

ColdBox from Zero To Hero

Description

In this workshop you will be introduced to the latest version of the most popular CFML MVC framework; **ColdBox 5**. We will go over the basics of installation, templating and configuration to the most advanced features like HMVC development with modules, RESTful APIs, integration testing, interception points and much more.

Prerequisites

- Latest CommandBox CLI (You don't need to know how to use it.)
- MySQL GUI
 - <https://sequelpro.com>
 - <https://www.heidisql.com/>
 - <https://sqlectron.github.io>
- A local database server - We will be using MySQL **5.7** in the course. MAKE SURE IT IS 5.7. You can start a local database service with docker using the following command if needed.

```
docker run -d \  
  --name soapbox \  
  -p 33306:3306 \  
  -e MYSQL_ROOT_PASSWORD=root \  
  -e MYSQL_DATABASE=soapbox \  
  mysql:5
```

You can then stop and start the database server with ``docker stop soapbox`` and ``docker start soapbox``.

Schedule / Outline

Project Overview

This course will focus on building a real twitter clone; SoapBox, based on different concepts and tool methodologies. We will use ColdBox 5 features, including Behavior Driven Development (BDD) testing and leveraging several ForgeBox modules.

1. Course Introduction

- Introductions
- Software Pre-Requisites

- Course Expectations

2. App Skeleton

- Scaffold a ColdBox Template using CommandBox
- Build and configure the Test Harness
- Build and configure Test Runners
- CommandBox Test Watchers

3. Intro to ColdBox MVC

- ColdBox.cfc Intro
- Development Settings
- Discovering application router
- Handlers
- What is the Request Context
- Views/Layouts by convention
- Reiniting your application

4. Layouts

- Create a bootstrap theme layout for our app

5. Database Migrations

- Intro to Migrations
- Installation of migration modules
- Migration Commands
- Intro to CommandBox Environment Variables
- Configure our environment variables
- Creating and running our first migration
- Configure application and test harness for database access
- Setup Test Harness and Base Spec

6. Intro to Models

- Scaffold a UserService
- Add our list story to our integration test
- Implement the list() method to retrieve all users
- Inject our UserService into our Main Handler
- Call the list() method from our Main Handler and dump the data
- Access the data returned from your main.index view.

7. Building the Registration Flow

- Create the registration spec
- Install bcrypt
- Write the registration code as a ColdBox resource with appropriate spec and model methods

8. Login & Logout Flow

- Install cbmessagebox
- Create the specs
- Create routes
- Create the User Sessions handler
- Create the user login screen
- Install cbauth
- Scaffold a User model
- Update User Service for cbauth usage
- Update specs for login/logout actions
- Update handlers for login/logout actions
- Update layouts for login usage
- Update spec for registration with auto-login
- Create auto-login with registration
- Customize MessageBox
- Leverage MessageBox for messages

9. Rants

What we will do:

- Create new migration for rants
- Scaffold a rant resource
- Update the resource route
- Update the Rant object
- Update the Rantservice
- Create the basic unit tests for both and talk about pragmatism of unit test vs bdd
- Build and Test the CRUD setup.
- Change application default event to the rants
- Update the main layout for adding a rant

10. Security

- Installing cbsecurity
- Create the security rules
- Update the User Service for user validation

- Confirm security

11. View User Rants

- Create the spec for the profile page
- Add routes for the profile
- Update the rant service to get rants per user
- Create the handler
- Create a reusable viewlet for our rants

12. Add Rant Actions

- Create the migrations for bumps and poops
- Scaffold a Reaction Service for the actions
- Update Rant object for reactions
- Update Tests
- Update the rant viewlet
- Learn about Wirebox Convention vs Configuration

13. Make Reactions Functional

- Update the viewlet
- Showcase HTML Helper and its usages
- Create spec for tracking bumps in the User and ReactionService
- Create the get reaction methods in the reaction service
- Create the hasReaction methods in the user
- Create new routes and handlers
- Create the integration tests
- Implement the handlers