Web Development

MVC & PHP Frameworks

- Everything shoved into one file. ☹ Not Good!

Typical PHP Code

```
<?
$link = mysql_connect('localhost', 'myuser', 'mypass');
if (!$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>
…
```

Better but still not great

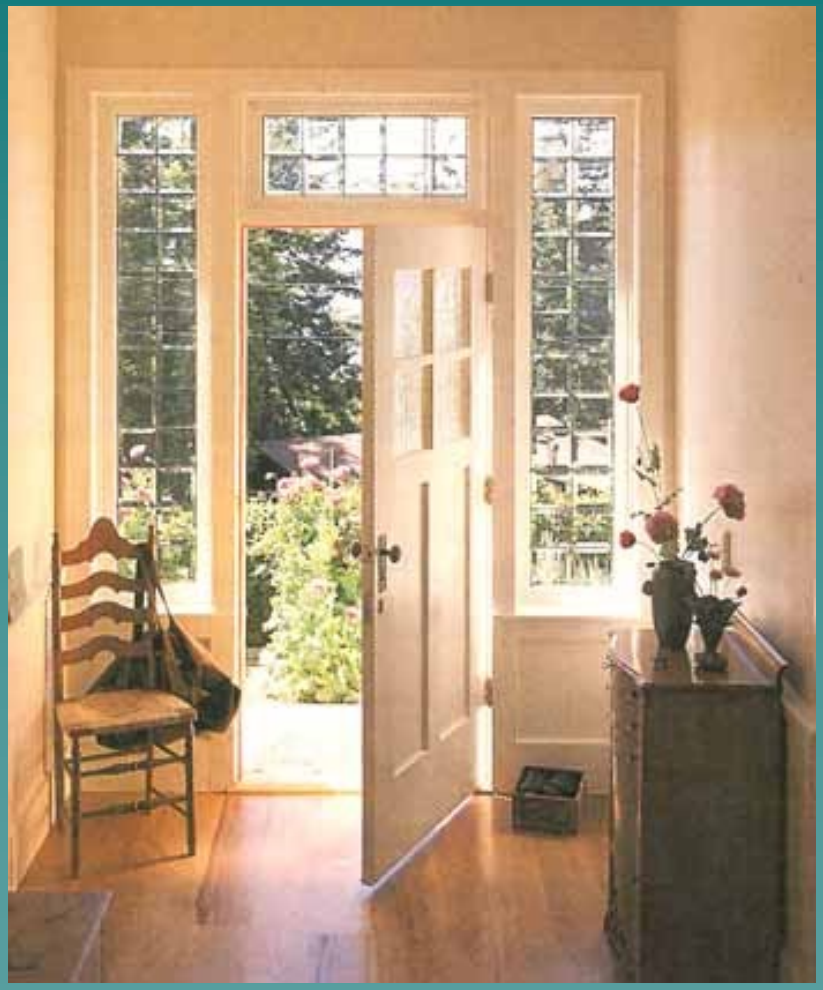
```
<?
require_once("config.inc.php");
require_once("database.inc.php");

$dbh = dbConnect();
if($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

1. Overview of Design Patterns
2. What is MVC architecture?
3. PHP Frameworks

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”.

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)

Content

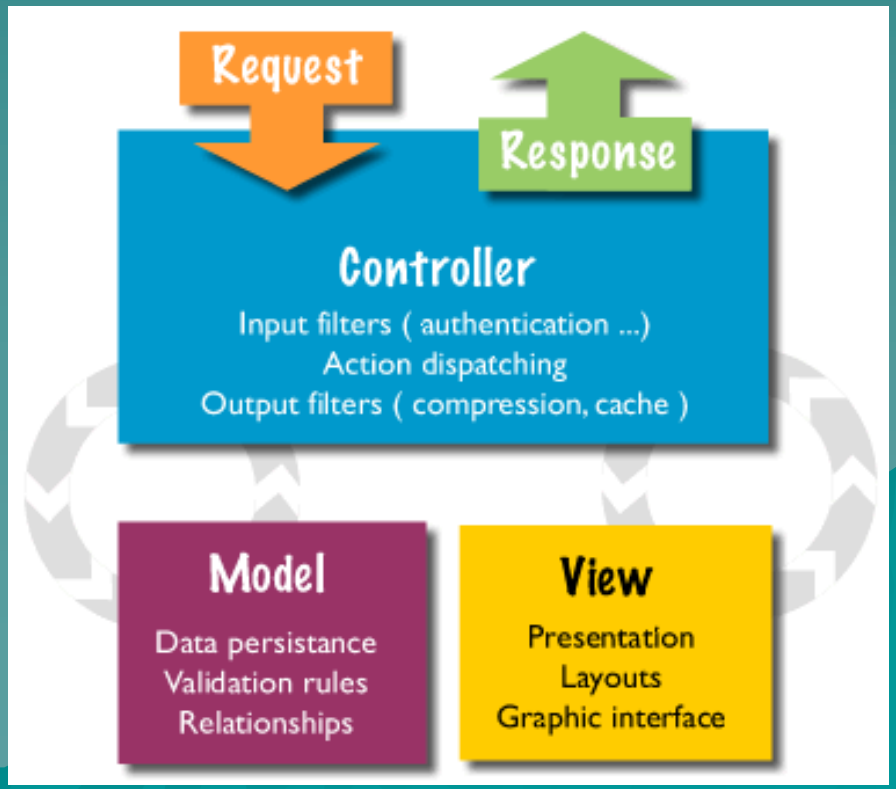
1. Overview of Design Pattern
2. What is MVC architecture?
3. PHP Frameworks

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)

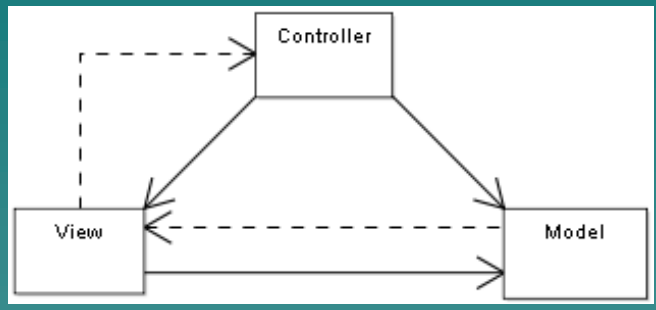
- MVC decouples views and models by establishing a subscribe/notify protocol between them
  - whenever model changes it notifies the views that depend on it
  - in response each view gets an opportunity to update itself
- This architecture allows you to attach multiple views to a model
  - it is possible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

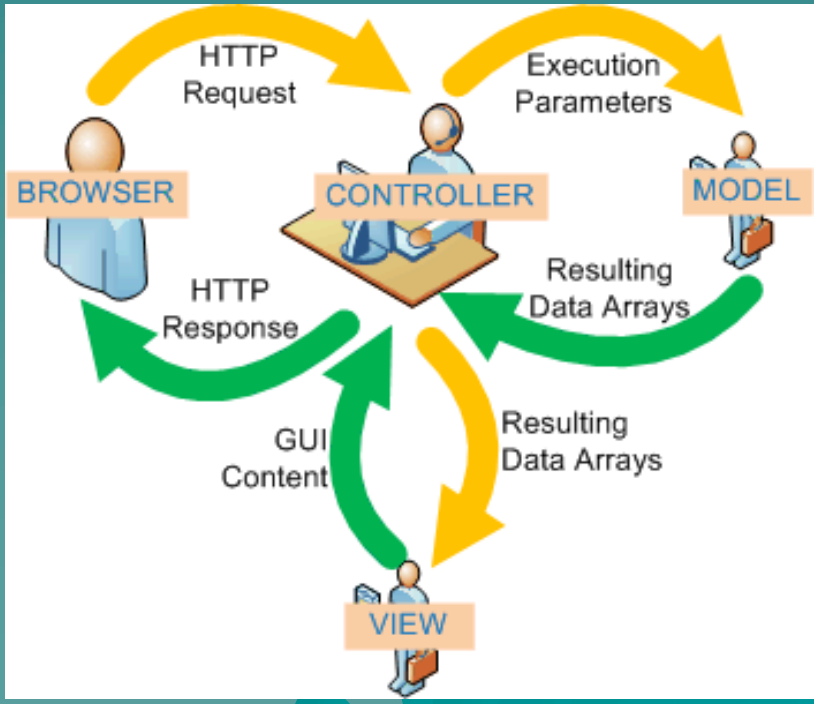
MVC Architecture in Web Applications

- 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**





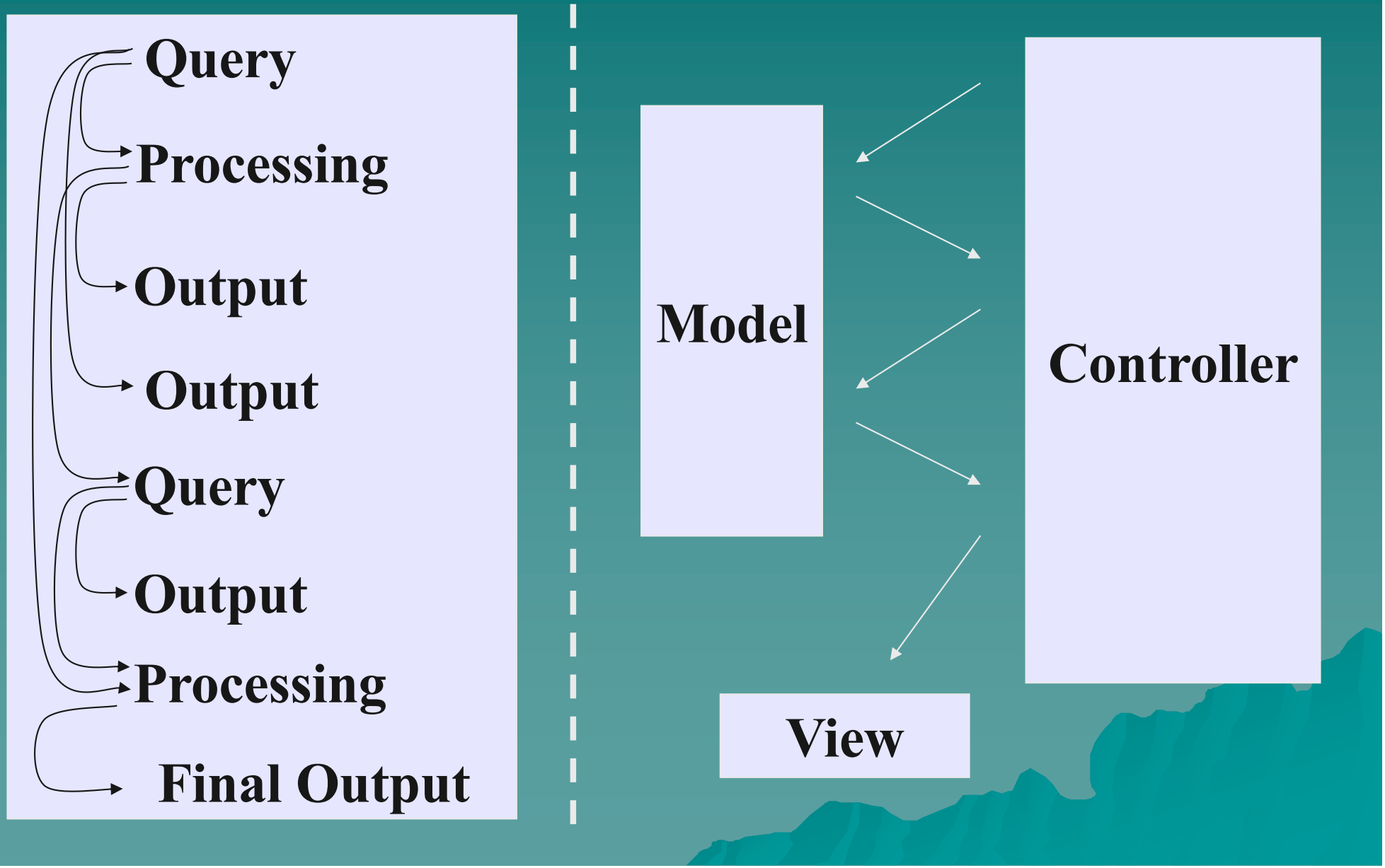
MVC Framework for Web Applications

- 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

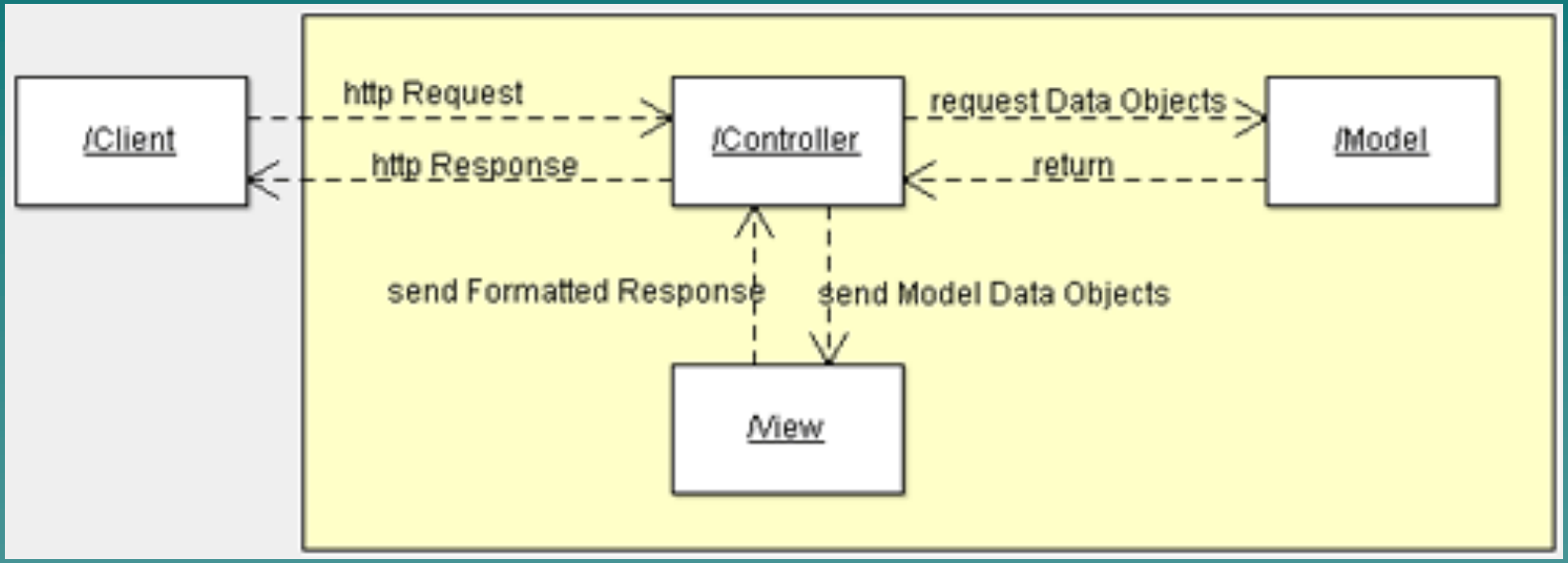
Flow: Traditional vs. MVC

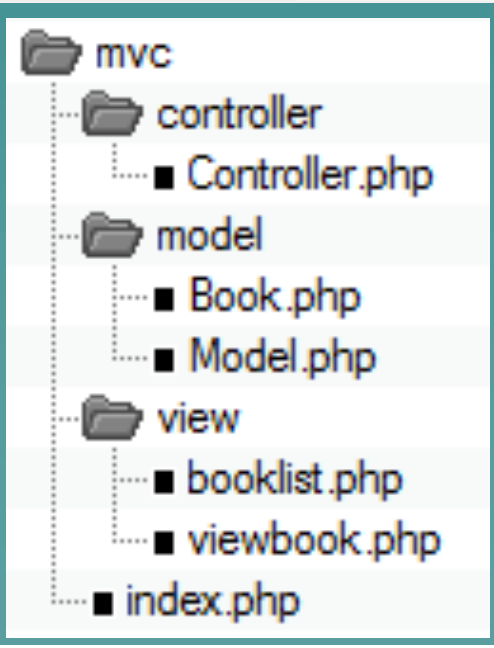


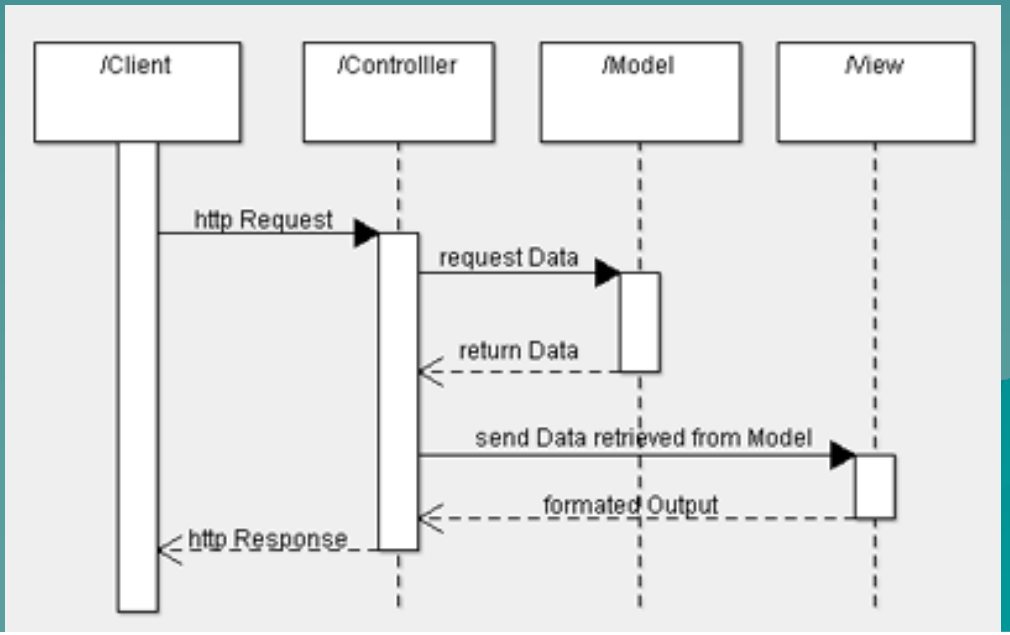
Content

1. Overview of Design Patterns
2. What is MVC architecture?
3. PHP Frameworks

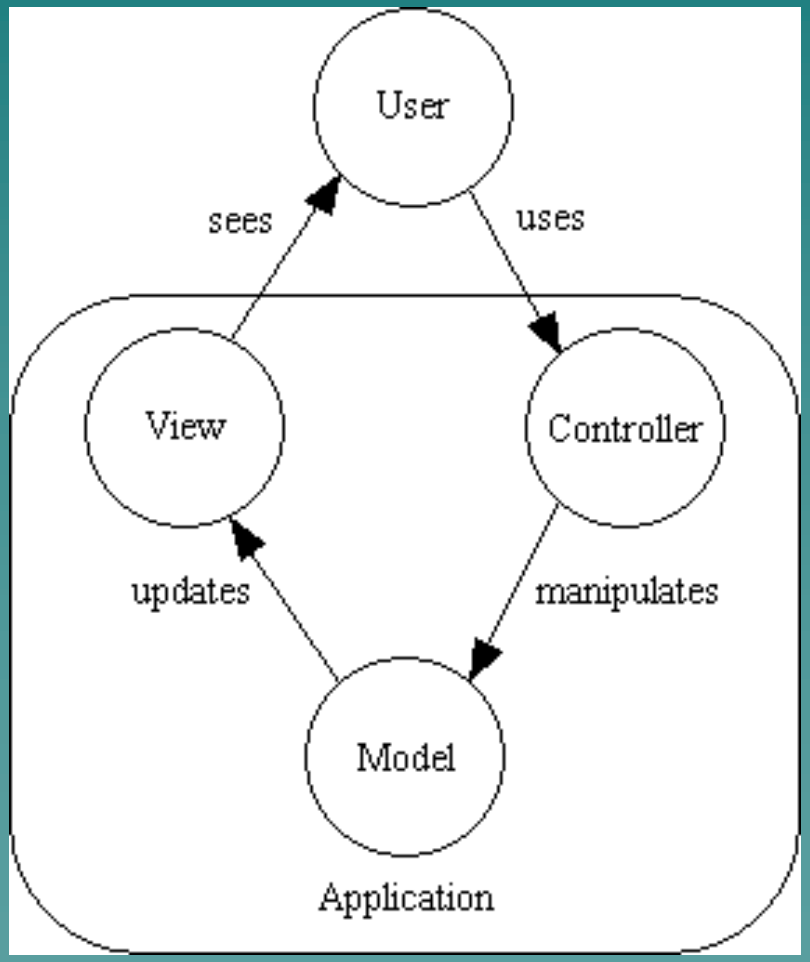
3.1. Your own framework

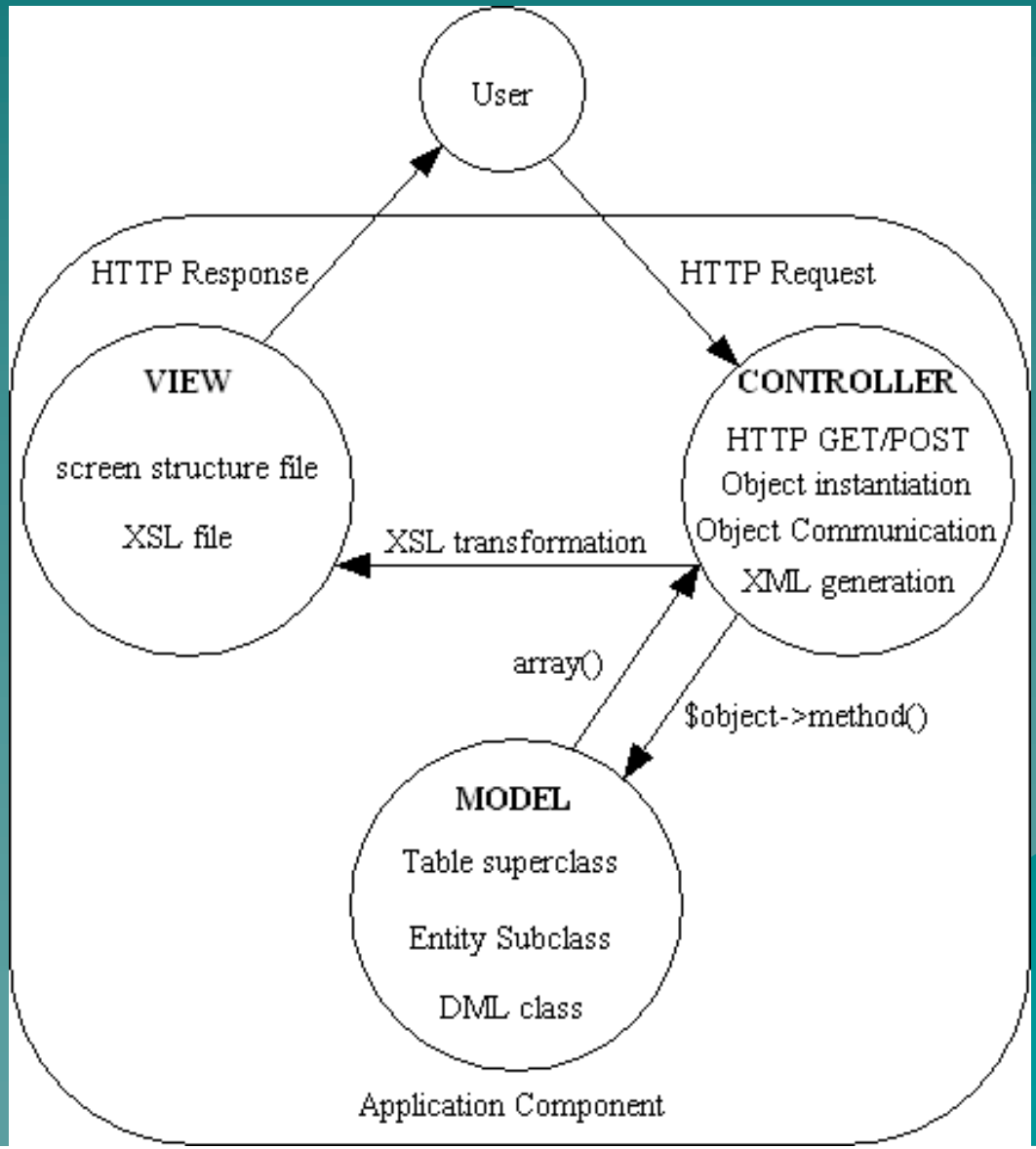




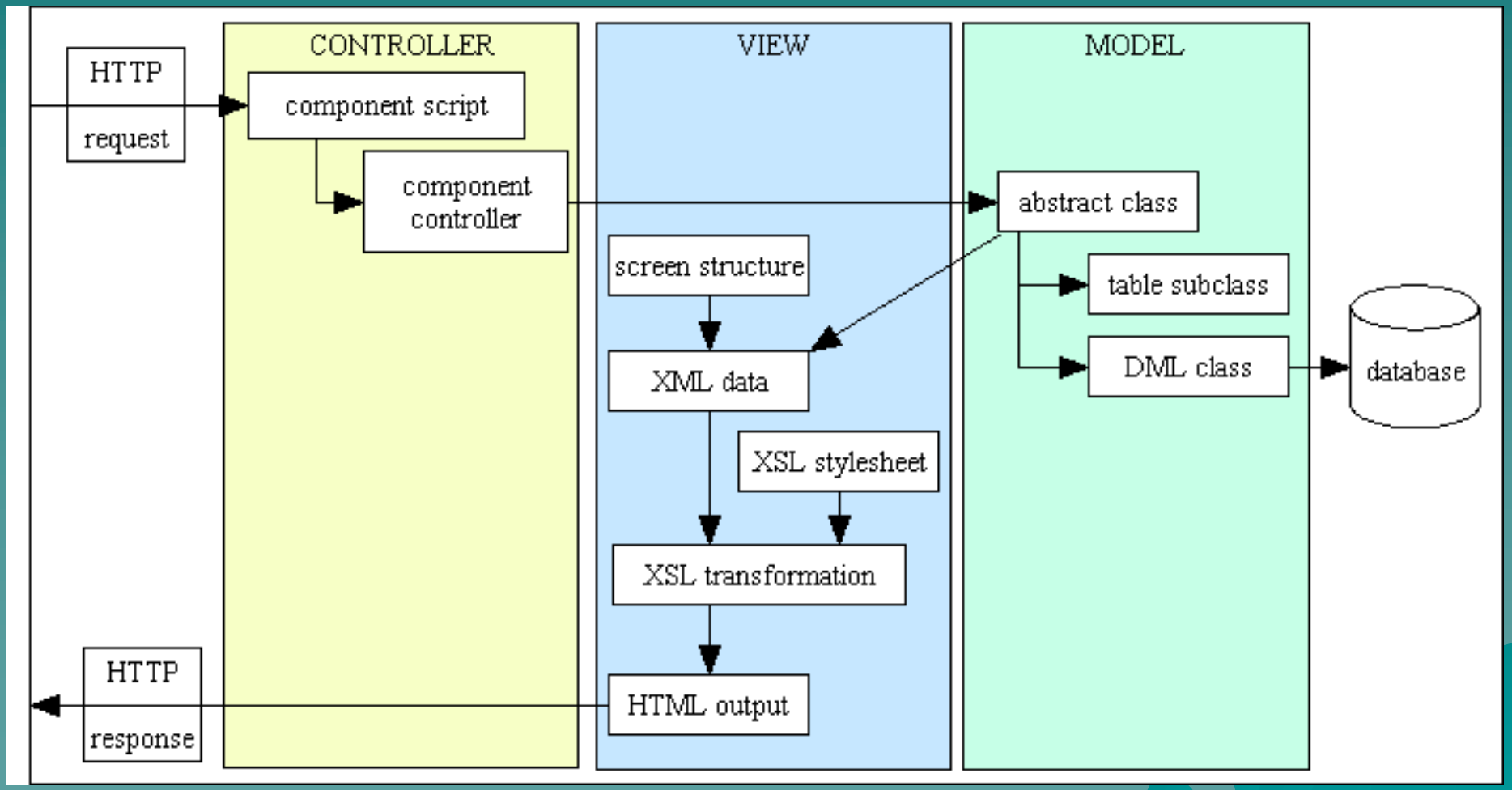


3.1. Your own framework (2)





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
- But I ask to NOT USE any framework

Question?