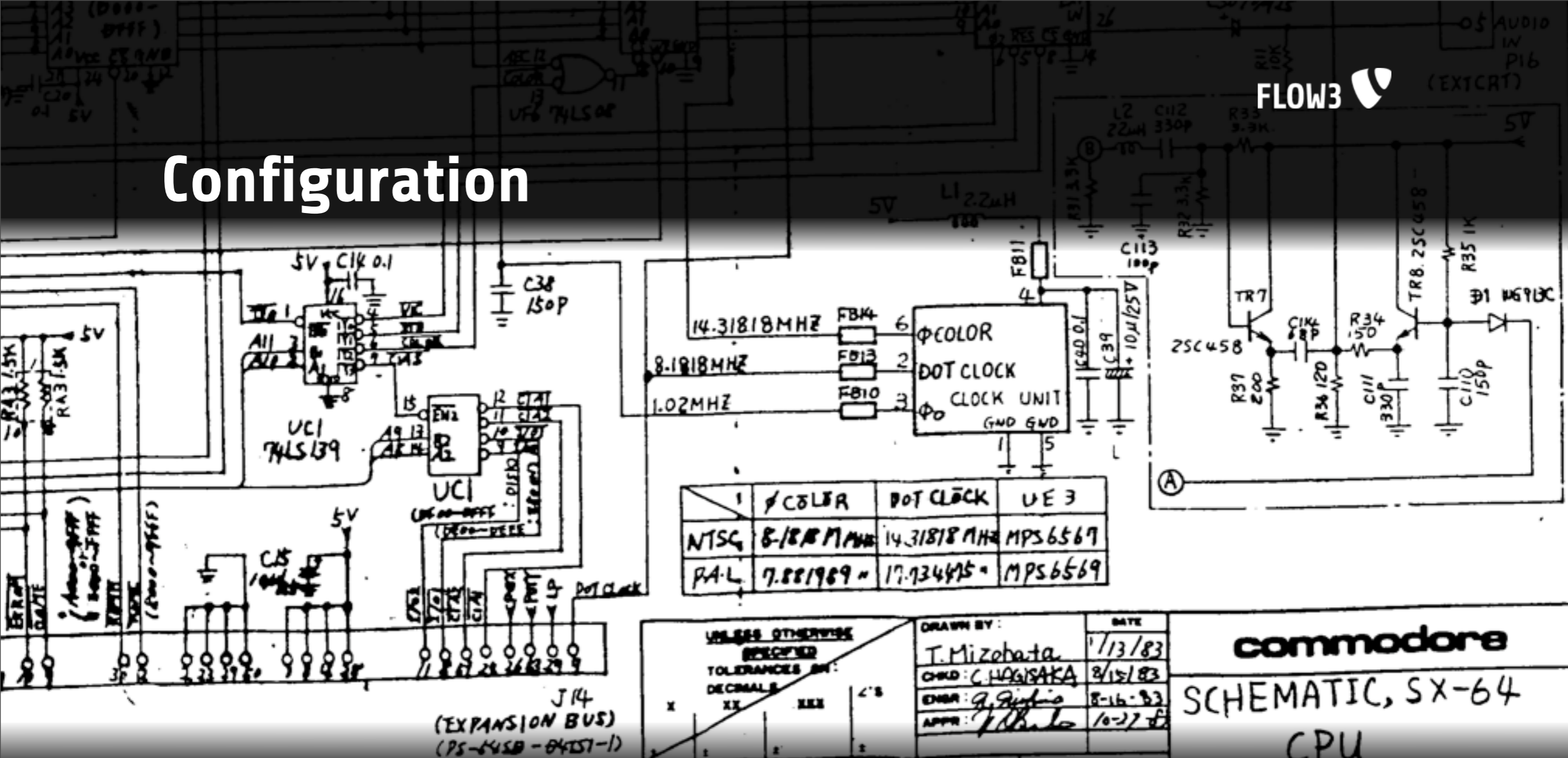
5. Create Default Action

```
F3_Demo_Controller_Default.php X
1 <?php
2
3 class F3_Demo_Controller_Default extends F3_FLOW3_MVC_Controller_ActionController {
4
5
6
7 }
8 ?>
```

Configuration



Configuration



Configuration Format

- ❖ The default configuration format is PHP
- ❖ Configuration options reside in a configuration object
- ❖ The configuration object provides array access and a fluent interface
- ❖ Configuration options are self-documenting

Configuration Format

```
1 <?php
2 declare(ENCODING="utf-8");
3
4 /*.....*
5 *. Configuration for the FLOW3 Framework.....*
6 *.....*
7 *. This file contains the default base configuration for the FLOW3.....*
8 *. Framework. Don't modify this file but add configuration options to.....*
9 *. the FLOW3.php file in the in global Configuration/ directory instead...*
10 *.....*/
11
12 /**
13 *. @package FLOW3
14 *. @version $Id: FLOW3.php 689-2008-04-03 10:57:33Z robert $
15 */
16
17 /**
18 *. Defines the global, last-resort exception handler.
19 */
20 *. @type F3_FLOW3_Error_DevelopmentExceptionHandlerInterface
21 */
22 $c->exceptionHandler->className = 'F3_FLOW3_Error_ProductionExceptionHandler';
```

Configuration Types

- ❖ FLOW3 distinguishes between different configuration types for different purposes:
 - FLOW3 - reserved for FLOW3 configuration
 - Package - package related configuration
 - Component - configuration for components, including Dependency Injection
 - Routes - special configuration for defining MVC routes
 - Settings - mainly user-level settings for any purpose

Configuration Types

```
Components.php 689 4/3/08 12:57 PM robert
FLOW3.php 689 4/3/08 12:57 PM robert
Packages.php 689 4/3/08 12:57 PM robert
README 645 3/18/08 12:22 PM robert
Routes.php 689 4/3/08 12:57 PM robert
Settings.php 689 4/3/08 12:57 PM robert
```


The Cascade

- ❖ Each package defines possible configuration options by setting default values
- ❖ Default configuration can be altered by user-defined configuration files
- ❖ User configuration can only modify existing configuration options
- ❖ Modifying non-existent configuration options results in an error

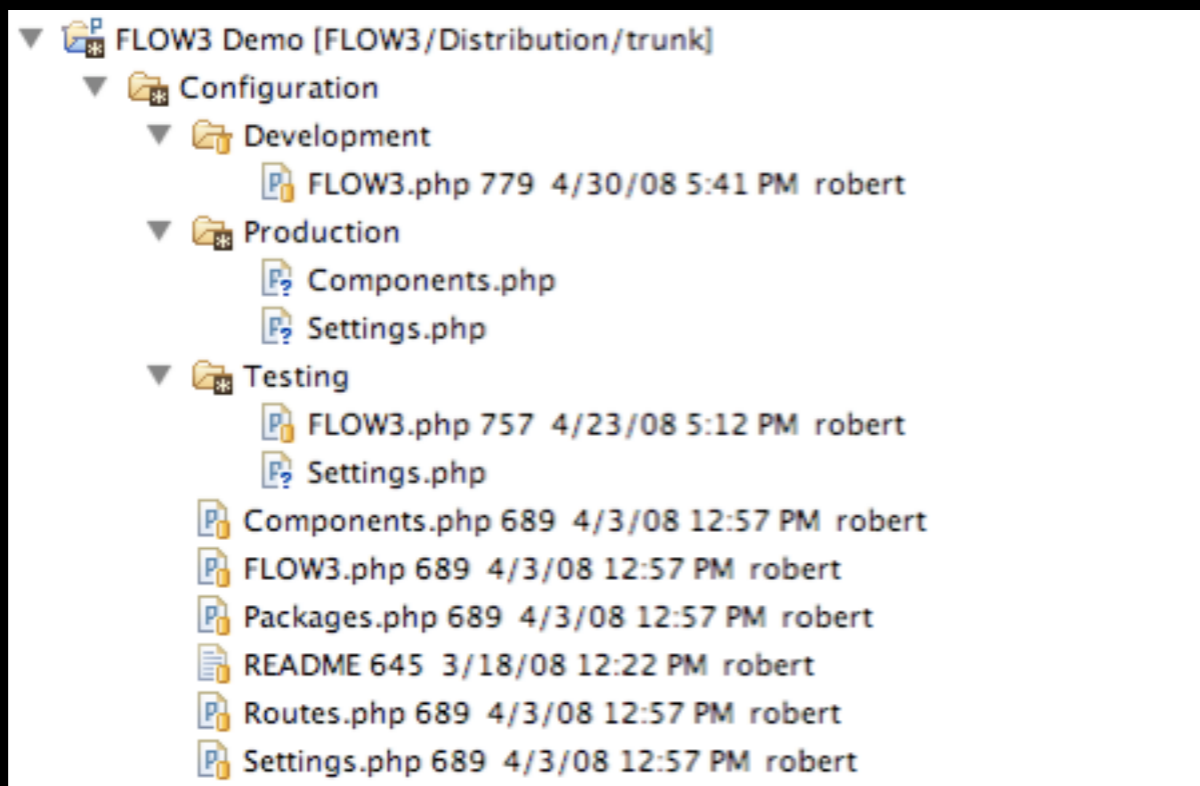
Application Context

- ☛ An application context is a set of configuration for a specific context
- ☛ FLOW3 is shipped with configuration for these contexts:
 - Production
 - Development
 - Testing
 - Staging
- ☛ FLOW3 is always launched in one defined context
- ☛ Additional, user-defined contexts are possible

Application Context

- ❖ Configuration defined in the top level of a *Configuration* directory is the base configuration
- ❖ Specialized configuration for application contexts reside in subdirectories named after the context
- ❖ Application context configuration overrides the base configuration

Application Context



Bootstrap (or: how to launch a rocket)



Hitchhiker's Guide to FLOW3

Inspiring people to
share

Public/index.php

- ☛ This file is the default main script
- ☛ It launches FLOW3 in the **Production** context
- ☛ The webserver's web root should point to the **Public** directory

```
define('FLOW3_PATH_PUBLIC', str_replace('\\', '/', dirname(__FILE__)). '../');  
require_once(FLOW3_PATH_PUBLIC. '../Packages/FLOW3/Classes/F3_FLOW3.php');  
  
$framework = new F3_FLOW3();  
$framework->run();
```

Public/index_dev.php

- ☛ This script is used for development
- ☛ It launches FLOW3 in the *Development* context
- ☛ More scripts like this can be created for additional contexts

```
define('FLOW3_PATH_PUBLIC', str_replace('\\', '/', dirname(__FILE__)).../'/');  
require_once(FLOW3_PATH_PUBLIC..'../Packages/FLOW3/Classes/F3_FLOW3.php');  
  
$framework = new F3_FLOW3('Development');  
$framework->run();
```

Public/index_dev.php

- ❖ Don't forget to run FLOW3 in **Development** context while you're developing because
 - component configuration is cached in production mode, so new classes won't be recognized
 - resources are cached in production mode, so changes won't be detected
 - and many more things might be cached which lead to unexpected errors if you change some code in your package

\$FLOW3->run()

- ♥ *run()* is a convenience method which
 - initializes the FLOW3 framework
 - resolves a request handler
 - handles and responses to the request

\$FLOW3->initialize()

- ☛ The initialization process is divided into different stages:
 - Initialize FLOW3
 - Initialize the packages
 - Initialize the components
 - Initialize the settings
 - Initialize the resources
- ☛ The configuration for each level can't be changed once the initialization level is reached

Packages

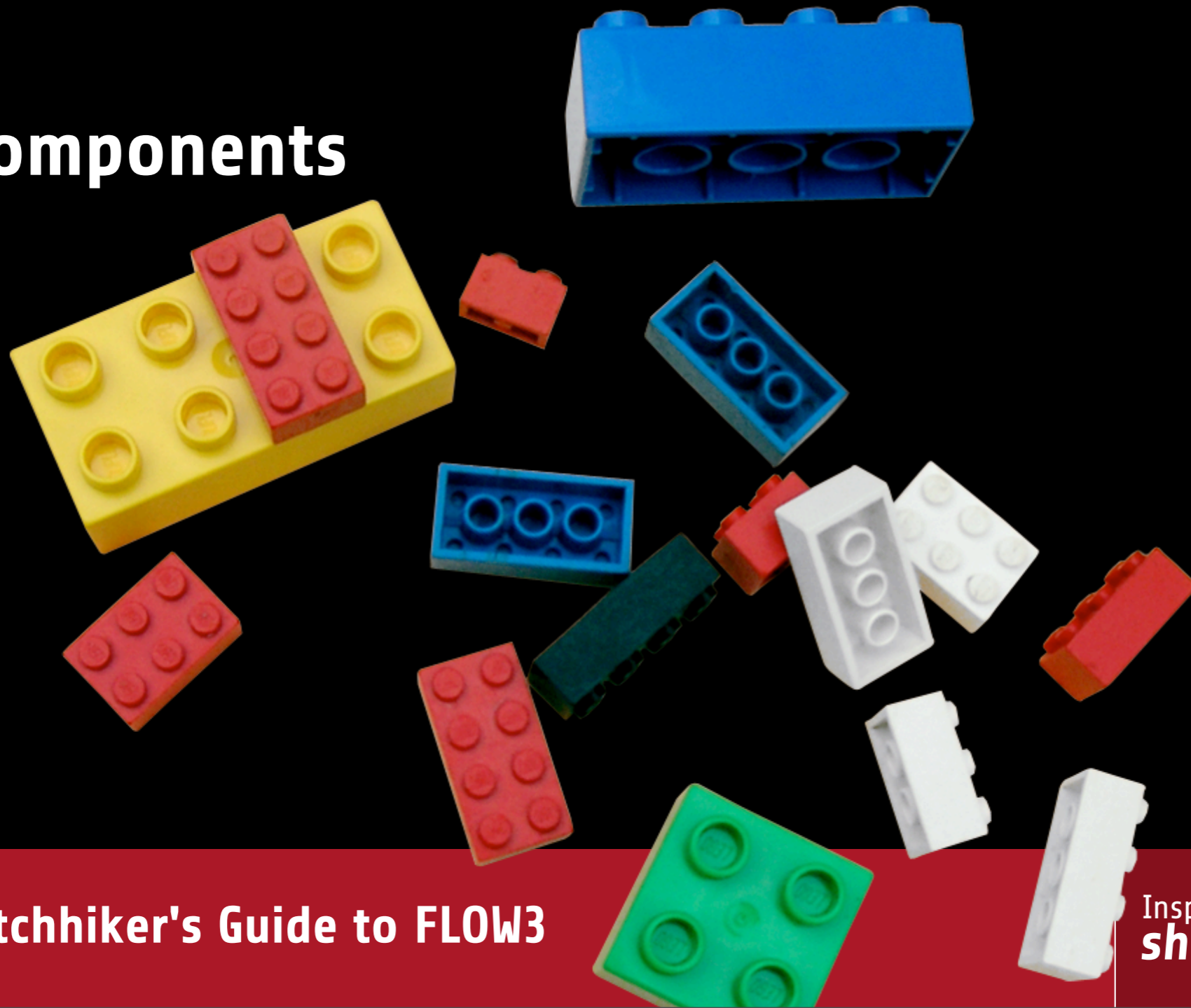
Package Manager

- ✔ Like the good old Extension Manager - but without UI yet
- ✔ Scans the *Packages* directory for packages
- ✔ Will connect to the FLOW3 Package Repository
- ✔ Package file format is just plain .zip
- ✔ Will provide access via Web / CLI and offer Web Services

Meta/Package.xml

- ❖ Contains meta information about a FLOW3 package
- ❖ The format is defined by a RelaxNG schema:
<http://typo3.org/ns/2008/flow3/package/Package.rng>
- ❖ The Package.xml will soon be mandatory

Components



Hitchhiker's Guide to FLOW3

Inspiring people to
share



Components

- ❖ Components are re-usable, properly encapsulated objects
- ❖ The lifecycle of a component and the combination of active components is managed by the **Component Manager**
- ❖ All classes in the TYPO3 context are considered as components
- ❖ Components are configurable



Class $\hat{=}$ Component

- ❖ Classes are automatically registered as components if
 - they reside in the **Classes** directory of a package and
 - their name follows the FLOW3 naming conventions



Example

```

1 <?php
2 declare(ENCODING="utf-8");
3
4 /*.....**
5 * This script is part of the TYPO3 project -- inspiring people to share! ..**
6 *.....**
7 * TYPO3 is free software; you can redistribute it and/or modify it under
8 * the terms of the GNU General Public License version 2 as published by
9 * the Free Software Foundation.....**
10 *.....**
11 * This script is distributed in the hope that it will be useful, but
12 * WITHOUT ANY WARRANTY; without even the implied warranty of MERCHAN
13 * TABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General
14 * Public License for more details.....**
15 *.....**/
16
17 /**
18 * @package Demo
19 * @version $Id: F3_Demo_Controller_Default.php.123.2008-01-01.12:00:00Z.robert.$
20 */
21
22 /**
23 * The Default Controller of the Demo package
24 **
25 * @package Demo
26 * @version $Id: F3_Demo_Controller_Default.php.123.2008-01-01.12:00:00Z.robert.$
27 */
28 class F3_Demo_Controller_Default {
29
30 }
31
32 ?>

```



Playing with building blocks

- ❖ The combination of components used is configurable (orchestration)
- ❖ The less components know about each other the easier it is to reuse them in a variety of contexts
- ❖ Create your own LEGO set by creating cleanly separated, decoupled components!



Component Dependencies

- ❖ Components seldomly come alone
- ❖ Components depend on other components which depend on other components which ...
- ❖ Problem:
 - Components explicitly refer to other components:
`$phoneBookManager = new PhoneBookManager`



Dependency Injection

- ❖ A component doesn't ask for the instance of another component but gets it *injected*
- ❖ This methodology is referred to as the "Hollywood Principle":
"Don't call us, we'll call you"
- ❖ Enforces loose coupling and high cohesion
- ❖ Makes you a better programmer

Constructor without Dependency Injection

```
/**  
 * Some demo component  
 */  
@package Demo  
*/  
class F3_Demo_Foo {  
    protected $cacheManager;  
    /**  
     * Constructs the demo component  
     */  
    public function __construct() {  
        $this->cacheManager = F3_FLOW3_Cache_Manager::getInstance();  
    }  
}
```

Component with Constructor Injection

```
/**  
 * Some demo component  
 */  
@package Demo  
*/  
class F3_Demo_Foo {  
    protected $cacheManager;  
    /**  
     * Constructs the demo component  
     */  
    public function __construct(F3_FLOW3_Cache_Manager $cacheManager) {  
        $this->cacheManager = $cacheManager;  
    }  
}
```

Component with Setter Injection

```
/**  
 * Some demo component  
 */  
@package Demo  
*/  
class F3_Demo_Foo {  
    protected $cacheManager;  
    /**  
     * Injects the cache manager  
     */  
    public function injectCacheManager(F3_FLOW3_Cache_Manager $cacheManager) {  
        $this->cacheManager = $cacheManager;  
    }  
}
```

Autowiring

- ❖ FLOW3's framework tries to autowire constructor arguments and arguments of *inject** methods
- ❖ The type of the component to be injected is determined by the argument type (type hinting)
- ❖ Autowiring does not work with Setter Injection through regular setters (*set** methods)
- ❖ Dependencies are only autowired if no argument is passed explicitly

Fetching components manually

- ❖ Although Dependency Injection is strongly recommended, there might be cases in which components need to be created or retrieved manually
- ❖ Use the `getComponent()` method in these cases.

```
$component = $componentManager->getComponent($componentName, $arg1, $arg2, ...);
```

Component scope

- ❖ Component objects always live in a certain scope
- ❖ Currently supported scopes are:
 - Singleton - Only one instance exists during one script run
 - Prototype - Each ***GetComponent()*** call returns a fresh instance

Component scope

- ☛ The scope can be defined through
 - an annotation in the component class (recommended)
 - through the component configuration in a ***Components.php*** file
- ☛ The default scope is "Singleton"

Component scope

```
1 <?php
2
3 /**
4  * This is a singleton component
5  *
6  * @package Demo
7  * @scope singleton
8  */
9 class F3_Demo_SomeSingleton {
10
11 }
12
13 ?>
```

```
1 <?php
2
3 /**
4  * This is a prototype component
5  *
6  * @package Demo
7  * @scope prototype
8  */
9 class F3_Demo_SomePrototype {
10
11 }
12
13 ?>
```

Creating Prototypes

- ☛ Dependency Injection can be used in almost any case, there's no need to call `getComponent()`
- ☛ But what if you need to instantiate a component **within** a method?


```
1 <?php
2
3 /**
4  * @package Demo
5  */
6 class F3_Demo_Controller_Phonebook {
7
8     public function addEntryAction() {
9         ...
10        $phoneBookEntry = new F3_Demo_Model_PhonebookEntry();
11        ...
12    }
13 }
```

Creating Prototypes

☛ Solution A: Call `getComponent()`

```
1 <?php
2
3 /**
4  * @package Demo
5  */
6 class F3_Demo_Controller_Phonebook {
7
8     public function addEntryAction() {
9         ...
10        $phoneBookEntry = $this->componentManager->getComponent('F3_Demo_Model_PhonebookEntry');
11        ...
12    }
13 }
14
15 ?>
```

Creating Prototypes

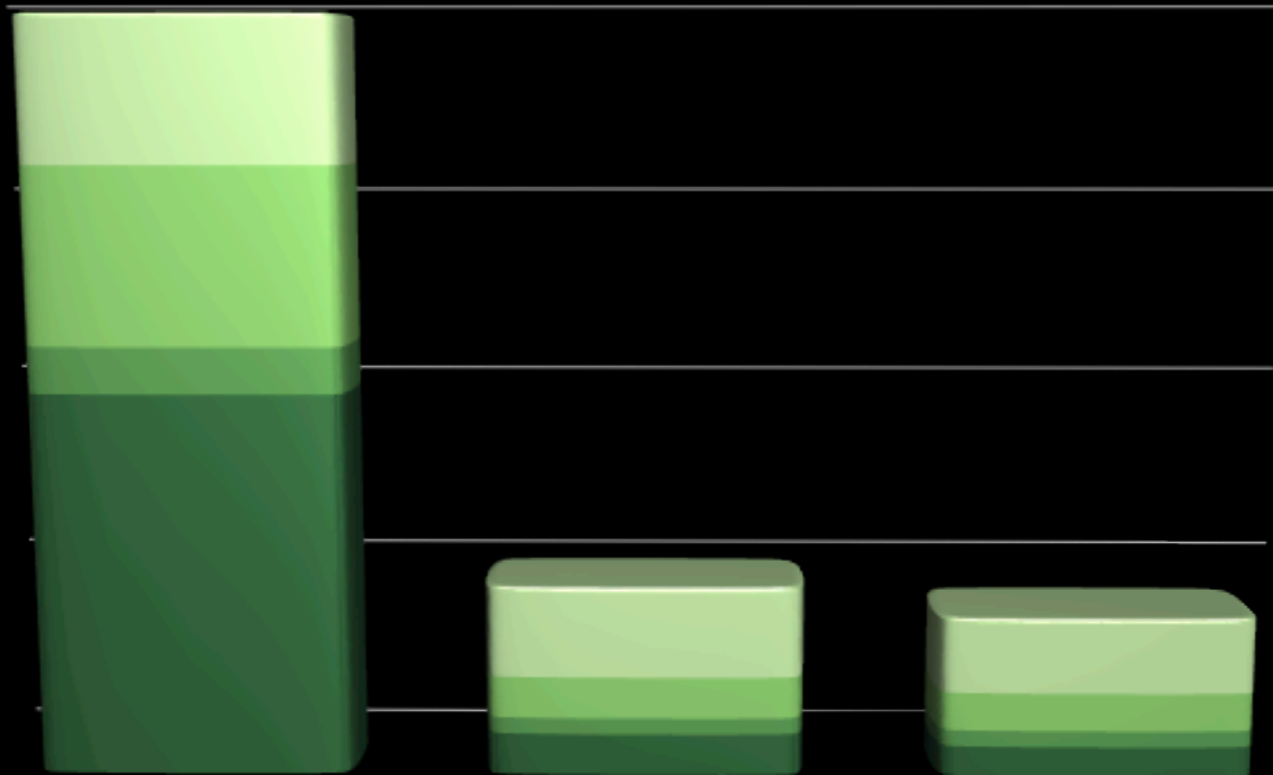
 Solution B: Call a factory method

```
1 <?php
2
3 /**
4  * @package Demo
5  */
6 class F3_Demo_Controller_Phonebook {
7
8     public function addEntryAction() {
9         ...
10        $phoneBookEntry = $this->createPhonebookEntry();
11        ...
12    }
13
14    /**
15     * @return F3_Demo_Model_PhoneBookEntry
16     */
17    abstract protected function createPhonebookEntry();
18 }
19
```

Creating Prototypes

- Planned feature: Automatically generated factory methods

Caching



Caching

- ❖ FLOW3 comes with a generic caching mechanism
- ❖ Different kinds of cache frontends (aka "Caches") are supported:
 - Variable cache: Caches all kinds of variables, including objects
 - File cache: Is optimized for caching files
- ❖ Various kinds of cache backends (aka "Storages") can be used:
 - File backend: Store cache content in files
 - Memcached backend: Store cache content in memory
- ❖ More frontends and backends are planned
- ❖ User-defined frontends and backends can be used as well

Cache Configuration Example

- ❖ The component configuration is cached in **Production** context
- ❖ This is achieved by enabling the cache in the production configuration

```
/**  
 * Enable or disable caching of the component configurations. If caching is  
 * enabled, a cache backend must be properly configured.  
 */  
 * @type boolean  
 */  
 $c->component->configurationCache->enable = TRUE;  
  
/**  
 * Define the backend used for caching component configurations. Specify the  
 * name of a component implementing the F3_FLOW3_Cache_BackendInterface.  
 */  
 * @type F3_FLOW3_Cache_BackendInterface  
 */  
 $c->component->configurationCache->backend = 'F3_FLOW3_Cache_Backend_File';  

```

Cache Files Example

```
robsmac:/tmp/FLOW3 robert$ _
```

How to Cache

- ❖ Create a new cache frontend – backend pair
- ❖ Configure the frontend as necessary
- ❖ Store data using the frontend's API
- ❖ Retrieve data using the frontend's API

Caching



How to Cache

DEMO

Hitchhiker's Guide to FLOW3

Inspiring people to
share

Cache Manager

- ❖ Provides a registry for reusing caches
- ❖ Caches are registered through the ***registerCache()*** method and can be retrieved again by calling the ***getCache()*** method
- ❖ Caching can be done without the Cache Manager, too. Registration is not mandatory and only needed if you want to share the cache object among different places

Caching



How Use the Cache Manager

DEMO

Hitchhiker's Guide to FLOW3

Inspiring people to
share

More ...

SYNTAX

```
more [-dlfpcsu] [-num] [+ / pattern] [+ linenum] [file ...]
```

OPTIONS

Command line options are described below. Options are also taken from the environment variable MORE (make sure to precede them with a dash ('`-`')) but command line options will override them.

- num This option specifies an integer which is the screen size (in lines).
- d more will prompt the user with the message "[Press space to continue, 'q' to quit.]" and will display "[Press 'h' for instructions.]" instead of ringing the bell when an illegal key is pressed.
- l more usually treats ^L (form feed) as a special character, and will pause after any line that contains a form feed. The -l option will prevent this behavior.
- f Causes more to count logical, rather than screen lines (i.e., long lines are not folded).
- p Do not scroll. Instead, clear the whole screen and then display the text.

More ...

Coding Guidelines

- 👉 Malte and Tim create the FLOW3CGL package
- 👉 CGL document will be on forge.typo3.org soon

More ...



DEV3

TYPO3 Enterprise Development and TypoScript Editor for Eclipse – DEV3

http://www.dev3.org/

DEV3
TYPO3 ENTERPRISE DEVELOPMENT

Deutsch

DEV₃ - TYPO3 Enterprise Development

Our Mission

"SweeTS -delicious TypoScript Development", developed by Eckhard M. Jäger, was the first real development application for TYPO3 und Typoscript. "SweeTS" offers many features for a rapid and secure TypoScript development. SweeTS is not yet totally independent from operating systems and does not offer all interfaces that an enterprise development tool needs.

With the DEV₃ project we answer the call of many developers to make all ideas and possibilities of "SweeTS" available on an Eclipse system. Together with the [FLOW3DE-Projekt](#) we want to create not only an open source solution for the whole TYPO3/ TypoScript and PHP development that is independent of operating systems, we want to create a solution for the upcoming and modern framework [FLOW3](#). too. We will develop DEV₃ on the base of the [PHP Developer Tools \(PDT\)](#) which can easily supplemented by the [Aptana-Studio](#).

Wee need you!

If you are a TYPO3 developer you can help us in different ways:

- Donate money for a professional and fast development of DEV₃
- Send us your commented TypoScript code to integrate as DEV₃ code snippets
- Share your professional ideas for DEV₃ with us
- Help our development by sharing your professional Java knowledge
- Be our alpha and beta tester

Support DEV₃

Spenden

MasterCard VISA givemon PayPal

...10,-Euro can make a difference.

TYPO3 CMS

TYPO3
DEV₃ at TYPO3 Forge

Powered by

JRE42
Agentur & Systempartner

CCM
Cross Content Media

Inspiring people to
share

More ...



AOP Browser

Component Browser

- Find a component
- AOPBrowser
- FLOW3
- Log
- PHP6
- PHPUnit
- Testing
- TestPackage
 - Controller
 - SubDirectory
 - View
 - AspectForClassWithOneConstructorArgument
 - AspectForClassWithOptionalConstructorArgument
 - AspectForClassWithStaticFunction
 - BasicClass
 - BasicClassInterface
 - ClassToBeReplaced
 - ClassWithFinalConstructor
 - ClassWithOneConstructorArgument
 - ClassWithOptionalConstructorArgument
 - ClassWithSomeImplementationInjected
 - ClassWithStaticFunction
 - ClassWithUnmatchedRequiredSetterClass
 - FinalClass
 - GetSomeChinesePropertyAspect
 - InjectedClass
 - InjectedClassWithDependencies

```
<?php
declare(ENCODING = 'utf-8');

/*
 * This script is part of the TYPO3 project -
 * inspiring people to share! *
 *
 * TYPO3 is free software; you can redistribute it and/or modify
 * the terms of the GNU General Public License version 2 as published
 * by the Free Software Foundation.
 *
 * This script is distributed in the hope that it will be useful,
 * WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
 * General Public License for more details.
 *
 *
 *
 */
/**
 * Fixture class for testing the AOP framework
 *
 * @package      TestPackage
 * @version      $Id: F3_TestPackage_GetSomeChinesePropertyAspect.php 103-25 14:03:49Z robert $
 * @author      Robert Lemke <robert@typo3.org>
 * @copyright    Copyright belongs to the respective authors
 * @license      http://opensource.org/licenses/gpl-
 *               license.php GNU Public License, version 2
 * @aspect
 */
class F3_TestPackage_GetSomeChinesePropertyAspect implements F3_TestPackage_GetSomeChinesePropertyAspectInterface
```

Outline

- F3_TestPackage_GetSomeChinesePropertyAspect
 - afterReturningFromFourtyTwo
 - Advises methods:
 - getSomeProperty
 - afterThrowing
 - aroundAroundFourtyTwoToChinese
 - Advises methods:
 - getSomeProperty
 - aroundFourtyTwoToChinese
 - Advises methods:
 - getSomeProperty
 - beforeFourtyTwoToChinese
 - getFlags

Inspiring people to
share

More ...

FLOW3 

Known Issues

- ❖ FLOW3 (or rather PHP) currently causes Apache crashes – why ever ...
- ❖ Tests consume a lot of memory (> 400 MB)
- ❖ Access is comparably slow even in Production context (~ 3 req/s) and needs much memory (~ 20 MB)
- ❖ Many aspects are work in progress and neither optimized nor finished



Hitchhiker's Guide to FLOW3

Inspiring people to
share

Links

- 👉 FLOW3 Website
<http://flow3.typo3.org>
- 👉 TYPO3 5.0 Subsite
<http://typo3.org/gimmefive>
- 👉 TYPO3 Forge
<http://forge.typo3.org>

So long and thanks for the fish



Questions

Hitchhiker's Guide to FLOW3

Inspiring people to
share

TYPO3

