Model View Controller

Design Architecture

Douglas Putnam

What is the MVC

We are building an web application that uses the Model-View-Controller (MVC) design architecture.

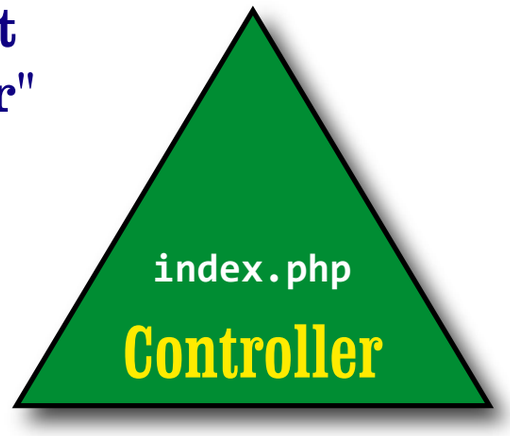
The MVC has become popular in web site design for several reasons.

- The Model (Business Tier) encapsulates application data, application flow, and business logic. Which means, the Model handles the database/persistence functionality.
- The View (Presentation Tier) and formats data from the Model for presentation.
- The Controller receives the Request input, directs the application flow, and translates it for the Model and View

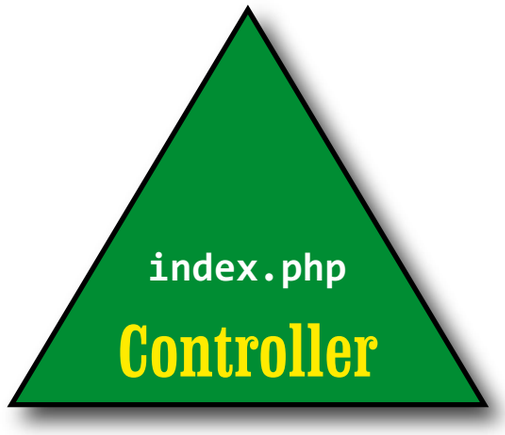
Our Version of the MVC

Our Version of the MVC

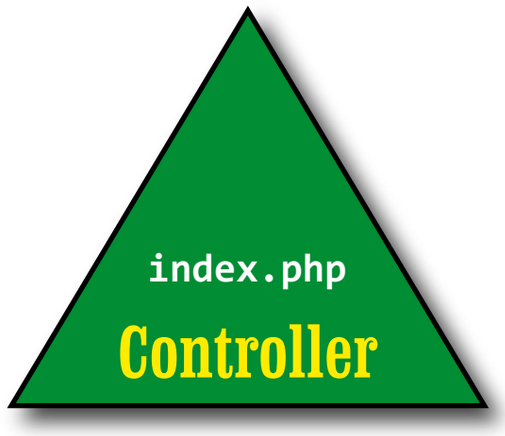
In our project the "controller" is `index.php`



Our Version of the MVC



Our Version of the MVC

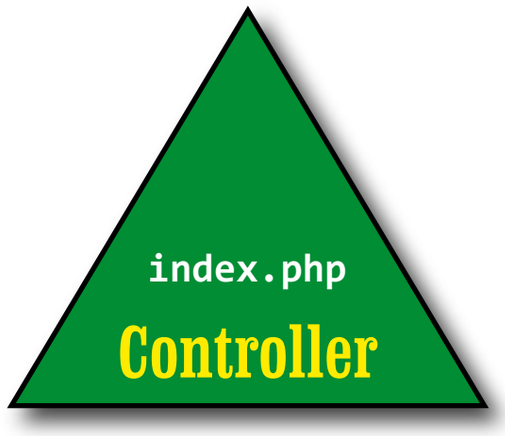


"**Views**" are kept in `phplib/views/`

```
<?php
echo $content_for_view;
?>
```

View

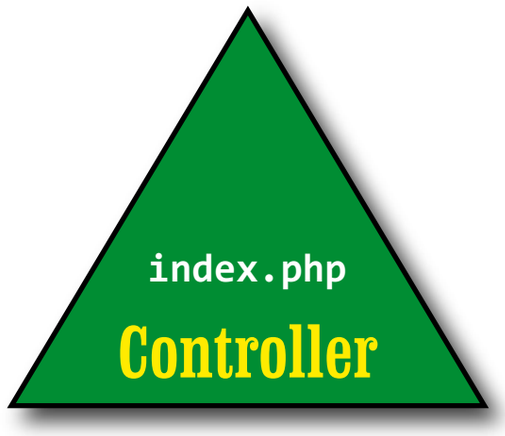
Our Version of the MVC




```
<?php
echo $content_for_view;
?>
```

View

Our Version of the MVC



Our project "models" are kept in `phplib/models/`

```
<?php
$content_for_view ='Hi';
?>
```

Model

```
<?php
echo $content_for_view;
?>
```

View

Our Version of the MVC

index.php

Controller

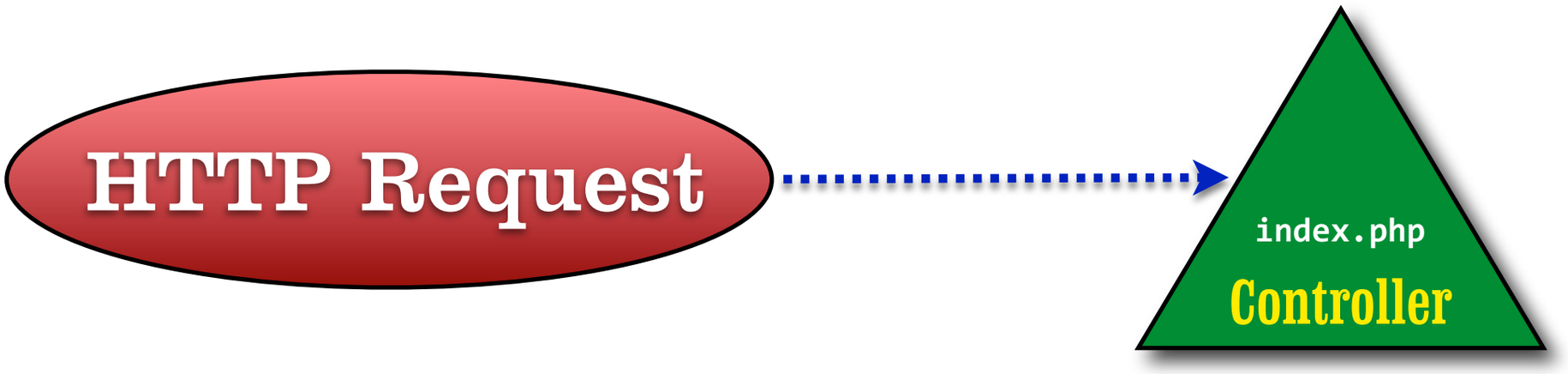
```
<?php
$content_for_view ='Hi';
?>
```

Model

```
<?php
echo $content_for_view;
?>
```

View

Our Version of the MVC



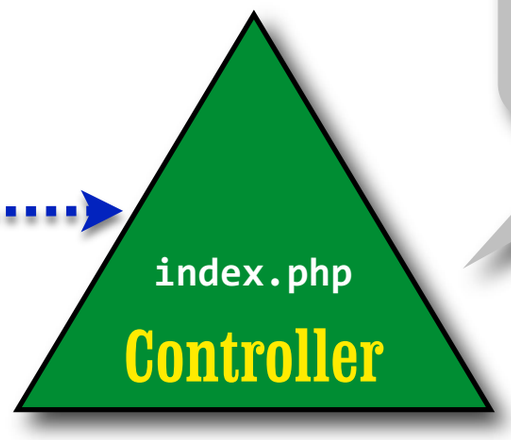

```php
<?php
$content_for_view ='Hi';
?>
```

Model

```php
<?php
echo $content_for_view;
?>
```

View

Our Version of the MVC



Upon receiving the request, the controller decides which view to display, and which model to run

```php
<?php
$content_for_view ='Hi';
?>
```

Model

```php
<?php
echo $content_for_view;
?>
```

View

Our Version of the MVC

HTTP Request

index.php

Controller

UPON RECEIVING THE REQUEST, THE CONTROLLER DECIDES WHICH VIEW TO DISPLAY, AND WHICH MODEL TO RUN

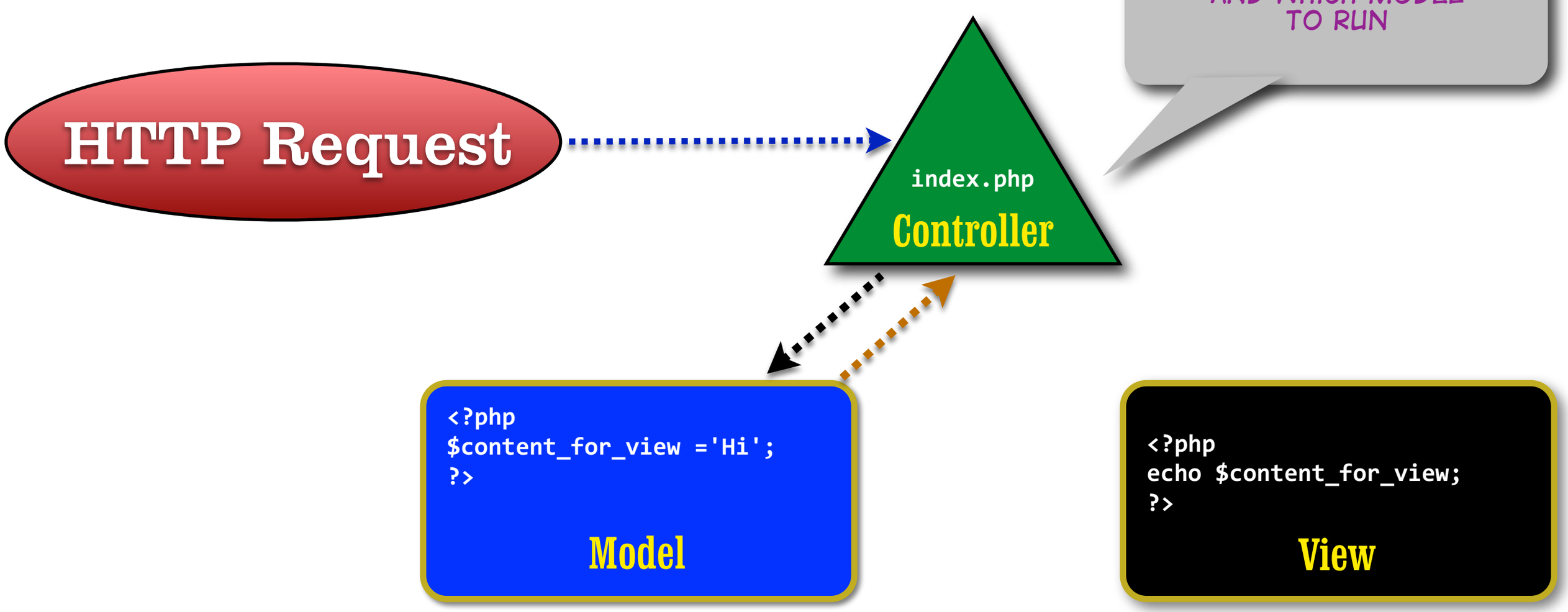
```
<?php
$content_for_view ='Hi';
?>
```

Model

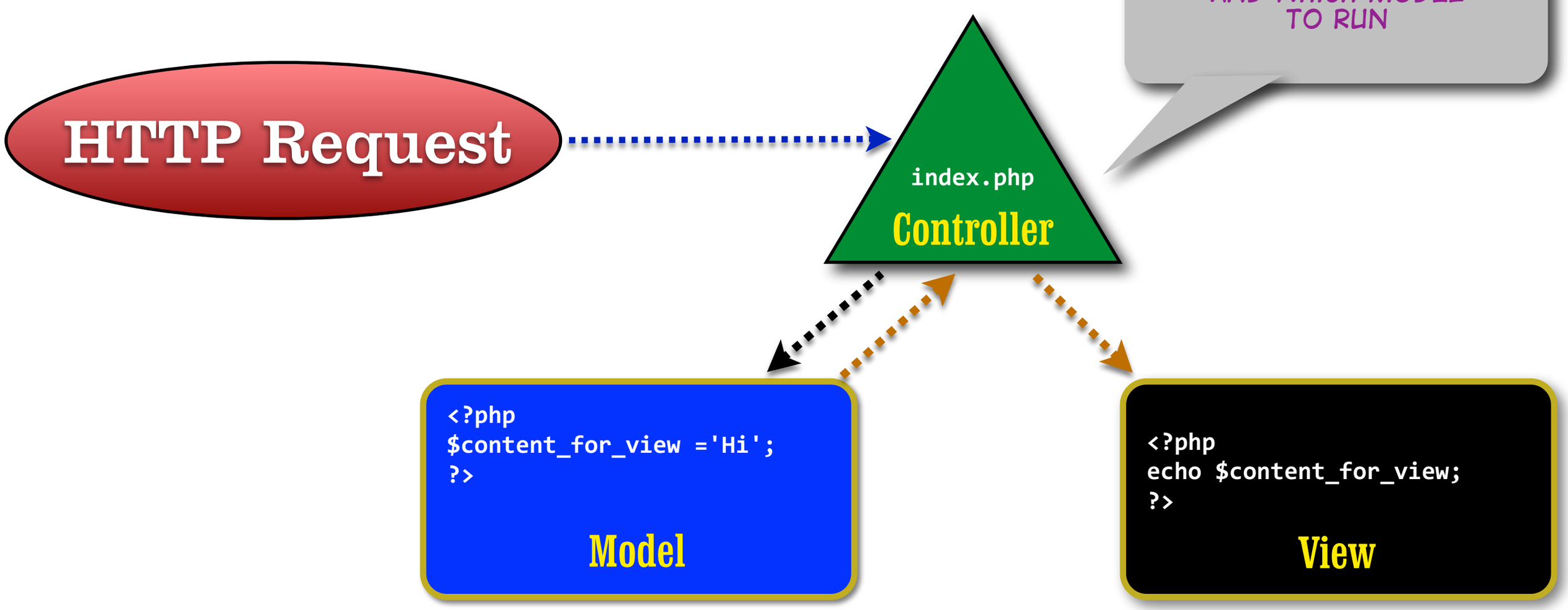
```
<?php
echo $content_for_view;
?>
```

View

Our Version of the MVC



Our Version of the MVC



Our Version of the MVC

UPON RECEIVING THE REQUEST, THE CONTROLLER DECIDES WHICH VIEW TO DISPLAY, AND WHICH MODEL TO RUN

HTTP Request

index.php

Controller

```
<?php
$content_for_view ='Hi';
?>
```

Model

```
<?php
echo $content_for_view;
?>
```

View

This is the output of my clever program

Separating Model and View

```
<!-- ecards.html -->
<table>
  <tr>
    <td>
<?php
  $files = get_file_list($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

Separating Model and View

PHP allows us to mix PHP code into HTML or text pages. In the MVC design architecture we take steps to separate code and presentation.

```
<!-- ecards.html -->
<table>
  <tr>
    <td>
<?php
  $files = get_file_list($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

Separating Model and View

PHP allows us to mix PHP code into HTML or text pages. In the MVC design architecture we take steps to separate code and presentation.

ALTHOUGH THIS CODE "WORKS", IT DOES NOT FOLLOW THE IDEA OF SEPARATING MODEL AND VIEW

```
<!-- ecards.html -->
<table>
  <tr>
    <td>
<?php
  $files = get_file_list($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

A Better Way

ecards.html

A Better Way

In our project, we will separate the PHP code from the View.

`ecards.html`

A Better Way

In our project, we will separate the PHP code from the View.

`ecards.html`

A Better Way

In our project, we will separate the PHP code from the View.

ecards.html

```
<!-- ecards.html -->

<?php
  // ecards.php in phplib/models/
  $files = read_files($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>

<table>
  <tr>
    <td>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

A Better Way

In our project, we will separate the PHP code from the View.

ecards.html

```
<!-- ecards.html -->

<?php
  // ecards.php in phplib/models/
  $files = read_files($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>

<table>
  <tr>
    <td>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

ecards.php

```
<?php
  // ecards.php in phplib/models/
  $files = read_files($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>
```

A Better Way

In our project, we will separate the PHP code from the View.

ecards.html

```
<!-- ecards.html -->









<table>
  <tr>
    <td>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

ecards.php

```
<?php
  // ecards.php in phplib/models/
  $files = read_files($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>
```

A Better Way

In our project, we will separate the PHP code from the View.

ecards.html

```
<!-- ecards.html -->


<?php
  include MODEL_DIR.'ecards.php';
?>


<table>
  <tr>
    <td>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

ecards.php

```
<?php
  // ecards.php in phplib/models/
  $files = read_files($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>
```

An Even Better Way

ecards.html

An Even Better Way

We can have each view include its own model code. We simply put an include into every view page. Is there any way we can automate this procedure?

```
ecards.html
```

An Even Better Way

We can have each view include its own model code. We simply put an include into every view page. Is there any way we can automate this procedure?

ecards.html

An Even Better Way

We can have each view include its own model code. We simply put an include into every view page. Is there any way we can automate this procedure?

ecards.html

```
<!-- ecards.html -->

<?php
  // ecards.php in phplib/models/
  $files = read_files($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>

<table>
  <tr>
    <td>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

An Even Better Way

We can have each view include its own model code. We simply put an include into every view page. Is there any way we can automate this procedure?

ecards.html

```
<!-- ecards.html -->

<?php
  // ecards.php in phplib/models/
  $files = read_files($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>

<table>
  <tr>
    <td>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

ecards.php

An Even Better Way

We can have each view include its own model code. We simply put an include into every view page. Is there any way we can automate this procedure?

ecards.html

```
<!-- ecards.html -->

<?php
  // ecards.php in phplib/models/
  $files = read_files($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>

<table>
  <tr>
    <td>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

ecards.php

```
<?php
  // ecards.php in phplib/models/
  $files = read_files($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>
```

An Even Better Way

We can have each view include its own model code. We simply put an include into every view page. Is there any way we can automate this procedure?

ecards.html

```
<!-- ecards.html -->









<table>
  <tr>
    <td>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

ecards.php

```
<?php
  // ecards.php in phplib/models/
  $files = read_files($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>
```

An Even Better Way

We can have each view include its own model code. We simply put an include into every view page. Is there any way we can automate this procedure?

ecards.html

```
<!-- ecards.html -->
```

```
<?php
  include MODEL_DIR.'ecards.php';
?>
```

```
<table>
  <tr>
    <td>
    <?php echo $html; ?>
    </td>
  </tr>
</table>
```

ecards.php

```
<?php
  // ecards.php in phplib/models/
  $files = read_files($ecard_dir);
  foreach($files as $f) {
    $html = '<a href .... >';
  }
?>
```

Is There a Better Way?

Is There a Better Way?

We can have each view include its own model code. We simply put an include into every view page.

Is There a Better Way?

We can have each view include its own model code. We simply put an include into every view page.

Is there any way to automate this procedure?

Our Friend index.php

```
if(isset($_GET) && isset($_GET['v']) ) {
  if(file_exists(VIEW_DIR . basename($_GET['v']) . '.html')) {
     /* Look for model code associated with $VIEW */
     if(file_exists(MODEL_DIR . basename($_GET['v']))) {
         include MODEL_DIR . basename($_GET['v']);
     }
    $VIEW = basename($_GET['v']) . '.html';
  }
  else {
    $VIEW = 'error_404.html';
  }
}
```

Our Friend index.php

We can have index.php include the model at the time it determines the view.

```
if(isset($_GET) && isset($_GET['v']) ) {
  if(file_exists(VIEW_DIR . basename($_GET['v']) . '.html')) {
     /* Look for model code associated with $VIEW */
     if(file_exists(MODEL_DIR . basename($_GET['v']))) {
         include MODEL_DIR . basename($_GET['v']);
     }
    $VIEW = basename($_GET['v']) . '.html';
  }
  else {
    $VIEW = 'error_404.html';
  }
}
```

Our Friend index.php

We can have index.php include the model at the time it

We check to see if there is a model named $VIEW.php and include it.

```
if(isset($_GET) && isset($_GET['v']) ) {
  if(file_exists(VIEW_DIR . basename($_GET['v']) . '.html')) {
     /* Look for model code associated with $VIEW */
     if(file_exists(MODEL_DIR . basename($_GET['v']))) {
         include MODEL_DIR . basename($_GET['v']);
     }
    $VIEW = basename($_GET['v']) . '.html';
  }
  else {
    $VIEW = 'error_404.html';
  }
}
```

END