Stages for learning a programming language

* Basic structure and syntax
* Vocabulary and functionality
* Techniques and relations
* Scalable structures and architecture
* Design patterns, APIs
* Focus on delivering value

Keep in Mind

* We may ignore to have a model and/or a view file, but we must need a controller.

Introducing Codeigneiter

* Open-source PHP web application framework
* Rapid development
* Provides logical structure, reusable interface

Features

* Small footprint(lightweight)
* Fast performance
* Very little config

Why use a framework like Codeigneiter?

* Reusability and modularity
* Avoid copy/paste
* Components can be reused on multiple projects
* Maintainability
* Others can build, extend your application
* Delegation
* Focus on delivering value, not creating building blocks
* Organize code, logic

MVC

* Software design pattern
* Separates representation from interaction
* Code reusability, separation of responsibilities
* Organize code
* Codeigneiter uses loose MVC

Model

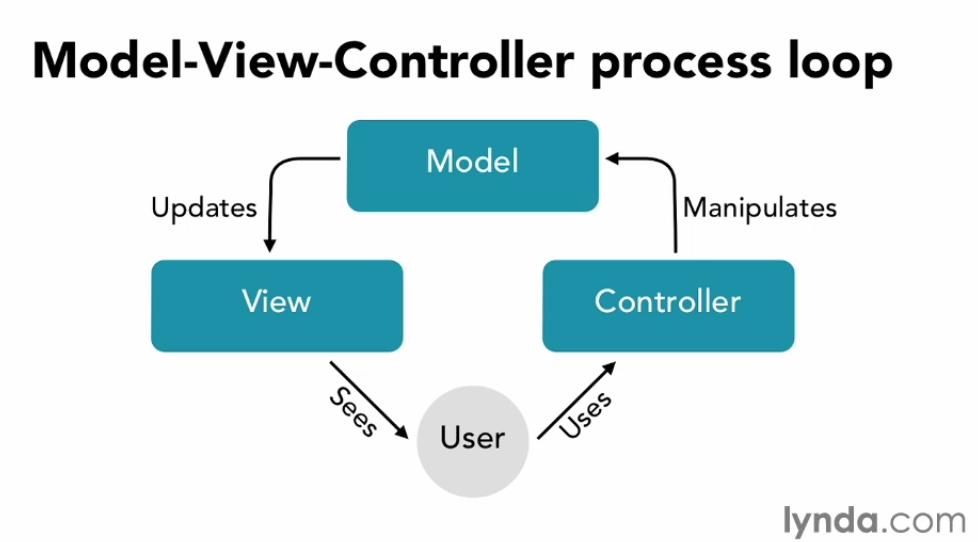
* Representation of data, rules and functions
* CRUD functionality

View

* Representation of info presented to user
* HTML rendering
* Simple if/then logic

Controllers

* Intermediary between model and view
* Sends commands e.g. – Model, save yourself e.g. View, show message



Codeigneiter and MVC

* Loose approach to MVC
* Codeigneiter’s models are untraditional
* Similar to domain objects (collection of properties)
* Models are optional, can be left out
* Model will be demonstrated

Codeigneiter Working Process

1. All configuration file resides under application/config folder.
2. To find database connection in all pages write database in $autoload[‘libraries’] = array(‘database’);

Controllers

* Class file
* Delegates work
* Named for access by URL
* /index.php/[controller-class]/[controller-method]/[arguments]
* E.g. <http://localhost/Codeigniter/MyApp/index.php/magazine>

View

* Partial or complete web page
* Mostly HTML, simple PHP
* Can be embedded in other views
* Hierarchies
* Cannot be called directly
* Loaded by controller

Active Record Class

* Codeigneiter uses a modified version of the Active Record Database Pattern. This pattern allows information to be retrieved, inserted, and updated in your database with minimal scripting.

$this

* Singleton Codeigneiter instance
* There can only be one throughout the entire application
* When functionality is loaded:
* Attached to instance
* Referenced like a property
* Model
* Application data
* Logic and business rules
* Functions
* PHP classes that work with information in database
* Collection of properties
* Similar to domain objects
* Named the same as database columns – 1:1
* No other functionality by default
* Create – Set default values
* Read – retrieve from database, and populate model
* Update – record in database saved
* Destroy – record in database deleted
* Acronym: CRUD
* Define database table, primary key
* CRUD methods
* Represent back issues of a magazine
* Two models:
* Publication
* Unifying record
* Issues

Model: Publication

* Publication\_id(int)
* Publication\_name(varchar)

Model: Issue

* Issue\_id (int)
* Publication\_id (int)
* Issue\_number (int)
* Issue\_date\_publication (date)
* Issue\_cover (varchar)
* Model & Controllers file name are in lower case

Codeigneiter active records

* Don’t need to create a class to connect to each table
* You should create model for each table
* Use Active Record in model to perform DB operations
* Database agnostic
* Query syntax generated automatically

Accessing the database

* $this->db->…
* Insert(), update(), delete() …
* Demonstrate by example
* Add persistence to models
* Extend CI\_Model
* Add CRUD
* My\_Model
* Codeigneiter requires the loading the model before using the model by using the method load.
* Leveraging Codeigneiter views
* Separate presentation from logic
* Pass data to view for rendering
* Either array or object
* Keys/parameters are variables within view
* Same name
* View for rendering a magazine
* Both Issue and Publication
* Use an array
* More complex (will not demo)
* Lazy instantiation to load Publication from Issue
* Codeigneiter common functions
* Available at any point in execution
* Html\_escape()
* Shortcut for htmlspecialchars
* Accepts string and arrays

Codeigneiter URL

