ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

Chapter 12. MVC & PHP Frameworks

```
<?
$link = mysql_connect('localhost', 'myuser', 'mypass');
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
if($submit) {
   $sql  = “INSERT INTO my_table (name,address,city,state,zip)
   VALUES (”;
   $sql .= “’$name’,’$address’,’$city’,’$state’,’$zip’)”;
   mysql_query($sql);
} else {
   $result = mysql_query(“SELECT * FROM my_table WHERE id = 1”);
   $userArray = mysql_fetch_array($result);
} ?>
<html>
<head><title>Add User</title></head>
<body>
<div>My HTML code blah blah</div>
<form method=“POST”>
   Name: <input type=“text” name=“name”
   value=“<?=$userArray[‘name’]?>”><br>
   …
</form>
…
```

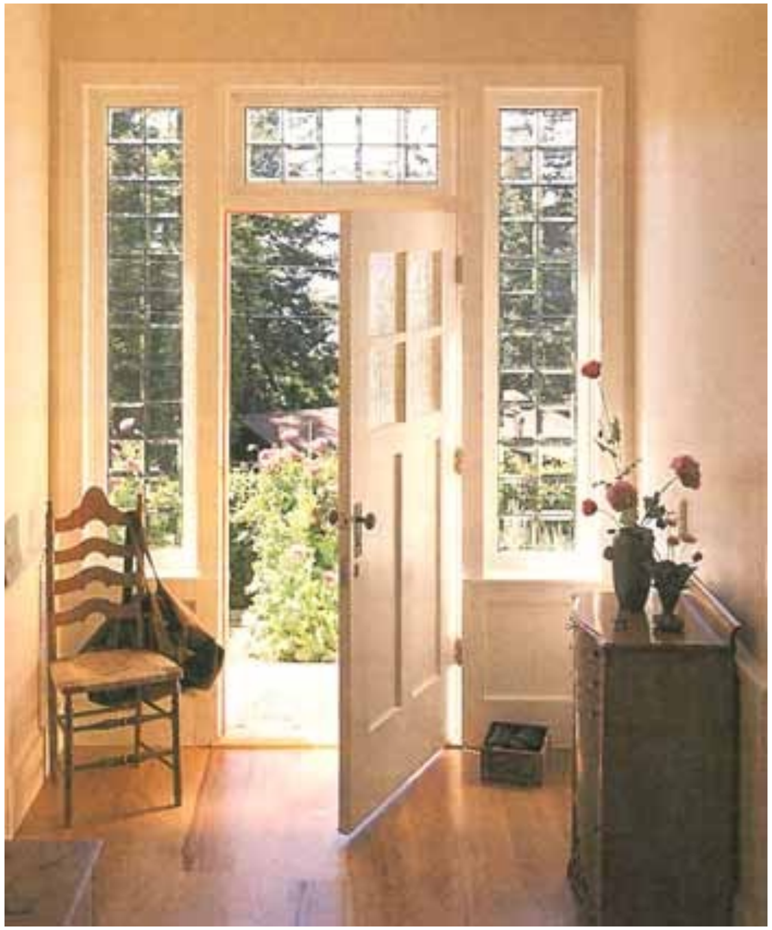
```
<?
require_once("config.inc.php");
require_once("database.inc.php");

$dbh = dbConnect();
if($submit) {
   $sql  = "INSERT INTO my_table
   (name,address,city,state,zip) VALUES (";
   $sql .= "'$name','$address','$city','$state','$zip')";
   $dbh->query($sql);
} else {
   $result = $dbh->query("SELECT * FROM my_table");
   $userArray = $dbh->fetchRow($result);
}
printHeader();?>
<div>My HTML code blah blah</div>
<form method="POST">
   Name: <input type="text" name="name"
   value="<?=$userArray['name']?>"><br>
   …
</form>
…
<? printFooter(); ?>
```

Content

Patterns in Architecture

- Does this room makes you feel happy?
- Why?
    - Light (direction)
    - Proportions
    - Symmetry
    - Furniture
    - And more…



What is a Design Pattern?

A description of a recurrent problem and of the core of possible solutions.

In Short, a solution for a typical problem

Why do we need Patterns?

- Reusing design knowledge
  - Problems are not always unique. Reusing existing experience might be useful.
  - Patterns give us hints to “where to look for problems”.
- Establish common terminology
  - Easier to say, "We need a Façade here“.
- Provide a higher level prospective
  - Frees us from dealing with the details too early
- In short, it’s a “reference”

History of Design Patterns

Christopher Alexander *The Timeless Way of Building* *A Pattern Language: Towns, Buildings, Construction*	Architecture	1970’
Gang of Four (GoF) *Design Patterns: Elements of Reusable Object-Oriented Software* 	Object Oriented Software Design	1995’
Many Authors	Other Areas: HCI, Organizational Behavior, Education, Concurent Programming…	2007’

GoF: Gamma et al (E. Gamma, R. Helm, R. Johnson, J. Vlissides)

Structure of a design pattern*

- Pattern Name and Classification
- Intent
  - a Short statement about what the pattern does
- Motivation
  - A scenario that illustrates where the pattern would be useful
- Applicability
  - Situations where the pattern can be used

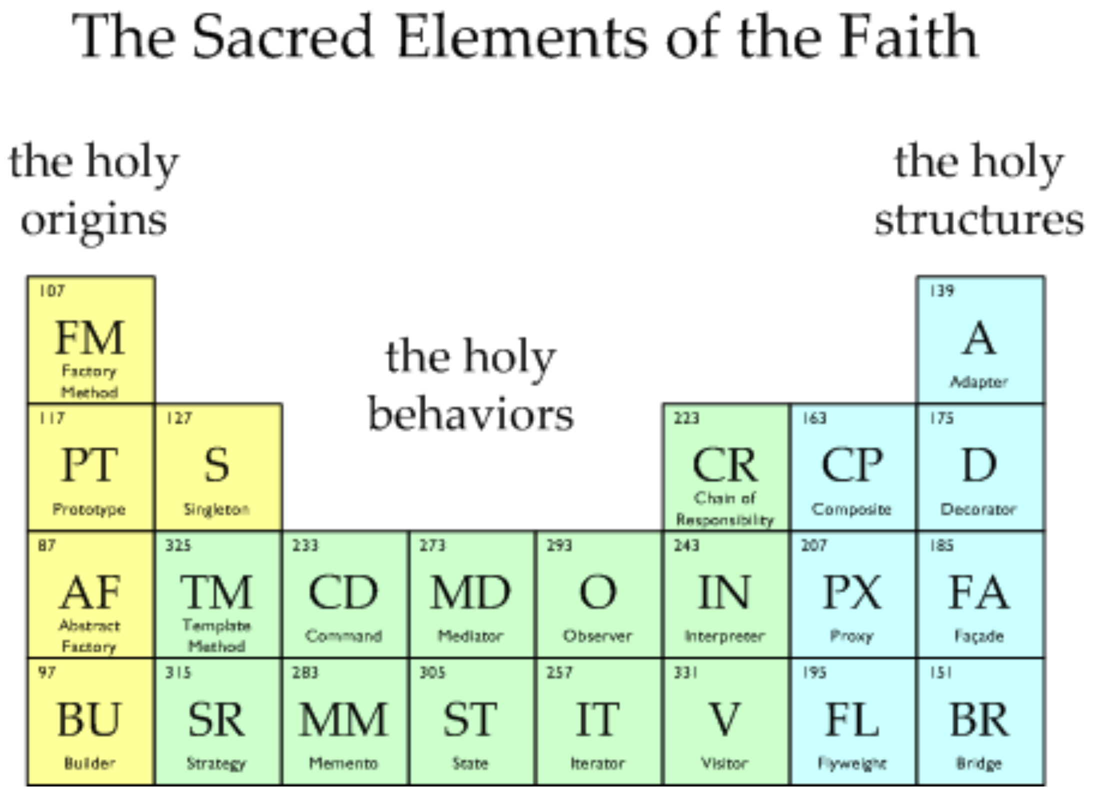
*According to GoF

Structure (2)

- Structure
  - A graphical representation of the pattern
- Participants
  - The classes and objects participating in the pattern
- Collaborations
  - How to do the participants interact to carry out their responsibilities?
- Consequences
  - What are the pros and cons of using the pattern?
- Implementation
  - Hints and techniques for implementing the pattern

Classification of GoF Patterns

- Types
  - Creational
  - Structural
  - Behavioral



Taken from Vince Huston's site about Design Patterns
http://home.earthlink.net/~huston2/dp/patterns.html

Observer **Behavioral**

Intent

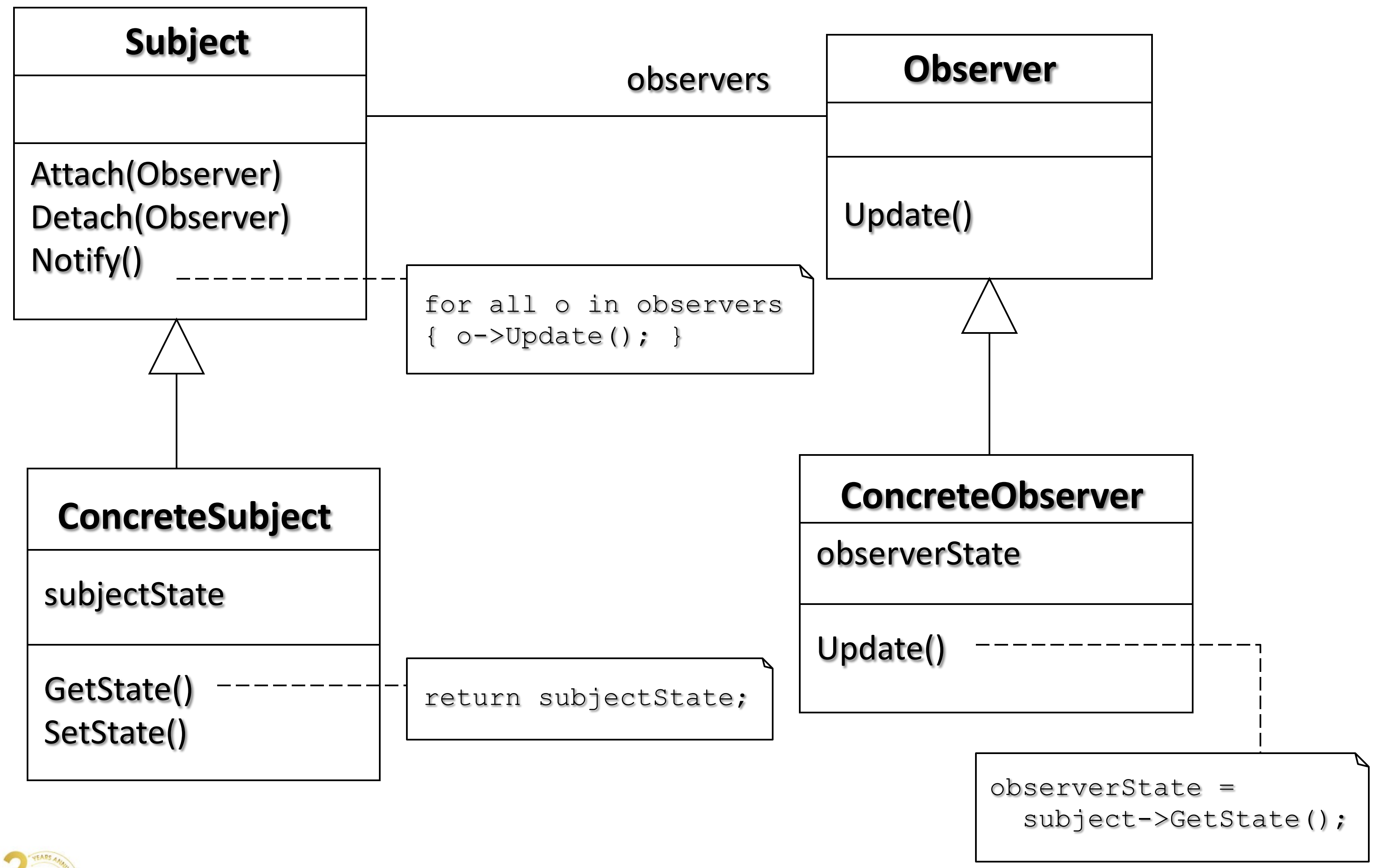
The Observer pattern defines an one-to-many dependency between a subject object and any number of observer objects so that when the subject object changes state, all its observer objects are notified and updated automatically.

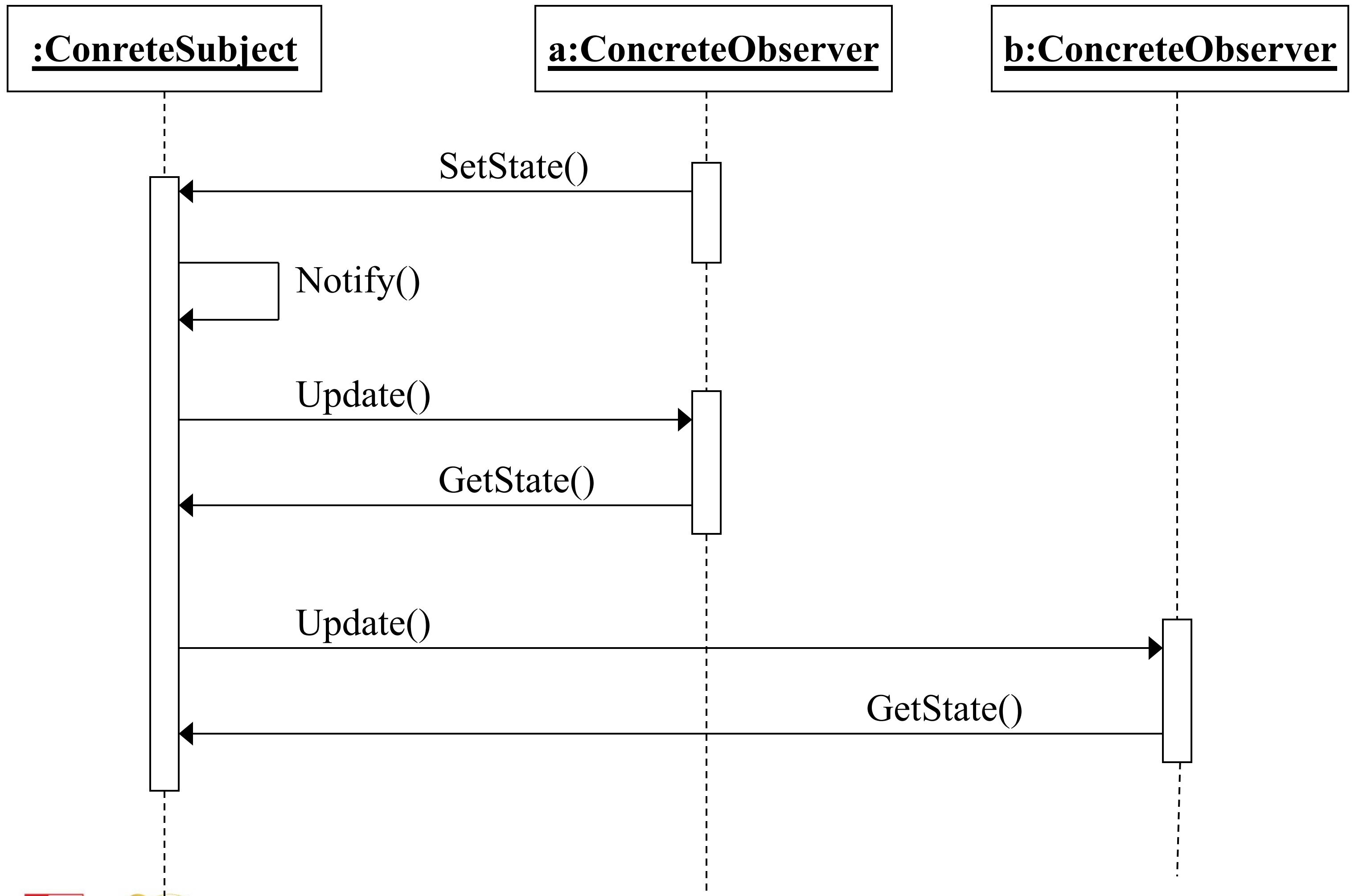
Motivation

The Observer design pattern has two parts and they are subject and observer. The relationship between subject and observer is one-to-many. In order to reuse subject and observer independently, their relationship has to be decoupled. An example of using the observer pattern is the graphical interface toolkit which separates the presentational aspect with application data. The presentation aspect is the observer part and the application data aspect is the subject part.

For example, in a spreadsheet program, the Observer pattern can be applied to separate the spreadsheet data from its different views. In one view spreadsheet data can be presented as a bar graph and in another view it can be represented as a pie chart.
The spread sheet data object notifies the observers whenever a there is a data change that can make its state inconsistent with its observers.

Class Diagram for the Observer Pattern




:ConreteSubject
a:ConcreteObserver
b:ConcreteObserver
SetState()
Notify()
Update()
GetState()
Update()
GetState()

Applicability

Use the observer pattern in any of the following situations:

- When the abstraction has two aspects with one dependent on the other. Encapsulating these aspects in separate objects will increase the chance to reuse them independently.
- When the subject object doesn't know exactly how many observer objects it has.
- When the subject object should be able to notify it's observer objects without knowing who these objects are.

Participants

- Subject
    - Knows it observers
    - Has any number of observer
    - Provides an interface to attach and detaching observer object at run time
- ConcreteSubject
    - Store subject state interested by observer
    - Send notification to it's observer
- Observer
    - Provides an update interface to receive signal from subject
- ConcreteObserver
    - Maintain reference to a ConcreteSubject object
    - Maintain observer state
    - Implement update operation

Consequences

Further benefit and drawback of Observe pattern include:

- Abstract coupling between subject and observer, each can be extended and reused individually.
- Dynamic relationship between subject and observer, such relationship can be established at run time. This gives a lot more programming flexibility.
- Support for broadcast communication. The notification is broadcast automatically to all interested objects that subscribed to it.
- Unexpected updates. Observes have no knowledge of each other and blind to the cost of changing in subject. With the dynamic relationship between subject and observers, the update dependency can be hard to track down.

Known Uses

- Smalltalk Model/View/Controller (MVC). User interface framework while Model is subject and View is observer.

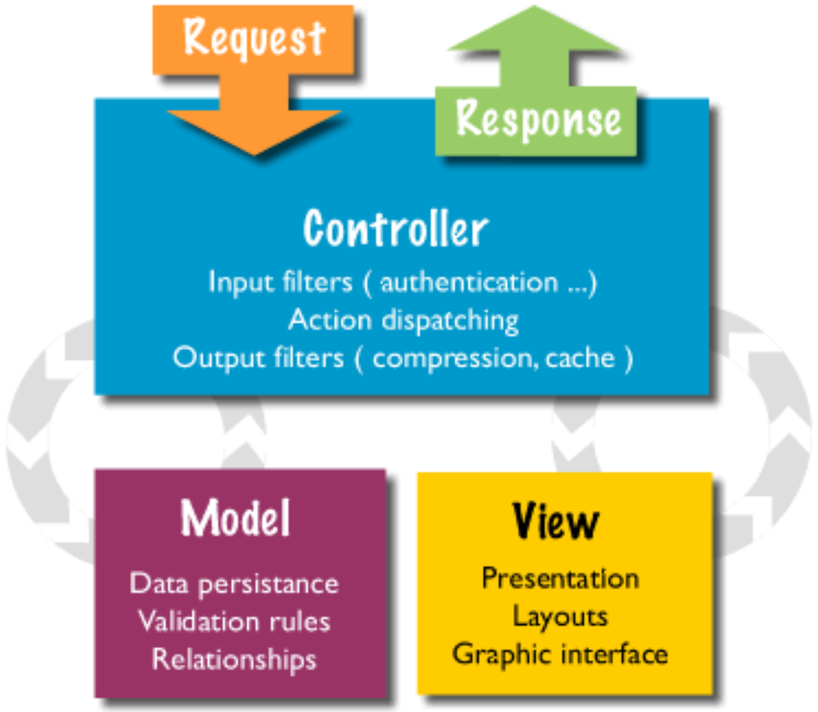
Content

1. What is MVC Architecture?

- MVC is a design structure for separating representation from presentation using a subscribe/notify protocol
- The basic idea is to separate
  - where and how data (or more generally some state) is stored, i.e., the model
  - from how it is presented, i.e., the views
- Follows basic software engineering principles:
  - Separation of concerns
  - Abstraction

1. What is MVC Architecture? (2)

- MVC consists of three kinds of objects
  - Model is the application object
  - View is its screen presentation
  - Controller defines the way the user interface reacts to user input



1. What is MVC Architecture? (3)

- MVC decouples views and models by establishing a subscribe/notify protocol between them
  - whenever model changes it notifies the views that depend on it
  - in response each view gets an opportunity to update itself
- This architecture allows you to attach multiple views to a model
  - it is possible to create new views for a model without rewriting it

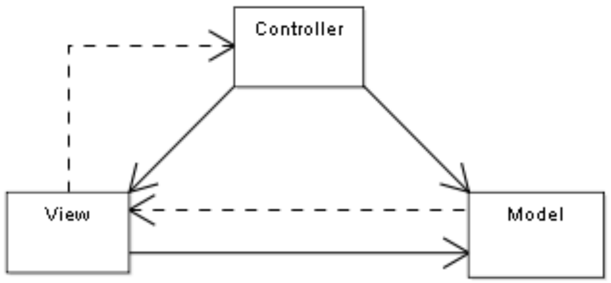
MVC Architecture in Web Applications

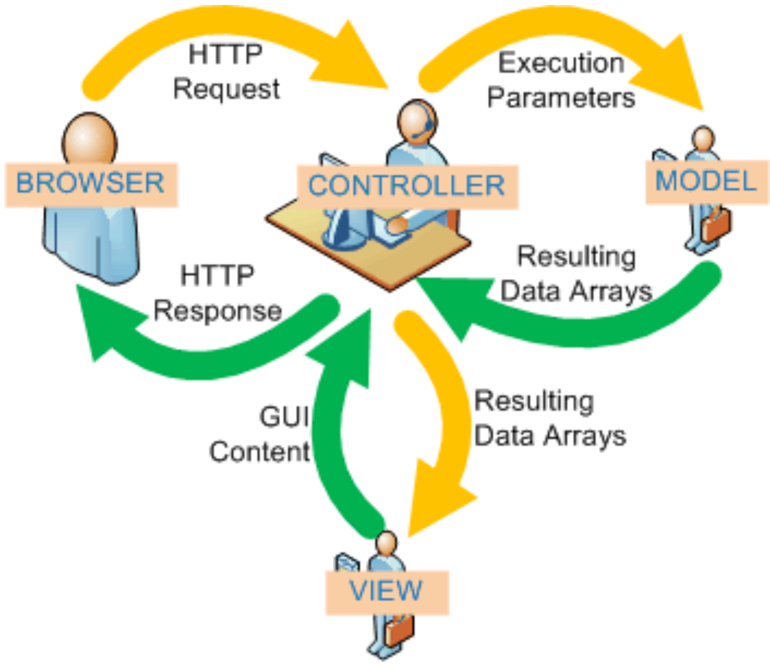
- Many web frameworks support web application development based on the MVC architecture
  - Ruby on Rails, Zend Framework for PHP, CakePHP, Spring Framework for Java, Struts Framework for Java, Django for Python, …

- MVC architecture has become the standard way to structure web applications

MVC Framework for Web Applications

- Model-View-Controller
- Separates:
  - **M: Data model**
  - **V: Presentation (UI)**
  - **C: Business logic**





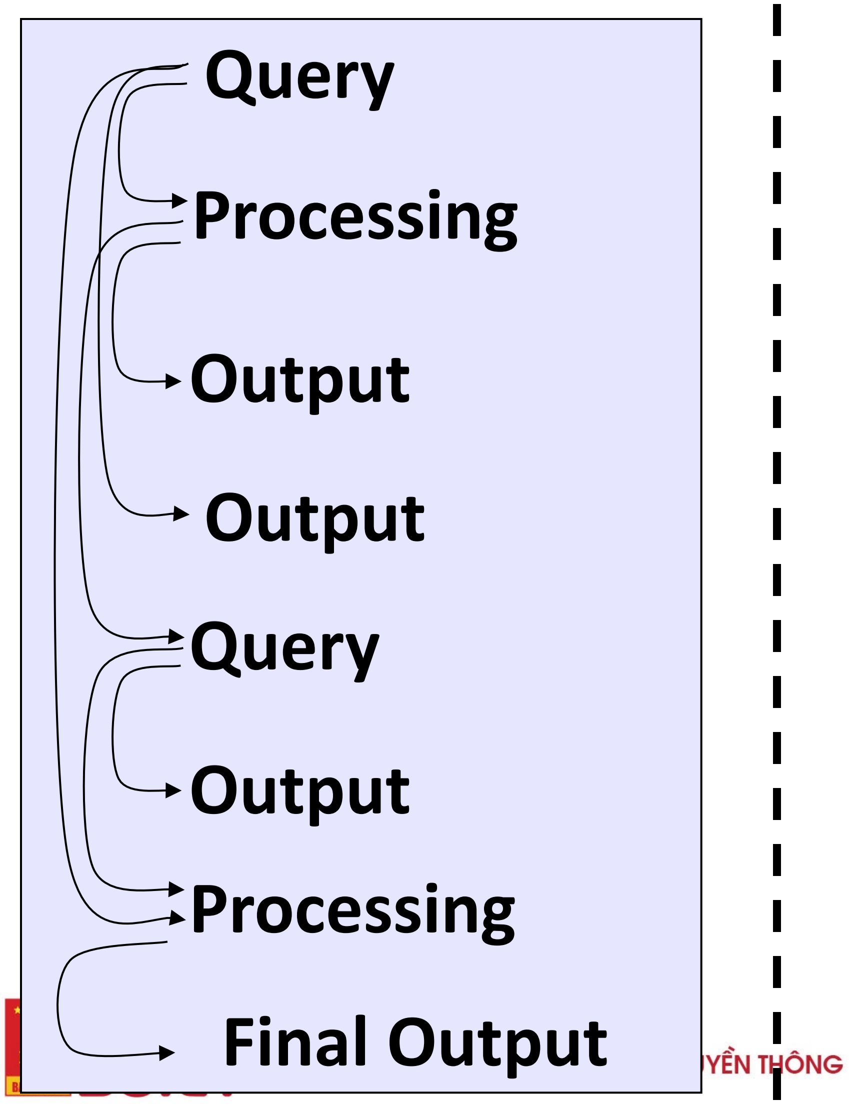
MVC Framework for Web Applications

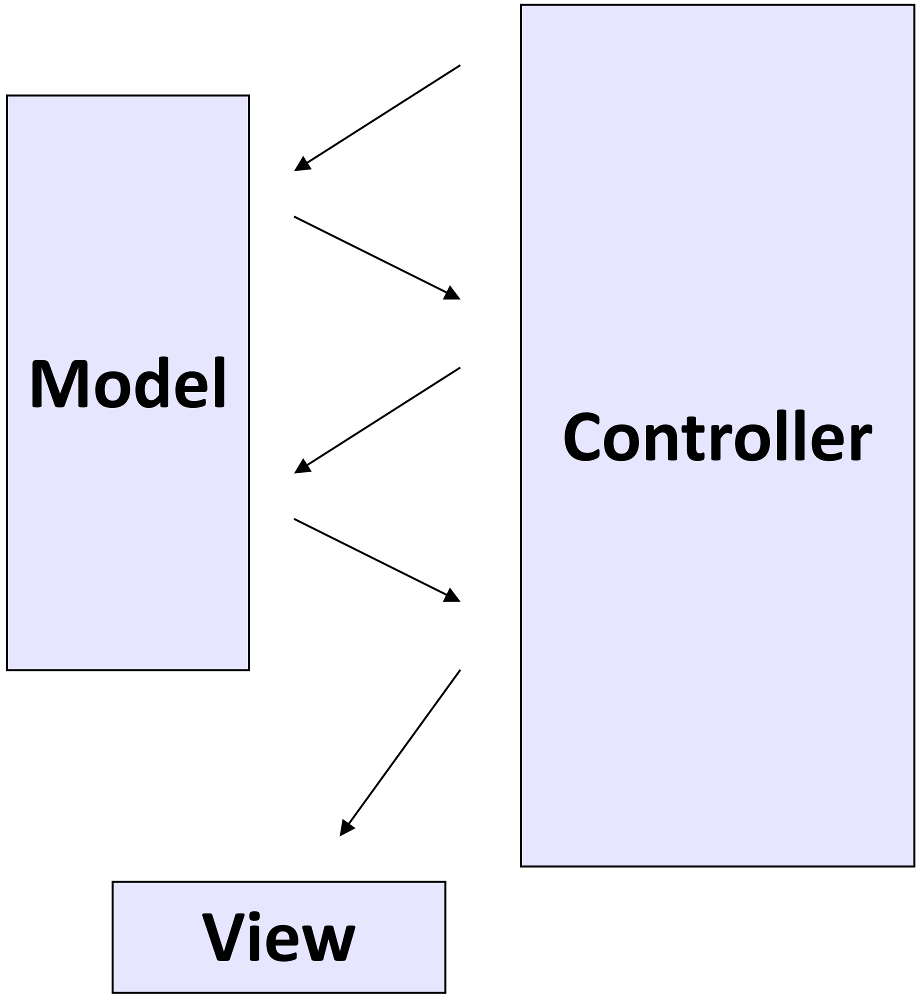
- Model: Data model which is an abstract representation of the data stored in the backend database. Typically uses an object-relational mapping to map the class structure for the data model to the tables in the back-send database
- Views: These are responsible for rendering of the web pages, i.e., how is the data presented in user's browser
- Controllers: Controllers are basically event handlers that process incoming user requests. Based on a user request, they can update the data model, and create a new view to be presented to the user

Why use an MVC framework?

- Avoid “reinventing the wheel”
- Use proven, tested code
- Automation (ORM, generators)
- Maintainability
- “Plugin” functionality

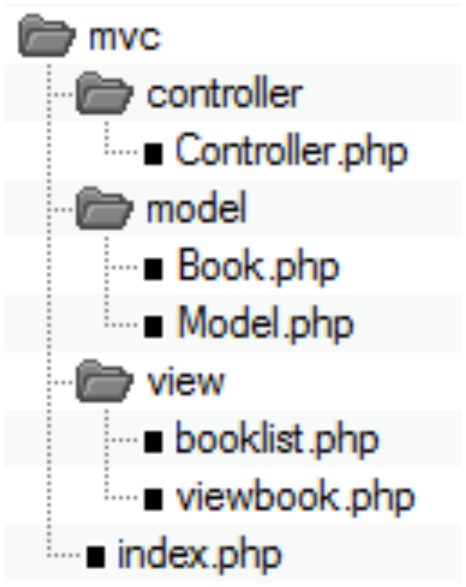
Flow: Traditional vs. MVC



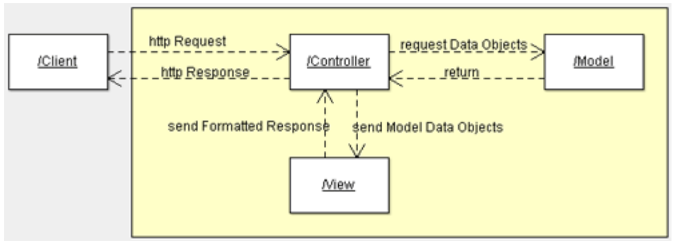


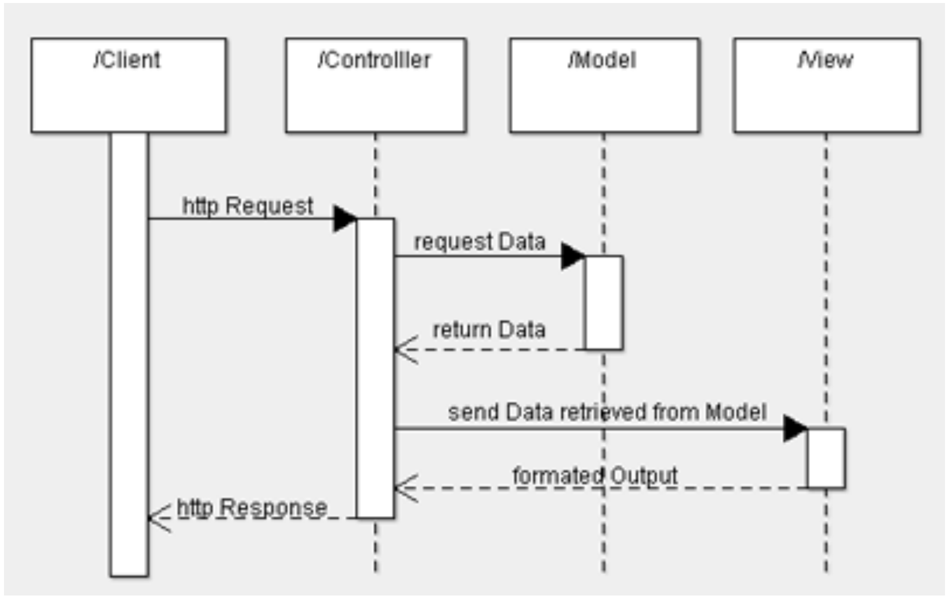
Content

3.1. Your own framework

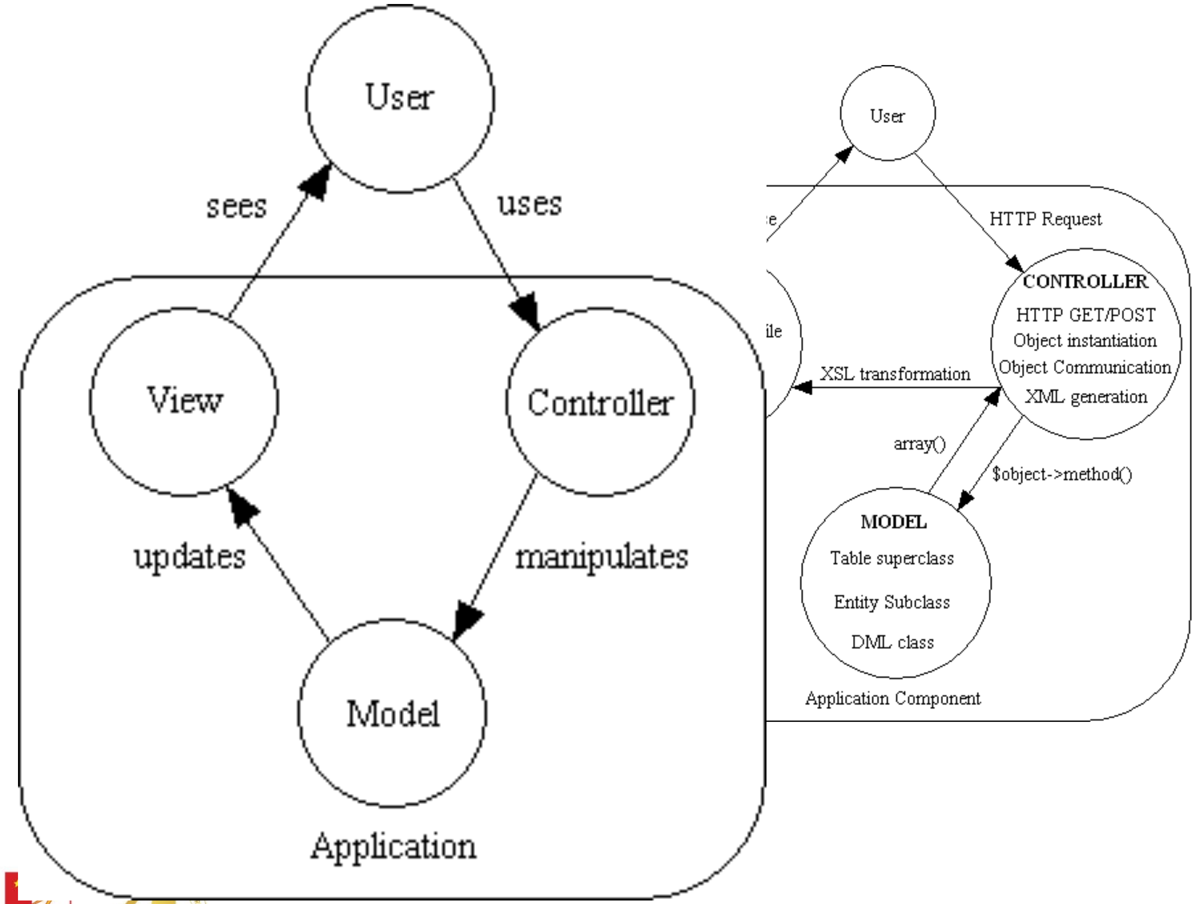


```
mvc
├── controller
│   └── Controller.php
├── model
│   ├── Book.php
│   └── Model.php
├── view
│   ├── booklist.php
│   └── viewbook.php
└── index.php
```

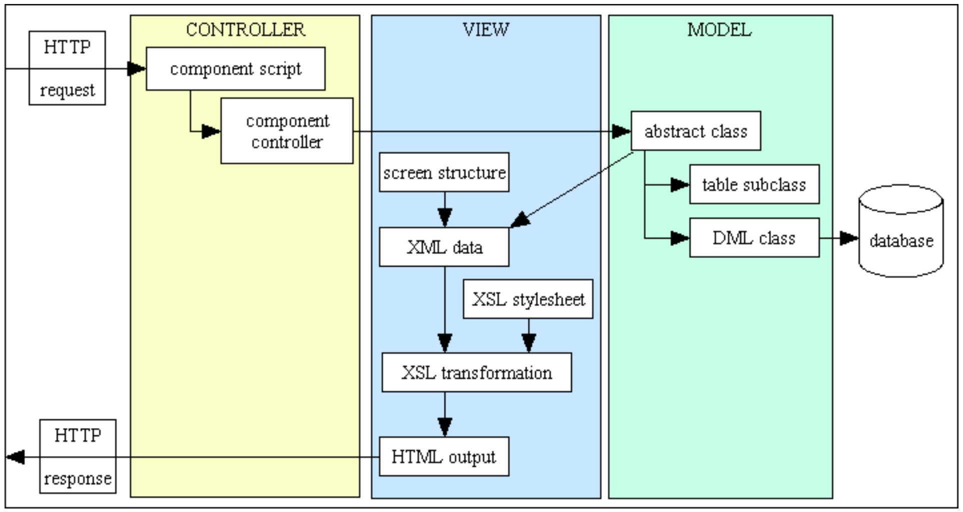




3.1. Your own framework (2)



3.1. Your own framework (2)



3.2. Existed PHP Frameworks

- Zend Framework for PHP: http://zend.com
- Symfony: http://symfony-project.org
- CakePHP: http://cakephp.org
- CodeIgniter: http://codeigniter.com
- Xisc: http://xisc.coom
- …

Popular PHP MVC Frameworks

- **CakePHP**
  - Documentation is somewhat lacking
  - Apparently difficult for beginners
- **Symfony**
  - Great documentation and community
  - Easy to get started
- **Zend**
  - Supported by Zend (official PHP company)
  - More of a library than complete framework

Should you use an existed MVC framework for your project?

- Are there complex hierarchical relationships in your data?
- Will this project need to be maintained by more than one person for more than a year?
- Do you need the ability to add advanced features like AJAX without writing the code from scratch?
- Probably Yes. (unless it's a throwaway)
  - Use a well-established framework with good documentation and a large community

Why did we choose Symfony?

- Steep learning curve, but…
- Great documentation
- Great community
- Well-written and tested code
- Nice deployment system (PEAR/SVN)
- Extensive use of existing projects, instead of rewriting everything from scratch

symfony

About Installation Documentation

Open-Source PHP Web Framework

Symfony is a full-stack framework, a library of cohesive classes written in PHP5.

It provides an architecture, components and tools for developers to build complex web applications faster. Choosing symfony allows you to release your applications earlier, host and scale them without problem, and maintain them over time with no surprise.

applications!

New users join the community every day, and that makes of symfony the most popular PHP5 framework around. A large community means easy-to-find support, user-contributed documentation, plugins, and free applications.

Download

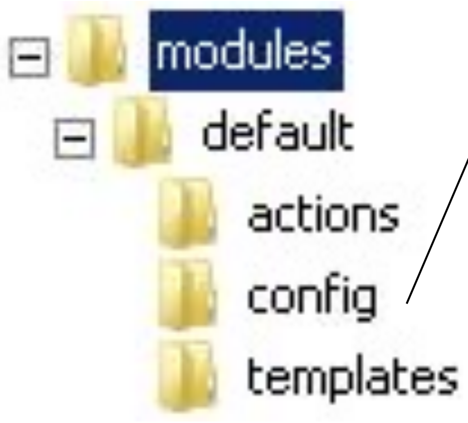
Download and install the latest stable release.

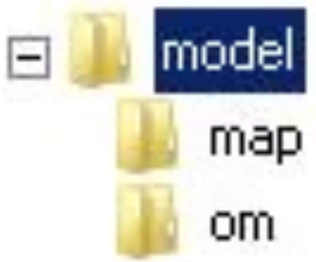
Discover

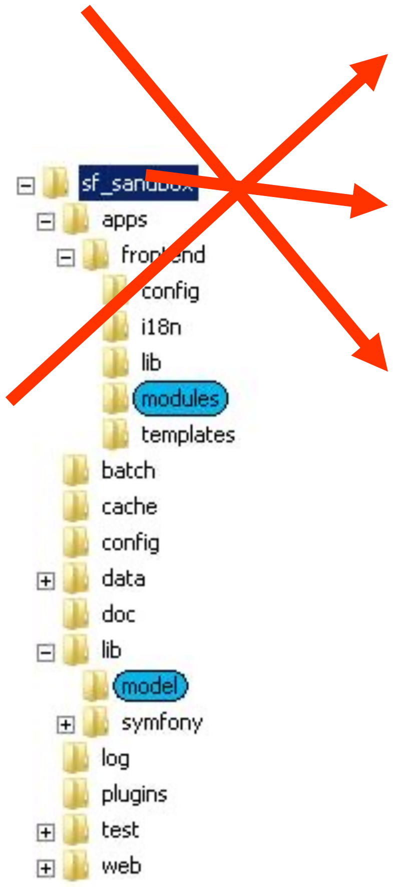
Build your first project or start browsing the book.

mailing-lists, join us in the #symfony IRC channel or ask a question on our forum

http://www.symfony-project.org





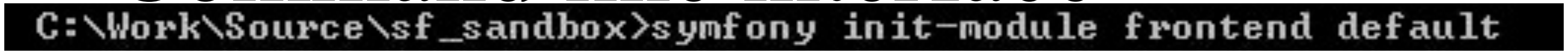


Model

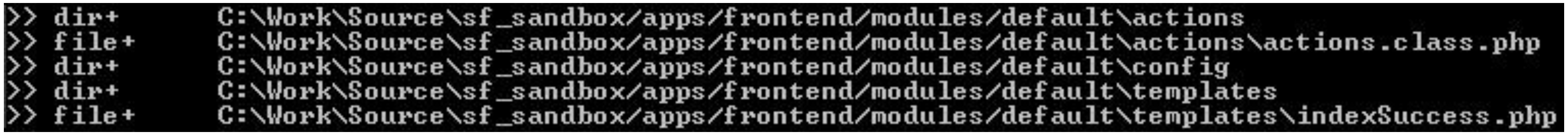
View

Controller

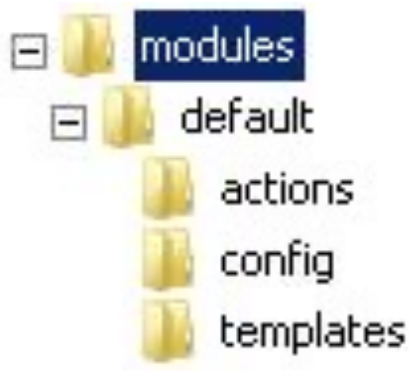
Command line interface



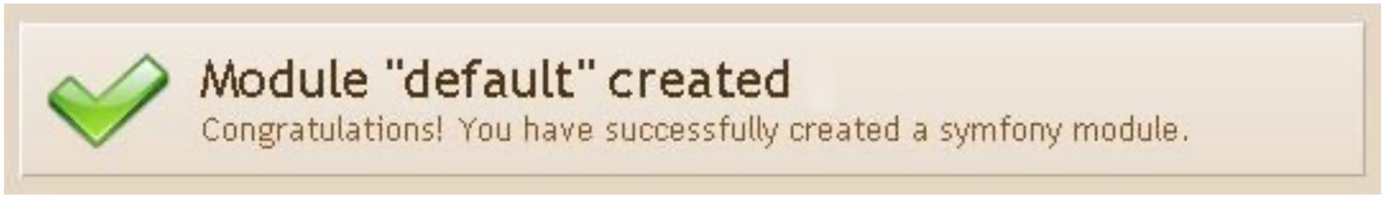
```
C:\Work\Source\sf_sandbox>symfony init-module frontend default
```



```
>> dir+      C:\Work\Source\sf_sandbox/apps/frontend/modules/default\actions
>> file+     C:\Work\Source\sf_sandbox/apps/frontend/modules/default\actions\actions.class.php
>> dir+      C:\Work\Source\sf_sandbox/apps/frontend/modules/default\config
>> dir+      C:\Work\Source\sf_sandbox/apps/frontend/modules/default\templates
>> file+     C:\Work\Source\sf_sandbox/apps/frontend/modules/default\templates\indexSuccess.php
```



- Has a command line interface
- Creates the basis for the module
- Helps with maintenance
- Easy to check if everything worked

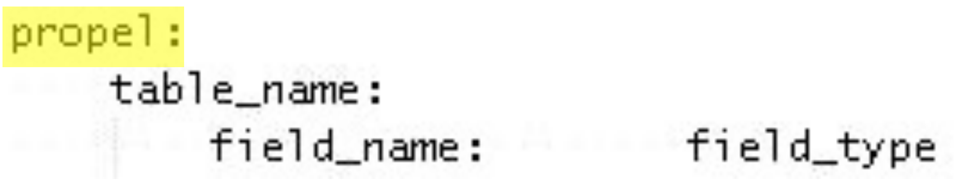


Module "default" created

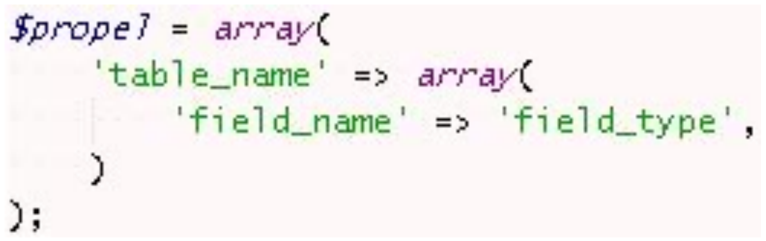
Congratulations! You have successfully created a symfony module.

Object Relational Mapper (ORM)

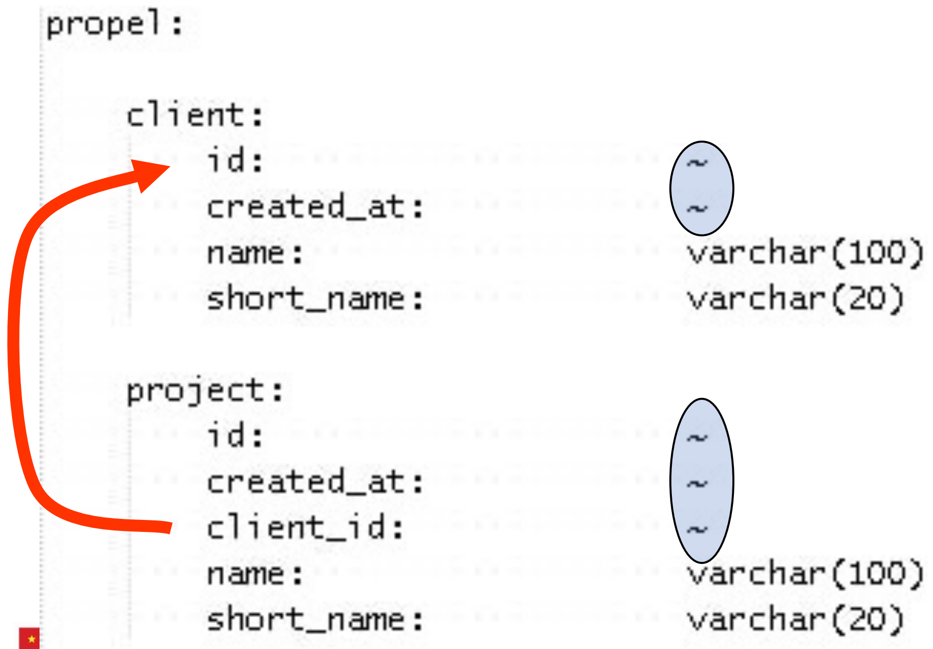
YAML (Used in RoR)



```
propel:
  table_name:
    field_name:         field_type
```



```
$propel = array(
    'table_name' => array(
        'field_name' => 'field_type',
    )
);
```



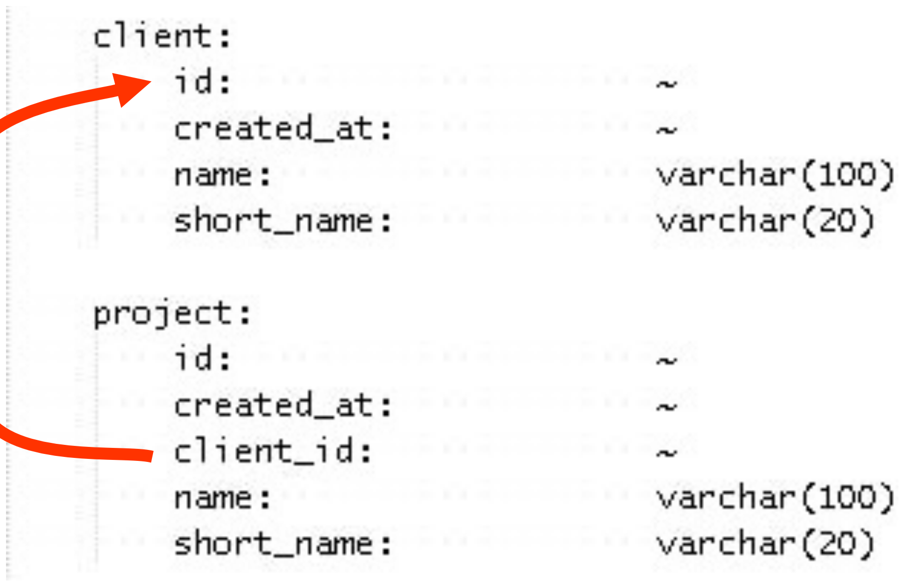
```
propel:

  client:
    id:                  ~
    created_at:          ~
    name:                varchar(100)
    short_name:          varchar(20)

  project:
    id:                  ~
    created_at:          ~
    client_id:           ~
    name:                varchar(100)
    short_name:          varchar(20)
```

- Propel gives database independence
- Id, created_at, & foreign key fields are autogenerated

Object Relational Mapper (ORM)



```
client:
    id:                       ~
    created_at:               ~
    name:                     varchar(100)
    short_name:               varchar(20)

project:
    id:                       ~
    created_at:               ~
    client_id:                ~
    name:                     varchar(100)
    short_name:               varchar(20)
```

- Database tables = classes & table rows = objects
- Auto-generated based on schema

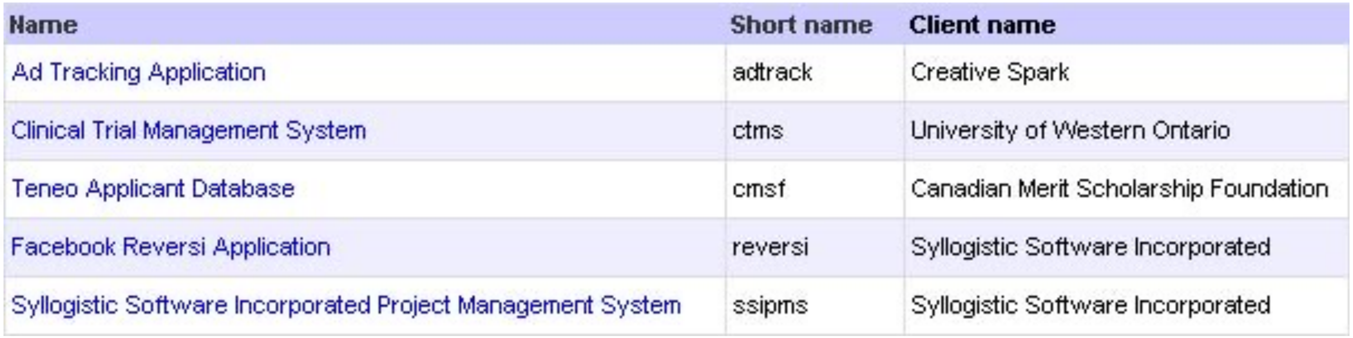
```
// Retrieve the first project by it Primary Key
$project = ProjectPeer::retrieveByPK(1);
```

[redacted]

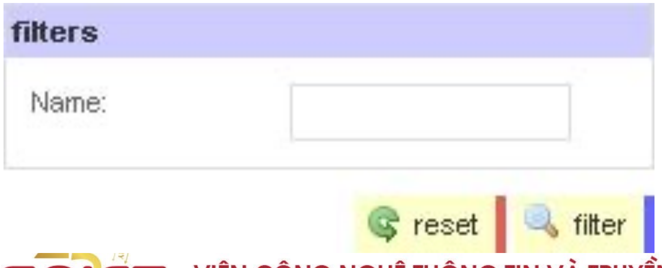
Admin Generator

```
generator:
  class:
  param:
    model_class:
    theme:

    list:
      title:   Project List
      display: [ =name, short_name, client_name, created_at ]
      filters: [ name ]
      click_action: show
      object_actions:
        _show:
        _edit:
        _delete:
```



Name	Short name	Client name
Ad Tracking Application	adtrack	Creative Spark
Clinical Trial Management System	ctms	University of Western Ontario
Teneo Applicant Database	cmsf	Canadian Merit Scholarship Foundation
Facebook Reversi Application	reversi	Syllogistic Software Incorporated
Syllogistic Software Incorporated Project Management System	ssipms	Syllogistic Software Incorporated



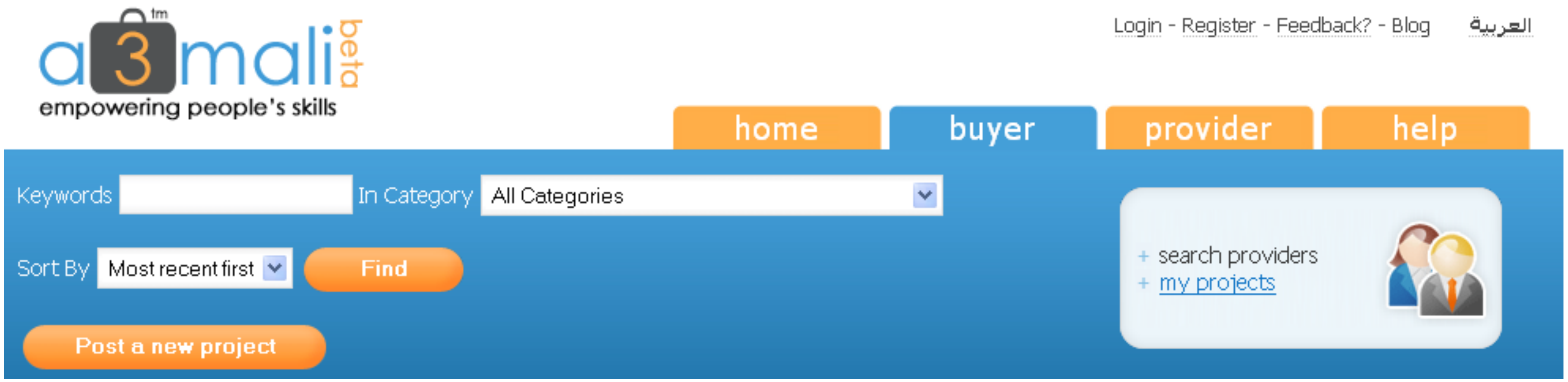
filters

Name:

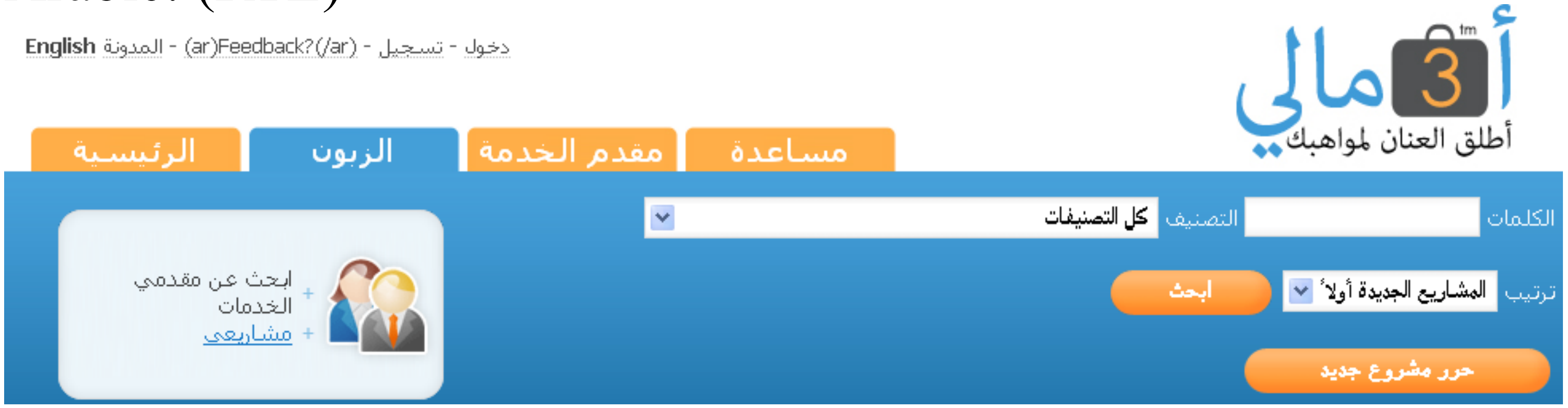
- List, show, edit, & add
- 4 pages, < 30 LOC

i18n and l10n

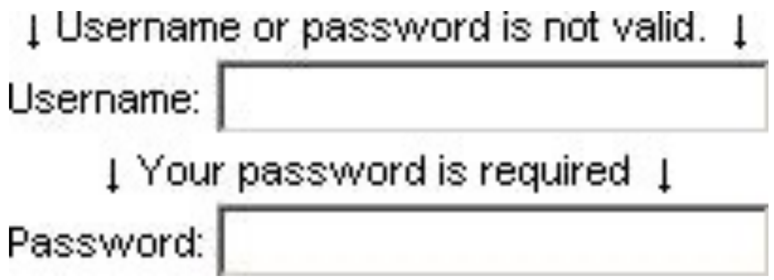
English: (LTR)



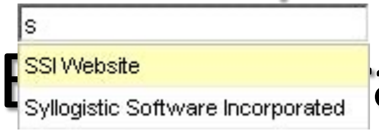
Arabic: (RTL)



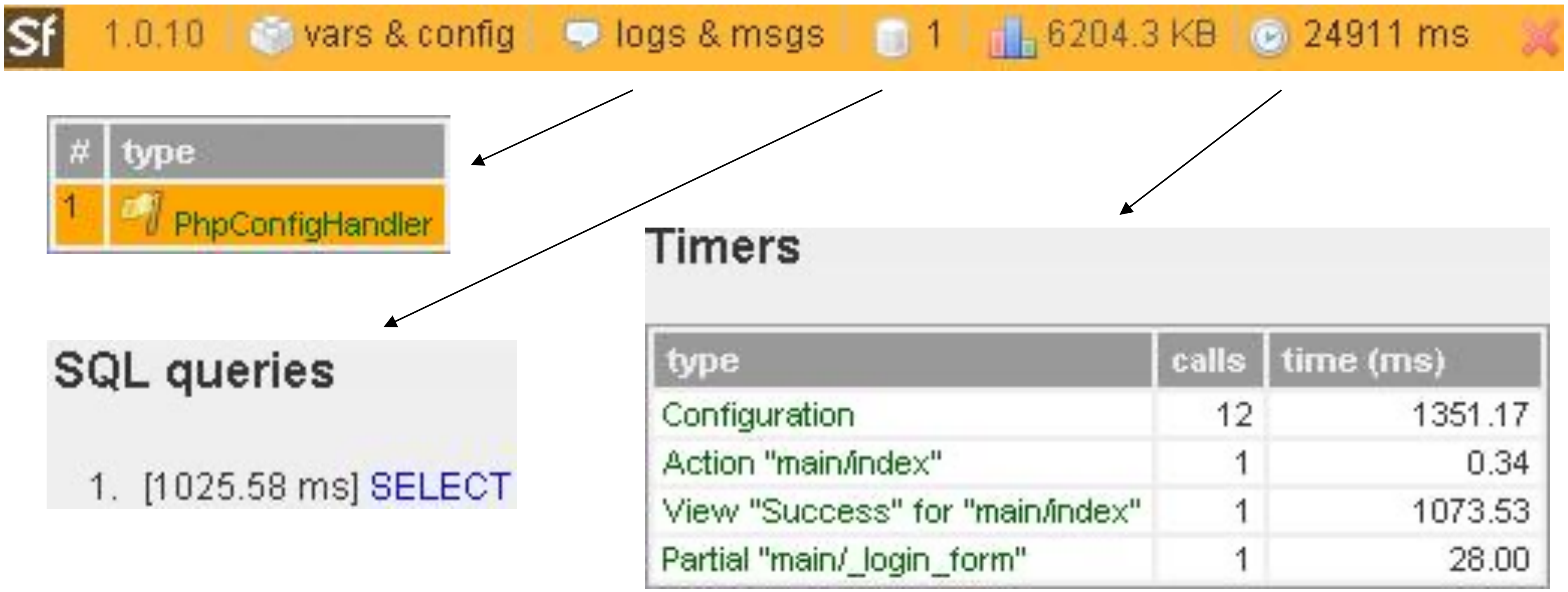
Symfony Plugins



- 100s of plugins
- E ation



- AJAX, CSS, CMS, SEO, Security, Flash, etc.



Sf 1.0.10 vars & config logs & msgs 1 6204.3 KB 24911 ms

#	type
1	PhpConfigHandler

SQL queries

1. [1025.58 ms] SELECT

Timers

type	calls	time (ms)
Configuration	12	1351.17
Action "main/index"	1	0.34
View "Success" for "main/index"	1	1073.53
Partial "main/_login_form"	1	28.00

- Diagnose Problems
- Check Execution Time
- Optimize SQL Queries

Question?