Sylar Deploy Guide Line

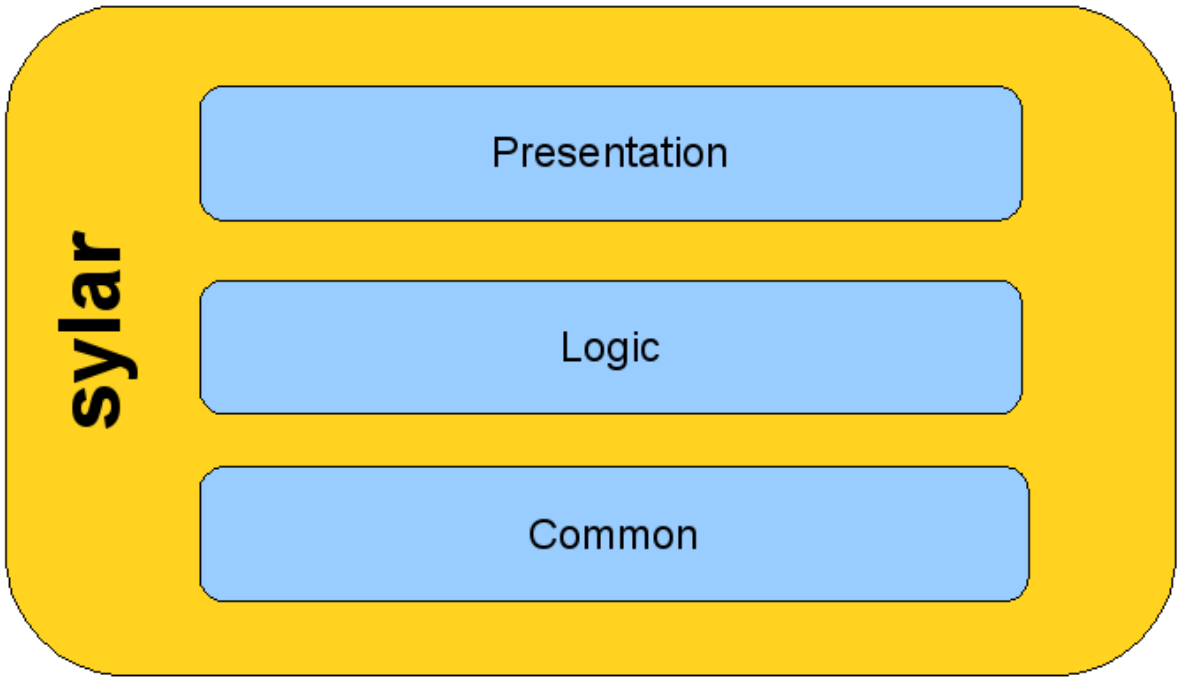
Sylar è un framework di sviluppo per applicazioni web PHP. Sylar è un progetto Open Source. Tutte le info sono reperibili presso la home del progetto:

https://launchpad.net/sylar/

Lo sviluppo è aperto a tutti coloro vogliono partecipare a vari livelli di competenze. Dalle semplici traduzioni allo sviluppo del codice.

Pattern

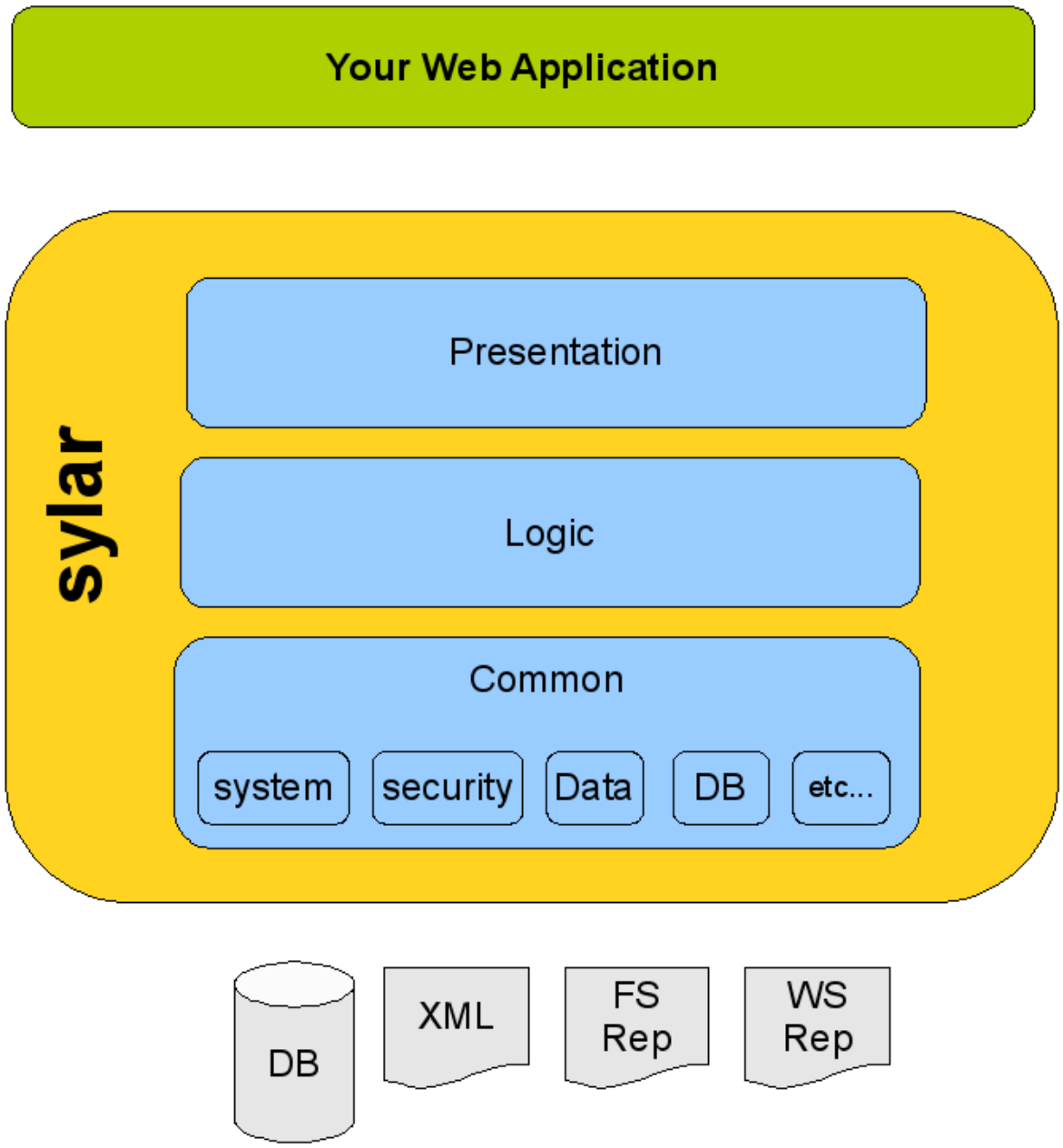
Sylar è sviluppato secondo un pattern molto simile all'MVC pensato per applicazioni Web tipo:



Il framework è stato disegnato per mantenere sempre separati i livelli di accesso ai dati/informazioni, la logica e la presentazione all'utente.
Si consiglia di mantenere lo stesso approccio nello sviluppo delle vostre applicazioni che utilizzeranno Sylar come base di sviluppo.

Esempio di applicazione

Più nel dettaglio lo scenario di una tipica applicazione web sviluppata con Sylar sarà:



Fondamentale è l'astrazione dell'applicazione dai dati e delle informazioni che possono essere forniti o contenuti da fonti diverse e che possono cambiare nel tempo.

Una volta reperiti i dati sarà compito delle classi della logica manipolarli e prepararli per una futura presentazione.

La pubblicazione dei dati ottenuti dalla logica sarà compito delle classi di presentation che potranno mostrare le info in una pagina web, in una mail, in un file, ecc...

Coding rules

Di seguito vengono elencate le regole imprescindibili a cui ci si deve attenere per poter partecipare allo sviluppo di sylar. Le regole sono importanti perchè definiscono degli standard che tutti conoscono e sono abituai a leggere, questo facilita e velocizza lo sviluppo e la comprensione del codice scritto da altri.

Source code documentation

La documentazione nel codice è importante quanto il codice. Non verranno presi in considerazione sorgenti scritti e proposti senza documentazione.
La documentazione va realizzata secondo le regole della PhpDoc simile a JavaDoc.
Il modo più semplice per imparare come documentare un file in sylar è guardare come sono documentati quelli esistenti.

Un ottimo aiuto è fornito dalle IDE di sviluppo evolute come ad esempio Eclipse, Zend Studio (non free), etc... che facilitano anche la scritttura dei tag di documentazione oltre al codice php.

Nel seguito saranno riportati i template di documentazione per ogni tipologia di oggetto come nel caso di classi, file, metodi, ecc...

Variable

Le variabili iniziano con la lettera minuscola.

Nel caso di variabili di tipo primitivo è buona norma far precedere la variabile da una lettera che indica il tipo di variabile, ad esempio:

```
$iNum = 100;                    // integer
$sName = "some text";           // string
$bIsSystemUp = true;            // boolean
$aDati = array();               // array
$oDbDriver = new Db();          // Object
$fLong = 10.56;                 // floating point aka double
$rResult = some_func_();        // resource
```

è importante documentare tutte le variabili nel codice, sia per comprendere la logica del programma sia per generare documentazione.

I commenti utili solo alla lettura del codice vanno inseriti in questo modo:

```
// Temp Var with name of user
$sNameUser = "Frank";
```

Mentre quelli mirati alla generazione della Documentazione sono questi:

```
/**
 * Email of user
 * @var string contains the user's email
 */
$sUserEmail = "pippo@pluto.com";
```

Classpath

Definisce la root dei package di sylar sul filesystem da cui il framework include le classi.
Vengono definiti nel file di configurazione:

sylar/settings/sylar.php

tramite le costante:

`SYLAR_CLASSES_ROOT_FS`

Il classpath dell'applicazione viene invece definito nel file di configurazione dell'applicazione ed è nel nostro esempio:

exampleApp/config/appConfig.php

tramite la costante:

`SYLAR_APPLICATION_CLASSES_ROOT_FS`

Package

Il package è composto da tutte lettere minuscole

Anche se non implementati fino alla versione 5.3 del PHP, Sylar utilizza una classificazione del codice con package. Un package è una subdirectory all'interno del classpath che contiene classi e/o altri package.
Ad esempio seguendo la struttura di Sylar il package:

`sylar.common.system`

corrisponde alla directory

SylarClasspath/common/system/

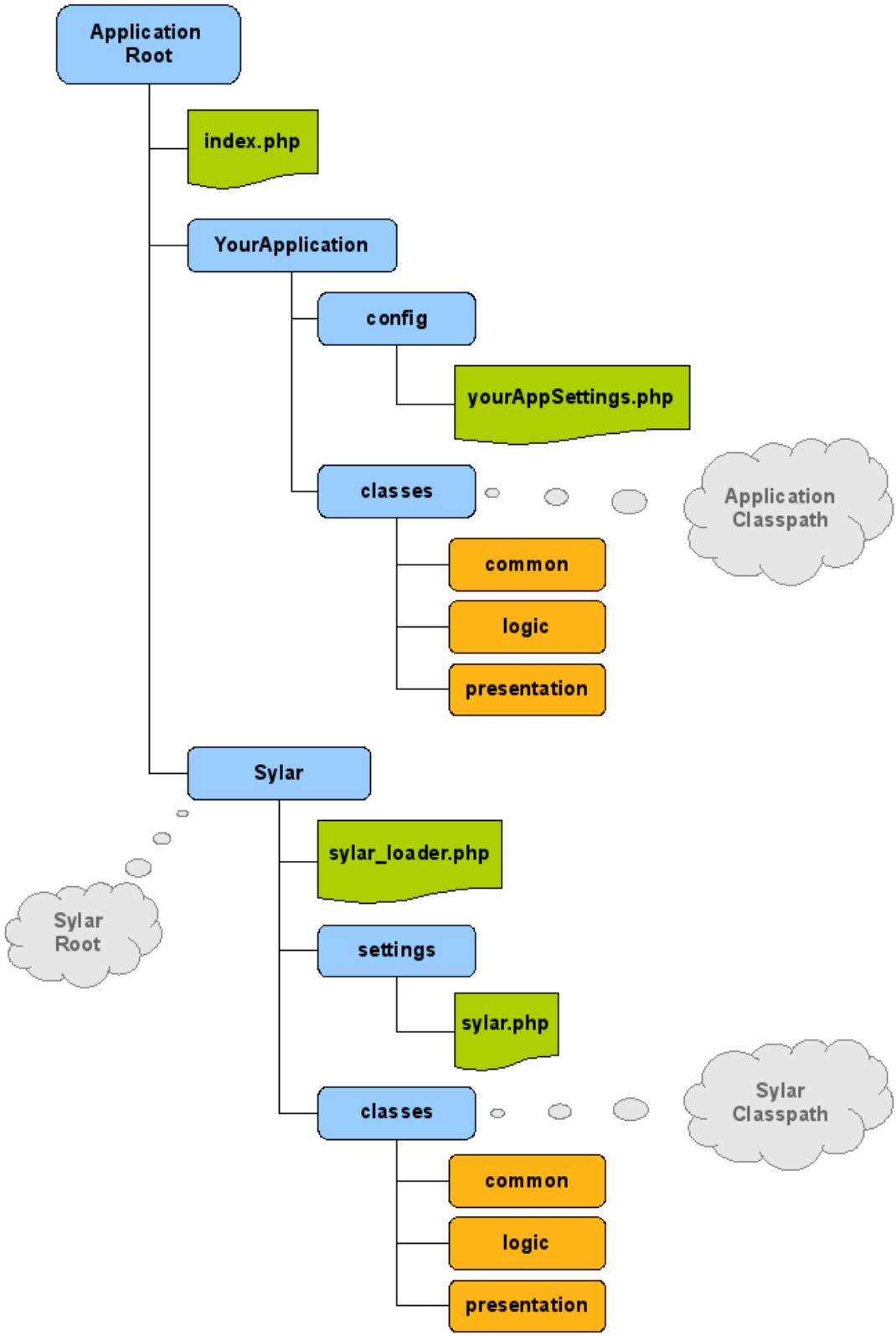
Se invece si fa riferimento ad un package relativo all'applicazione e non a sylar è possibile richiamarlo tramite il prefisso *app.* oppure omettendo il prefisso *sylar.* il framework punta di default al class path dell'applicazione.
Ad esempio il package:

`app.common.system.users`

Individuerà la directory su filesystem:

SylarClasspath/common/system/users/

Una struttura classica di filesystem per un'applicazione web che utilizza Sylar è rappresentata nella pagina seguente.


Application Root
index.php
YourApplication
config
yourAppSettings.php
classes
Application Classpath
common
logic
presentation
Sylar
Sylar Root
sylar_loader.php
settings
sylar.php
classes
Sylar Classpath
common
logic
presentation

Class

Il nome delle classi iniziano con la lettera maiuscole.

In questo modo è possibile riconoscere immediatamente una classe da un package o da un metodo che invece inizierà con lettera minuscola.
In particolare tutti le parole principali del nome della classe iniziano con maiuscolo. Ad esempio:

```
Sylar_SimpleTableHeader
```

Tutte le classi facenti parte di Sylar hanno il prefisso `Sylar_` seguito da il nome logico della classe che corrisponderà al nome del file su disco. Di regola ogni file di classe contiene una sola classe, o la classe principale che da il nome al file.
Nel caso in esempio il file si chiamerà:

SimpleTableHeader.php

> ***N.B.*** Nel caso di classi è possibile inserire nello stesso file più classi solo nel caso in cui la classe ospite è usata solo e unicamente dalla classe ospitante che da il nome al file.

Una classe è individuata insieme al suo package e viene importate nel framework per poter essere utilizzata. Un esempio di classe di sylar:

```
sylar.common.data.DataContainer
```

Un esempio di classe di tipo applicazione:

```
app.giano.presentation.FormatLogList
```

Documentazione di esempio:

```
/**
 * Configuration Box
 * A class that conteins all Sylar Configuration values read from
 * config and settings file.
 *
 * @package Sylar
 * @version 1.0
 * @since 16/feb/08
 * @author Gianluca Giusti [brdp] <g.giusti@giano-solutions.com>
 * @copyright Sylar Development Team
 */
```

Function and method

Il nome del metodo inizia con lettera minuscole.

Inizia con la lettera minuscola e continua con le iniziali delle parole principali del nome in maiuscolo.

Ad esempio:

```
getColumnsList()
setTableName($sName)
isConnectionOpened($oDb)
```

In questo modo la distinzione è netta rispetto alle classi che iniziano con lettera maiuscola, ad esempio:

```
Sylar_ConfigBox->isSylarInDebugMode()
```

Questo sopra è un esempio di chiamata ad un metodo statico della classe Sylar_ConfigBox.

Documentazione di esempio:

```
/**
 * Db type name
 * returns the name of db used, like mysql, oracle, ecc... The code must
 * be one of directory in the db package, for example:
 * db.mysql.MysqlDriver =>    name is "mysql"
 * db.oracle.OracleDriver     =>    name is "oracle"
 *
 * The name is also used tu include the class file
 *
 * @since 16/feb/08
 * @author Gianluca Giusti [brdp] <g.giusti@giano-solutions.com>
 *
 * @see SYLAR_USED_DB
 *
 * @todo to be done
 *
 * @return string The code of DbTyle like mysql, oracle
 * @param string $text text to parse
 * @param int iTotNodes number of node to parse
 */
```

File e filename

Nel caso in cui il file contiene una classe il file avrà il nome della classe senza il prefisso *Sylar_* come descritto in precedenza. Importante è ricordare che il php è case sensitive.

Documentazione di esempio:

```
/*
 * This file is part of Sylar.
 *
 * Foobar is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * Foobar is distributed in the hope that it will be useful,
 * but 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.
 *
 * You should have received a copy of the GNU General Public License
 * along with Foobar.  If not, see <http://www.gnu.org/licenses/>.
 *
 * @copyright Copyright Sylar Development Team
 * @license http://www.gnu.org/licenses/ GNU Public License V2.0
 * @see https://launchpad.net/sylar/
 * @see http://www.giano-solutions.com
 */
```

In ogni file dovrà essere ingluso questo preambolo e se il file non contiene un'unica classe e dunque si rende necessario documentare il file come ad esempio nel caso dei files di configurazione bisogna aggiungere anche una descrizione di questo tipo:

```
/**
 * Sylar - Framework Base Settings
 * Contains framework settings, constants and init instructions
 *
 * @package Sylar
 * @version 1.0
 * @since 02-2008
 * @author Gianluca Giusti [brdp] <g.giusti@giano-solutions.com>
 * @copyright Sylar Development Team
 */
```

Sylar Import

Sylar utilizza una funzione del framvork per importare i file da utilizzare pescando dal classpath di sylar e da quello dell'application.

La sintassi di import è semplice ed è la seguente:

```
import('sylar.common.sylar.db.mysql.MysqlDriver');   // import sylar class
import('app.giano.presentation.FormatLogList');      // import application class
import('giano.presentation.FormatLogList');          // import application class
```

Se non si specifica il prefisso *sylar.* o *app.* il framework importerà le classi pescando nel classpath dell'application per default.

Per convenzione gli import delle classi necessarie viene fatto all'inizio del file prima di ogni altra cosa.

Nella pagina successiva viene riportata una classe di esempio per mostrare

```php
<?php
/*
 * This file is part of Sylar.
 *
 * Foobar is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * Foobar is distributed in the hope that it will be useful,
 * but 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.
 *
 * You should have received a copy of the GNU General Public License
 * along with Foobar.  If not, see <http://www.gnu.org/licenses/>.
 *
 * @copyright Copyright Sylar Development Team
 * @license http://www.gnu.org/licenses/ GNU Public License V2.0
 * @see https://launchpad.net/sylar/
 * @see http://www.giano-solutions.com
 */

import('sylar.common.db.DataBaseDriver');
import('sylar.common.system.Logger');
import('sylar.common.system.ExceptionInSylar');


/**
 * Example Class
 *
 * A class designed as example in sylar
 * It's onlly an example
 *
 * @package Sylar
 * @version 1.0
 * @since 16/feb/08
 * @author Gianluca Giusti [brdp] <g.giusti@giano-solutions.com>
 * @copyright Sylar Development Team
 */
class Sylar_ExampleClass{
      private $sTitle = “name”;

      function __construct(){
            # nothing to do
      }
      function __destruct() {
            # nothing to do
      }

      public function setTableTitle($sTitle){
            $this->title = $sTitle;
      }
      public function getTableTitle(){
            return $this->title;
      }
}
?>
```

Gestione del progetto

Il progetto Sylar è attualmente gestito sulla piattaforma Launchpad.net all'indirizzo della home di progetto. Tutta la documentazione, la gestione dei bugs, le FAQ, ecc... sono gestite e disponibili in launchpad.

Version control

Sylar usa bazaar come software per la gestione delle versioni che si integra perfettamente con Launchpad.net.