CodeIgniter - [Overview]

CodeIgniter is a fast, minimalistic PHP 4/5 framework.

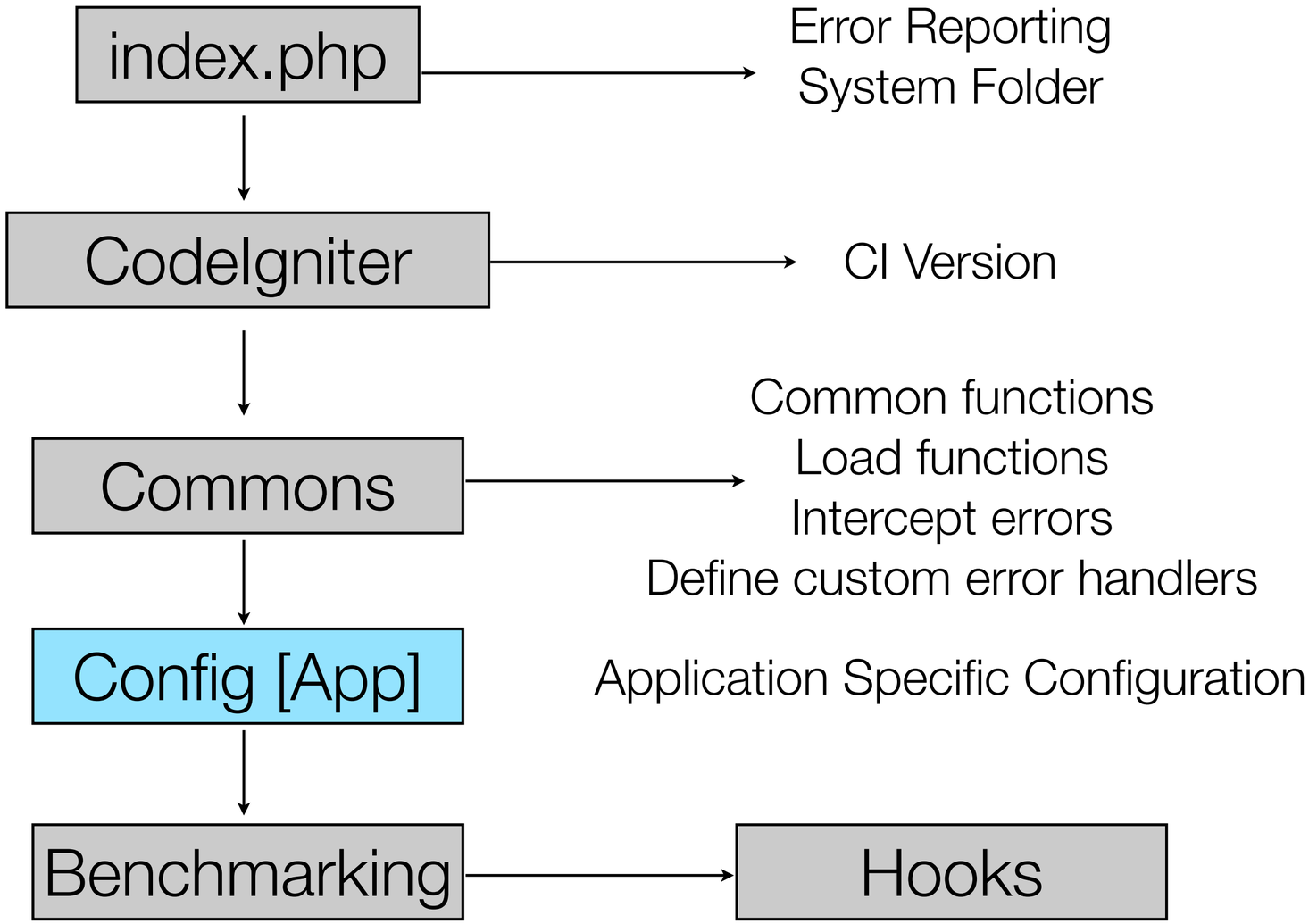
MVC

- CodeIgniter is built on a design pattern called MVC.
- MVC stands for Model View Controller and is a common design pattern for web design.
- Models encapsulate data and the manipulation thereof.
- Views handle presentation of the Models.
- Controllers handle routing, external libraries, and deciding which Model it’s going to send to which View.

“Under the Hood”

- CodeIgniter works by building a “super object” on every call.
- This makes the flow very easy to follow.

CodeIgniter Flow

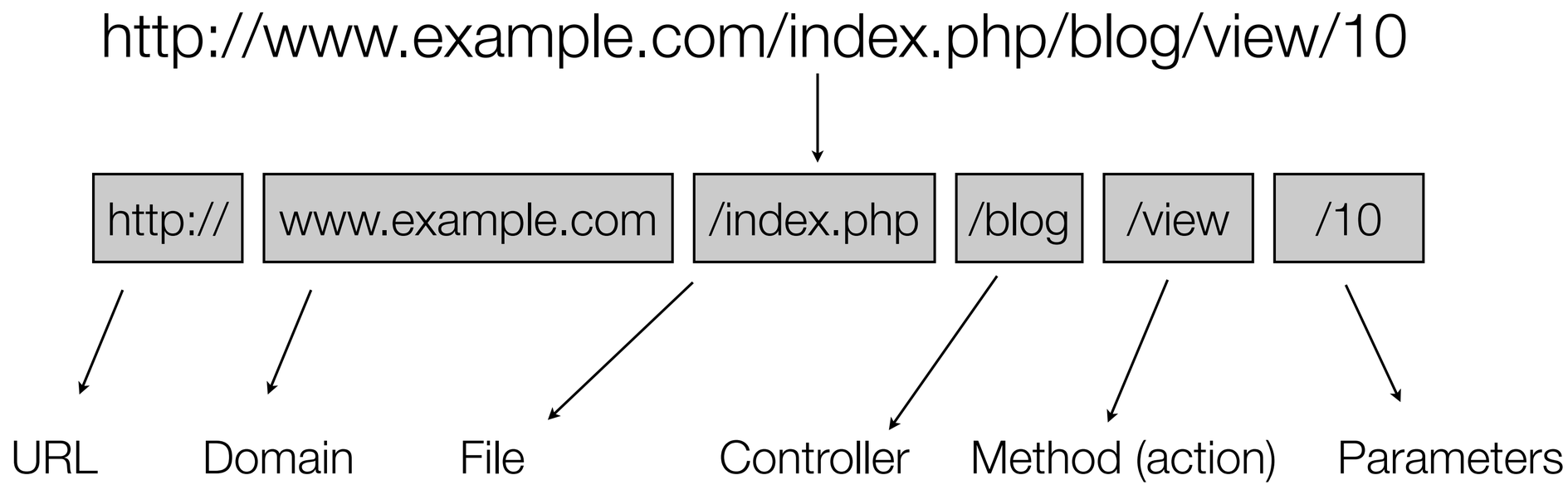


CodeIgniter Flow

- Questions?

URI Patterns

- URL's are URI's. URI's are Uniform Resource Identifier
- CodeIgniter uses the URI to determine what action to perform with the request.



Application Flow

- Controller objects are created
    - This loads any libraries / helpers needed as well as any Models you intend to reference
- Corresponding named function (action) is called and given any parameters present
    - This function typically loads views which are then displayed

¡Expert Tip!

The unsightly index.php from the URI can be removed with a .htaccess rule

```
RewriteEngine on                                    # Turn RewriteEngine on if needed
RewriteCond $1 !^(index\.php|images|css|js)         # Match everything except our non php files
RewriteCond %{HTTP_HOST} ^example\.com              # Only match one specific vhost
RewriteRule ^(.*)$ /blog/index.php/basic/$1         # Magic!
```

MVC Design with CodeIgniter

- CodeIgniter enforces MVC if a pretty lenient way.
- You could write all your logic in Views if you wanted to.
- You can completely disregard Models.
- **You shouldn’t misuse MVC.**
- It leads to problems down the road. Even if you aren’t planning on extending the Application.

Controllers

- Controllers get called in response to a URL route
- Controllers should take any given parameters and use them to create a Model.
- This Model then gets passed to the View.

- Do all Views have Models? Ideally, yes. However this is often impractical.
- Use private functions to encapsulate repetitive logic.
    - Are you checking for $this->session->user every time and redirecting if not found? Put it into a private function and call $this->_require_user() instead.

Models

- Models are objects. They typically interface with a data source.
- Active Record gives Models database connectivity and query ability.
- DataMapper (OZ) is has additional features *if you need them.*

- Simplicity to the point of insanity.

Views

- Views are almost 100% HTML.
- A view takes information passed to it and splices those values into the HTML document.
- Views are modular, you can load multiple views for any given request.
- You can load views from other views
- Create a directory structure that makes sense.

¡Expert Tip!

- Don’t use PHP short tags. They are from the devil.
- Don’t write logic in your views. If this show that: Fine. If this show that create new User: NO!
- Don’t ‘echo’ out HTML. This does not turn into a JSP where everything is escaped and written out anyway. This is HTML that is pre-processed by the PHP engine before being served.
- Do write separate views for separate things. Listing a bunch of blog posts?

```
foreach($posts as $post)
{
   $this->load->view(‘shared/post’, $post);
}
```

Libraries

- Not full of books.
- Libraries allow you to interface with external classes.
    - Session management
    - reCaptcha
    - API’s
    - Benchmarking, Calendar, Encryption, Forms, Profilers, URI, Validation, *etc.*
- CodeIgniter has some very useful ones built in. Check them out.

Helpers

- Globally available functions.
- Use them in Controllers, Views, or Models.

Tying It All Together

- CodeIgniter is a simple framework for you to build complex applications.
- Questions?