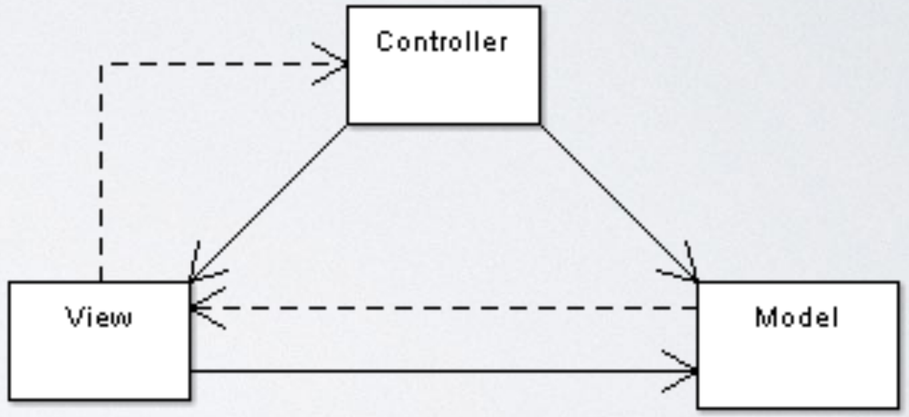
The Template Is Not The View

A Brief Introduction To Action-Domain-Responder

https://joind.in/11569

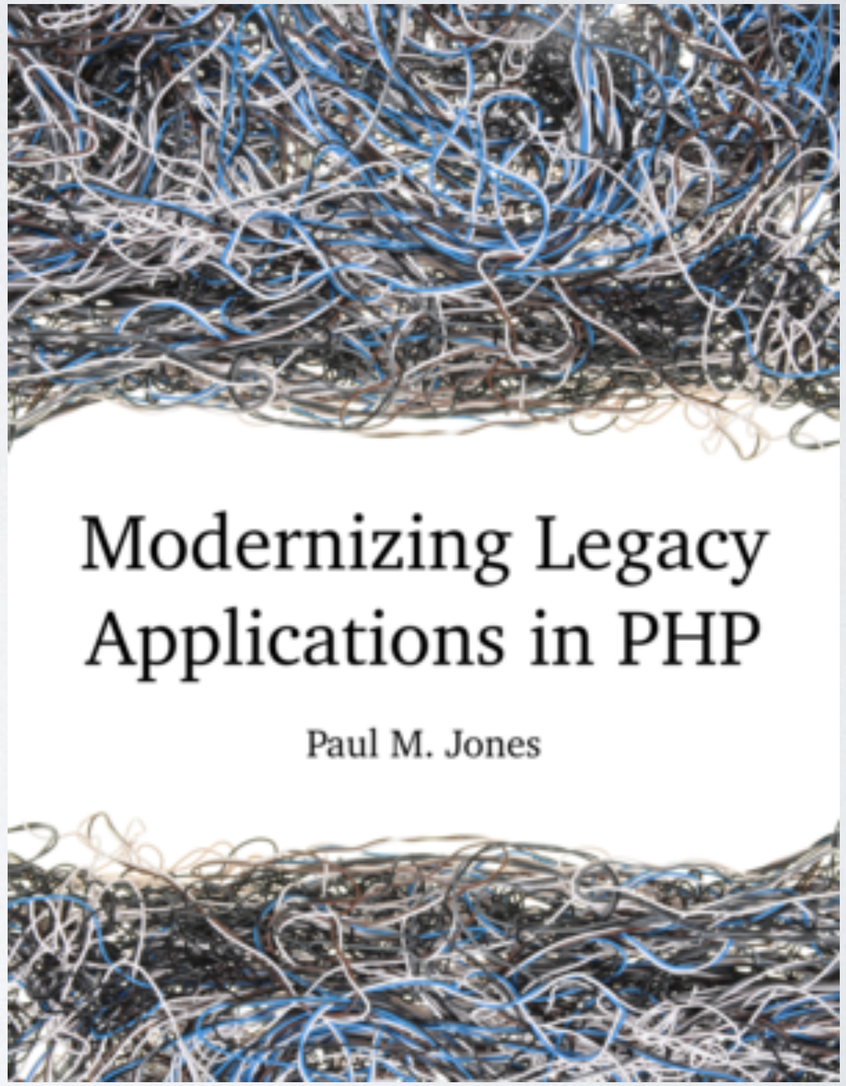
Model-View-Controller

- Separation of concerns
- Model contains business logic
- View contains presentation logic
- Controller puts them together
- One trio per screen element
- Not perfect but good-enough so far



Modernizing Legacy Applications

- Centralize classes and functions
- Replace globals with injection
- Extract SQL to Gateways
- Extract model logic to Transactions
- Extract view logic to Templates



Extract Presentation From This Page Script …

```php
<?php
require "includes/setup.php";

$current_page = 'articles';

include "header.php";

$id = isset($_GET['id']) ? $_GET['id'] : 0;
if ($id) {
    $page_title = "Edit An Article";
} else {
    $page_title = "Submit An Article";
}

?><h1><?php echo $page_title ?></h1><?php

$user_id = $user->getId();

$db = new Database($db_host, $db_user, $db_pass);
$articles_gateway = new ArticlesGateway($db);
$users_gateway = new UsersGateway($db);
$article_transactions = new ArticleTransactions(
    $articles_gateway,
    $users_gateway
);
```

```
if ($id) {
    $article_transactions->updateExistingArticle($user_id, $_POST);
} else {
    $article_transactions->submitNewArticle($user_id, $_POST);
}

$failure = $article_transactions->getFailure();
$input = $article_transactions->getInput();

?>

<?php
if ($failure) {
    $failure_text = implode("<br />\n", $failure);
    echo "<h2>Failure</h2>";
    echo "<p>We could not save the article.<br />";
    echo $failure_text. "</p>";
} else {
    echo "
        <h2>Success</h2>
        <p>We saved the article.</p>
    ";
}
?>
```

```
<form method="POST" action="<?php echo $_SERVER['PHP_SELF']?>">

    <input type="hidden" name="id" value="<?php echo $id ?>" />

    <h3>Title</h3>
    <input type="text" name="title" value="<?php
        echo $input['title']
    ?>" size="100">


    <h3>Article</h3>
    <textarea name="body" cols="80" rows="30"><?php
        echo stripslashes($input['body'])
    ?></textarea>

    <h3>Ratings</h3>
    <p>How many rated reviews do you want?</p>
    <select name='max_ratings'>
        <?php for ($i = 1; $i <= 10; $i ++) {
            echo "<option value='$i' ";
            if ($input['max_ratings'] == $i) {
                echo 'selected="selected"';
            }
            echo ">$i</option>\n";
        } ?>
    </select>
```

```
    <p>How many credits will you give for each rating?</p>
     <input type='text' name='credits_per_rating' value='<?php
         echo $input['credits_per_rating'];
     ?>' size='5' />

     <h3>Notes for Reviewers</h3>
     <input type="text" name="notes" value="<?php
         echo $input['notes']
     ?>" size="100">
     <label><input type="checkbox" name="ready" <?php
         echo $input['ready'] ? 'checked="checked"' : '';
     ?> /> This article is ready to be rated.</label>

     <p align="center">
         <input type="submit" value="Save" name="submit">
     </p>

</form>

<?php
include "footer.php";
?>
```

… To A Template Call

```php
<?php
require "includes/setup.php";

$user_id = $user->getId();

$db = new Database($db_host, $db_user, $db_pass);
$articles_gateway = new ArticlesGateway($db);
$users_gateway = new UsersGateway($db);
$article_transactions = new ArticleTransactions(
    $articles_gateway,
    $users_gateway
);

$id = isset($_GET['id']) ? $_GET['id'] : 0;
if ($id) {
    $article_transactions->updateExistingArticle($user_id, $_POST);
} else {
    $article_transactions->submitNewArticle($user_id, $_POST);
}

$template = new Template('/path/to/app/views');
$template->setView('articles.html.php');
$template->setVars(array(
    'id' => $id,
    'failure' => $article_transactions->getFailure(),
    'input' => $article_transactions->getInput(),
    'action' => $_SERVER['PHP_SELF'],
));
```

But What About Headers?

```
$template = new Template('/path/to/app/views');
$template->setView('articles.json.php');
$template->setVars(array(
    'id' => $id,
    'failure' => $article_transactions->getFailure(),
    'input' => $article_transactions->getInput(),
    'action' => $_SERVER['PHP_SELF'],
));
header('Content-Type: application/json');
```

The Server Presents A **Response**, Not A Template

- Client receives HTTP response of **both** body **and** headers
- This means the View in server-based MVC is **not** the template
- The View in server-based MVC is the **Response**

Suboptimal Separation Of Concerns In Typical MVC Applications

- Template Views generally build HTTP **body** values
- Remaining Controller logic manipulates HTTP **header** values
- Presentation is mixed between Views and Controllers

“Responder”

- Should set headers, cookies, etc. from within the presentation layer
- Responder implementation handles setting headers, status, etc
- Additionally uses templates for body content
- Controller method invokes Responder for the View, not a template

One Responder Per Controller Method

- Each Controller action method has its own set of status codes
- index(), create(), read(), update(), delete()
- Already have a template for each one
- IndexResponder, CreateResponder, ReadResponder, etc.

“Action” and “Domain”

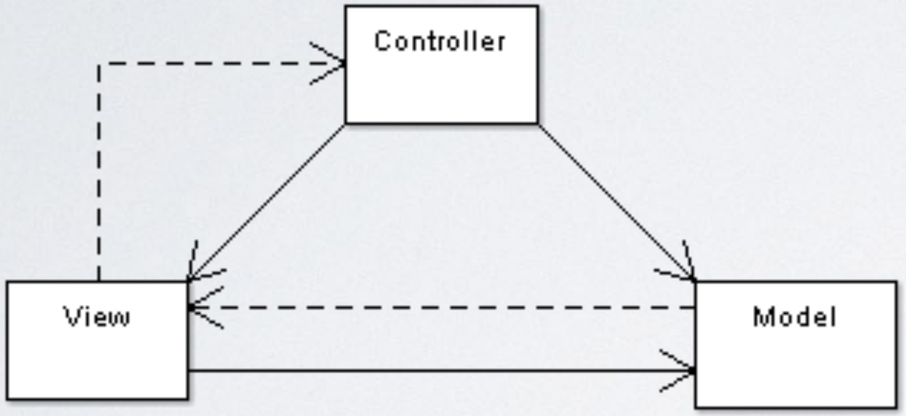
- Inject one Responder per action method?
- Instead of a Controller with index(), create(), read(), etc. …
- … one class per Action: IndexAction, CreateAction, ReadAction, etc.
- (“Domain” == “Model”, think Domain Driven Design)

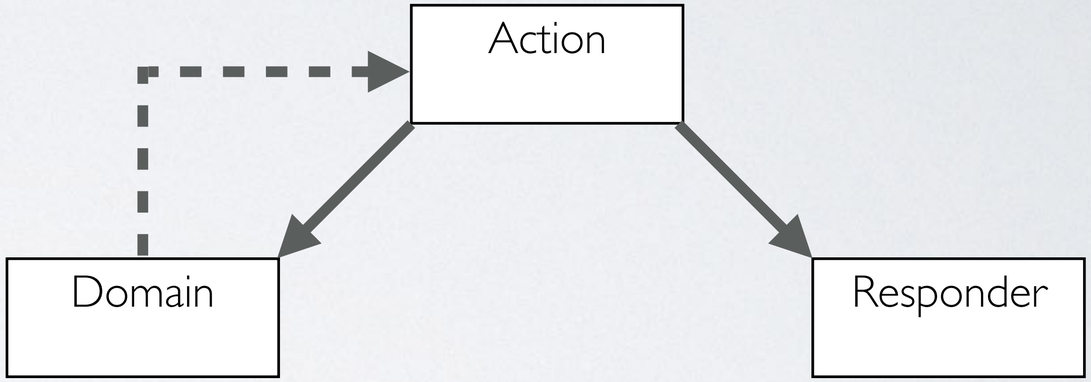
```php
<?php
namespace Blog\Action;

use Aura\Web\Request;
use Blog\Domain\BlogService;
use Blog\Responder\BlogReadResponder;

class BlogReadAction extends AbstractBlogAction
{
    public function __construct(
        Request $request,
        BlogService $domain,
        BlogReadResponder $responder
    ) {
        $this->request = $request;
        $this->domain = $domain;
        $this->responder = $responder;
    }

    public function __invoke()
    {
        $id = $this->request->params->get('id');
        $this->responder->blog = $this->domain->fetchOneById($id);
        return $this->responder->buildResponse();
    }
}
```

Action-Domain-Responder

- Action feeds input from HTTP request to a Domain layer
- Action feeds output from Domain layer to a Responder
- Responder builds the HTTP response headers and body

MVC vs ADR





Thanks!

- http://pmjones.io/adr (ADR Paper)
- https://leanpub.com/mlaphp (Modernizing Legacy Apps in PHP)
- http://paul-m-jones.com/ and @pmjones
- https://joind.in/11569