Step by step guide to upgrading procedural code to MVC & Why you should do it!

Ortus Solutions
A Software Revolution

Adobe ColdFusion CF Summit 2022

Gavin Pickin
Ortus Solutions

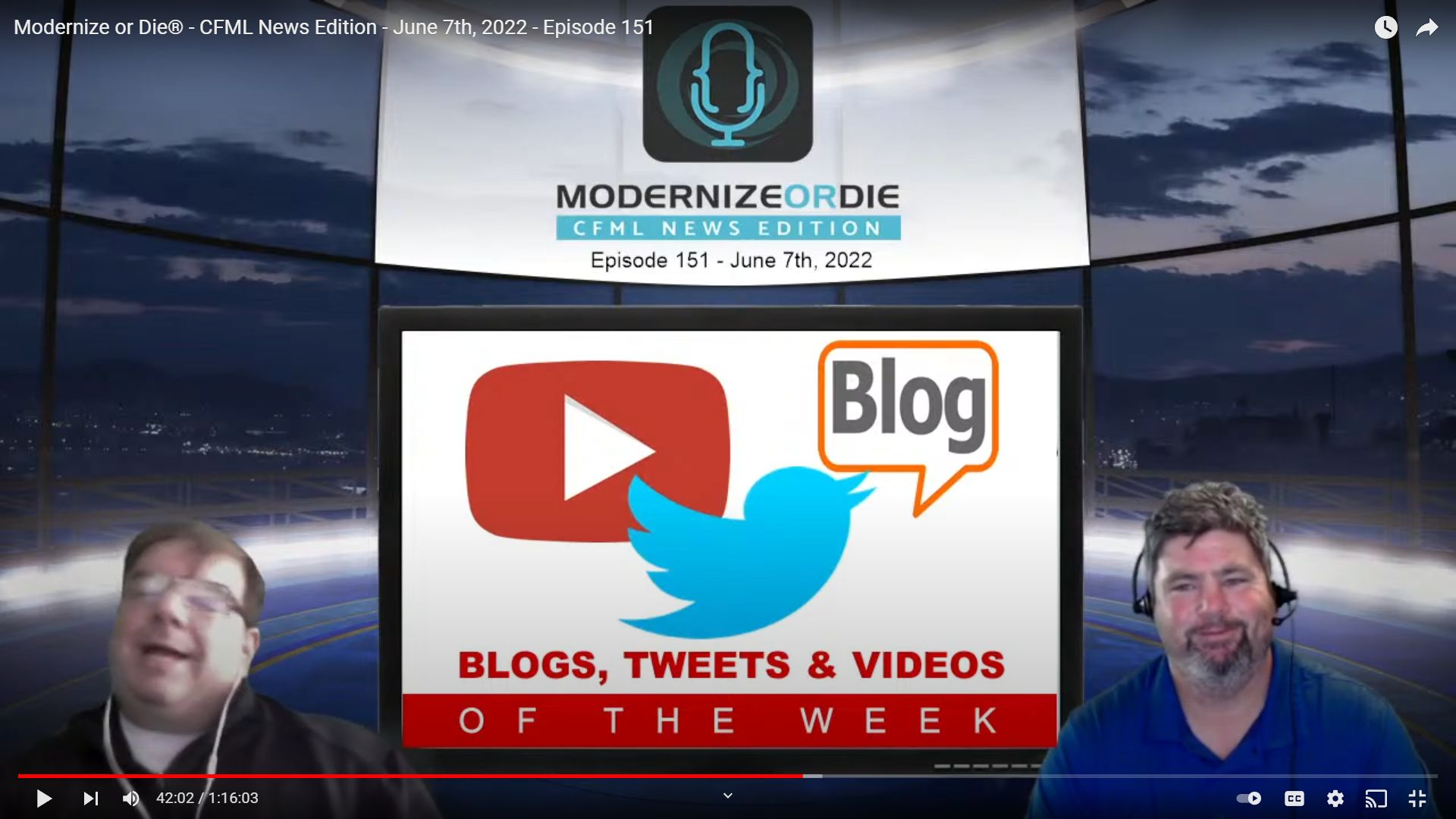
Who am I?

- Software Consultant for Ortus Solutions
- Work with ColdBox, CommandBox, …Box every single day
- Working with Coldfusion for 22 years
- Working with Javascript just as long
- Love learning and sharing the lessons learned
- From New Zealand, live in Bakersfield, Ca
- Loving wife, lots of kids, and countless critters

@gpickin on twitter

http://www.ortussolutions.com

Ortus Solutions
A Software Revolution


Modernize or Die® - CFML News Edition - June 7th, 2022 - Episode 151
MODERNIZEORDIE
CFML NEWS EDITION
Episode 151 - June 7th, 2022
Blog
BLOGS, TWEETS & VIDEOS
OF THE WEEK
42:02 / 1:16:03

Watch Now

Search

2022 VS Code Hint Tip and Trick of the Week

Beginner

As seen on the CFML News Podcast - Your Hosts will show off a VS Code Hint Tip and Trick of the week.

8 Videos 24 minutes

2022 ForgeBox Modules of the Week

Beginner

As seen on the CFML News Podcast - Your Hosts highlight a new ForgeBox Module of the Week

7 Videos 29 minutes

Publish Your First ForgeBox Package

Beginner

Learn alongside Gavin Pickin, how to publish your first ForgeBox package.

14 Videos 1 hour 16 minutes


KODING WITH THE KIWI+FRIENDS
Live Stream for Patreon Supporters

Play (k)

Planning and Building my Developer Feud Quiz API

with Gavin Pickin

Online ColdFusion Meetup:
Thurs July 7 2022 12pm US ET
298th episode

Your Host: Charlie Arehart
Spread the word: coldfusionmeetup.co
This meeting recording URL: the youtube me
Past meeting recordings: recordings.coldfusionn
Consider presenting: speak.coldfusionmeetu

CFMeetup recordings
Charlie Arehart, CArehart - 1 / 55

▶ 1:21:14 Planning and Building m Developer Feud Quiz AP
Charlie Arehart, CArehart

2 1:21:47 How to find, install and implement 3rd party lib
Charlie Arehart, CArehart

3 1:12:08 When Should I Use 3rd Libraries vs Roll My Ow
Charlie Arehart, CArehart

4 1:01:58 Code Reuse in CF: Is Sp Code still Spaghetti if it
Charlie Arehart, CArehart

Goals of this talk

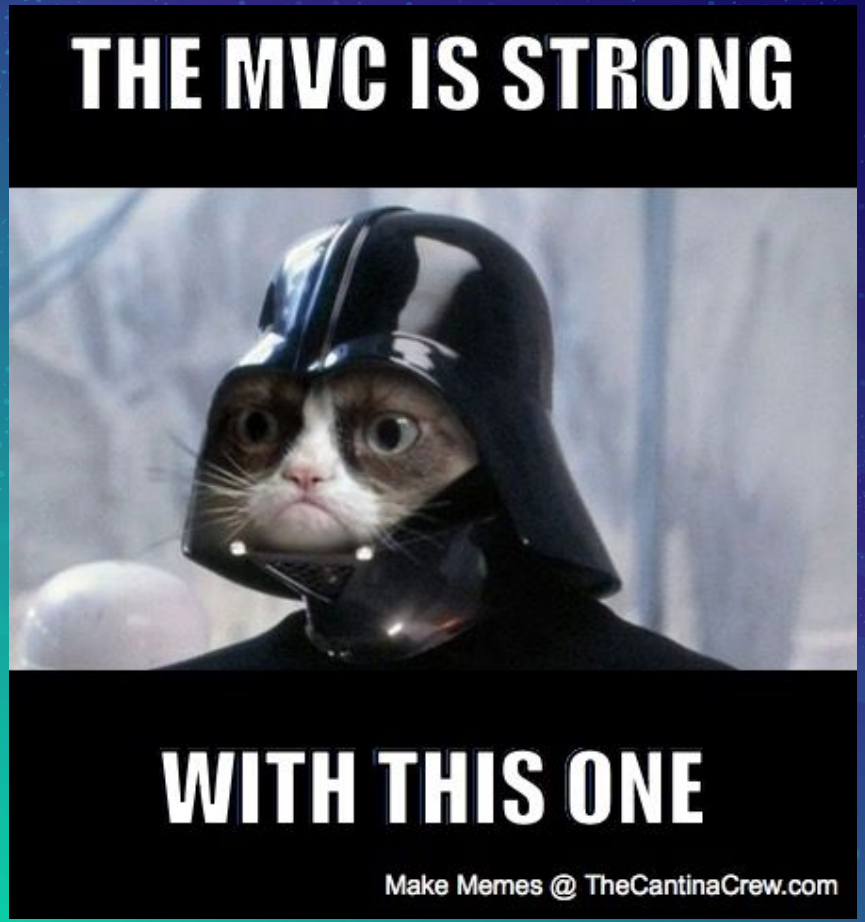
- Give you a little history on MVC
- Share a few PROs and Cons of MVC
- Look at the steps you can take to start using a MVC
- Get you inspired to give it a try

Goals of this talk

- Give you a little history on MVC
- Share a few PROs and Cons of MVC
- Look at the steps you can take to start using a MVC
- Get you inspired to give it a try

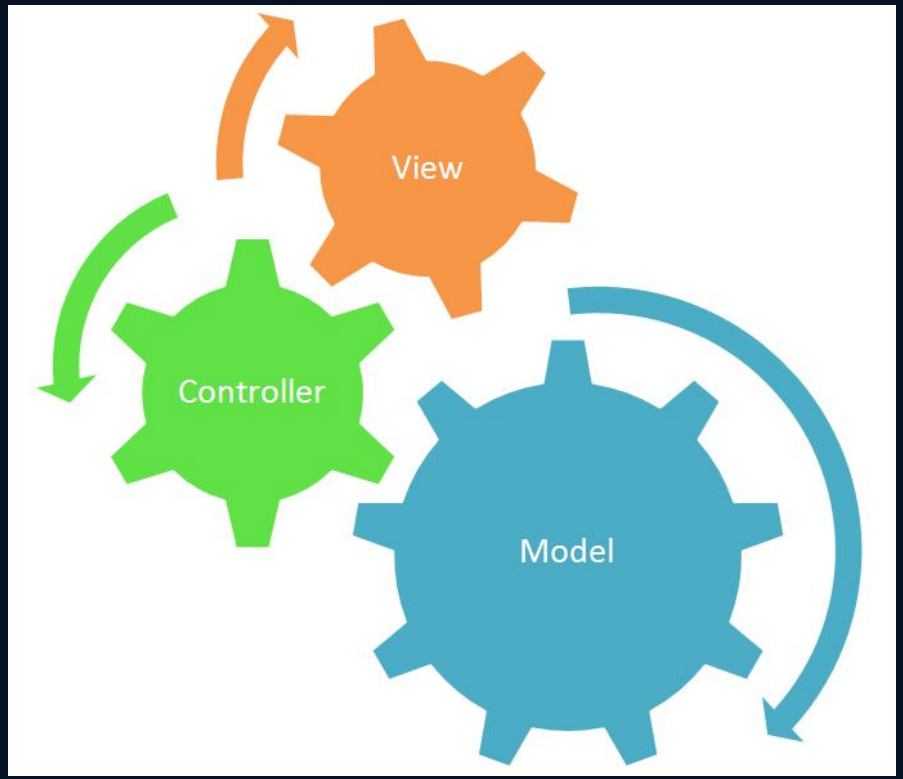
- Maybe - Bust a few Myths about ColdBox
  - But CFWheels and FW/1 are solid frameworks

Cat Meme for Raymond Camden ✓



What is MVC?

MVC is Model, View, Controller



What is MVC?

- MVC is a popular design pattern called Model View Controller which seeks to promote good maintainable software design by separating your code into 3 main tiers:

  - **Model** - Business Logic, Data, Queries, Etc
  - **View** - Representation of your models, queries, data.
  - **Controller** - Orchestrator of client request to the appropriate models and views

History of MVC

Trygve Reenskaug created MVC while working on Smalltalk-79 as a visiting scientist at the Xerox Palo Alto Research Center (PARC) in the late 1970s

History of MVC

In 1988, an article in The Journal of Object Technology (JOT) by two ex-PARC employees presented MVC as a general "programming paradigm and methodology" for Smalltalk-80 developers.

History of MVC

The use of the MVC pattern in web applications grew after the introduction of NeXT's WebObjects in 1996, which was originally written in Objective-C (that borrowed heavily from Smalltalk) and helped enforce MVC principles.

History of MVC

Later, the MVC pattern became popular with Java developers when WebObjects was ported to Java.

Later frameworks for Java, such as Spring (released in October 2002), continued the strong bond between Java and MVC.

History of MVC

In 2003, Martin Fowler published Patterns of Enterprise Application Architecture, which presented MVC as a pattern where an "input controller" receives a request, sends the appropriate messages to a model object, takes a response from the model object, and passes the response to the appropriate view for display.

Model

- The Model is the heart of your application.
- It is responsible for storing and retrieving all of your applications data and the objects that store that data.
- Your business logic should mostly live here in the form of:
  - services,
  - beans,
  - entities
  - and DAOs.

Views

- The Views are what the users see and interact with.
- This handles displaying the actual output to the user.
- It also handles communication with the user and the controller, via inputs and requests etc.
- Views are typically cfm/HTML, but it can also be
  - JSON,
  - XML,
  - data views, etc.

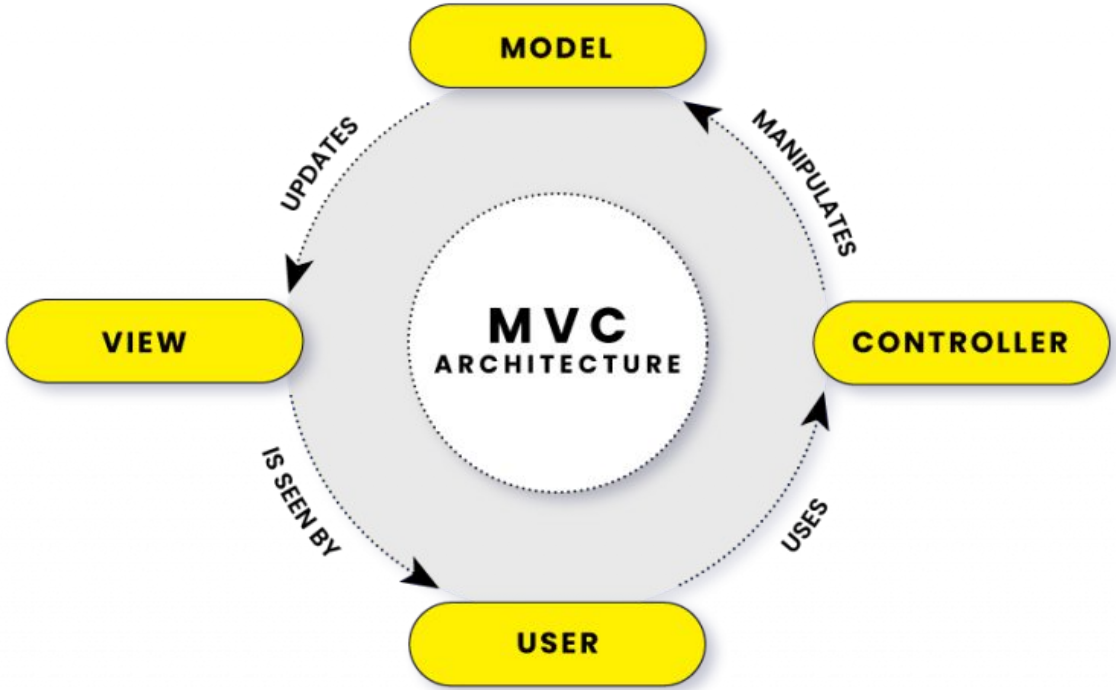
Views - Javascript Frameworks

- In modern times, your views can even be pure HTML with a combination of a JavaScript MVC framework.
- The major players in the MVC front-end world that we would recommend in order of personal preference:
  - AlpineJS - https://alpinejs.dev/
  - VueJS - https://vuejs.org/
  - Angular - https://angular.io/
  - ReactJS - https://reactjs.org/
  - EmberJS - https://www.emberjs.com/

Controllers

- Controllers are request handlers - Brain of your App
- Controllers are traffic cops of your application.
- They direct flow control, and interface directly without incoming parameters from FORM and URL scopes.
- It is the controller's job to take incoming request data, communicate with the appropriate models for processing, and set up either a view to display results or return serialized data like JSON, XML, PDF, etc.

What is MVC?



So Why use MVC?

Separation of Concerns

- The model and the view layers have different concerns about their implementations.
- A view layer is concerned with
  - how to render the data,
  - the type of browser,
  - or remote rendering, etc.
- The View also gives the user
  - Forms and inputs to interact with the system

So Why use MVC?

Separation of Concerns

- While the model is more concerned with
  - the business rules of the application,
  - how to store data
  - and even database operations.
- You use different development approaches to each layer.
- This makes organizing and maintaining small and large web applications easier.

So Why use MVC?
High Cohesion - Low Coupling

- **High Cohesion**
  MVC enables logical grouping of related actions on a controller together. The views for a specific model are also grouped together.
- **Low Coupling**
  The main goal by design with MVC is low coupling of display (views) from the data storage (models) with the Controllers handling requests.
  Dependency Injection (IOC) lowers Coupling more

So Why use MVC?
Faster Development Process

- MVC model allows one developer to work on a particular section (say, the view) while another can work on any other section (say, the controller) simultaneously.
  Example: View can use mock/fake data while models are built
- This allows for easy implementation of business logic as well as helps to accelerate the development process fourfold.
- It has been observed that when compared to other development models, the MVC model ends up showing higher development speeds (up to three times).

So Why use MVC?
Easier planning and maintenance

- The MVC paradigm is helpful during the initial planning phase of the application because it gives the developer an outline of how to arrange their ideas into actual code. It is also a great tool to help limit code duplication, and allow easy maintenance of the application.

So Why use MVC?
Multiple GUIs or Consumers

- Due to this separation, you can easily create multiple views for the same model data without affecting how the model works or is coded.
- The view layers can adapt to the model by coding their own implementations.
- This makes it really easy to create multiple GUI's for applications.

Example: One app, multiple consumers:
CFML Admin, API, Javascript FrontEnd, Mobile App

So Why use MVC?

Unit and Behavioral Testing

- Non-visual objects are easier to test than visual objects, in theory.
- With the introduction of Selenium, integration and visual UI testing has become easier.
- However, the key benefit here is that testing can be done separately.
- Modern MVC Frameworks even give you the ability to do UI and integration testing within its domain.

So Why use MVC?
Multiple GUIs or Consumers

- The most important benefit that we can arise out of the MVC pattern, is the direction of the dependencies.
- A view depends on its model data and controller,
- but the model itself does not depend on the view or controllers.
- This is how you want to build your business logic, encapsulated and providing a good API.

So Why use MVC?

SEO / User Friendly URLs

- MVC provides an easy way out to develop SEO-friendly RESTful URLs.
- Routing is a key feature of Modern MVC Frameworks
- You can easily change file structure without affecting your Routes / URLs

  Really important for Search Engines

- Obscuring your File System from the User is a Pro for Security too

Cons of using MVC?

Navigation of Code

- Navigation of Code can be more complicated
  - Traditional code would all be in one file or a shared cfc

- Solutions or Counterpoint
  - MVCs with solid conventions are easier to follow

    Note: You can usually override conventions but one of the strengths of MVCs are solid conventions, so keeping to them helps you save decision making for important decisions.
  - Modern IDEs have better file search and file management which can reduce this friction.

Cons of using MVC?
Single Change - Multiple Locations

- A single change might force changes in many locations
  - Models,
  - controllers,
  - possibly a view in each consumer your app is supporting

This complication is small in comparison to the gains MVC gives you, especially when supporting multiple consumers. Good abstraction allows your to maintain consistency and reduce brittle coupling.

Step by Step Conversion

Step 1 - MVC Framework

- Install your MVC Framework
- Configure your App with your Framework
  - Do you want it to take over or work alongside your existing code?

Step by Step Conversion

Step 2 - MVC Conventions

- Create your folders
  - Controllers
  - Models
  - Views
- Additional requirements
  - Layouts

Step by Step Conversion

Step 3 - Move Existing CFMs into Views

- Copy your Views into the Views Folder
- Re-create an Index.cfm in root
- Ensure your default file shows up

Step by Step Conversion

Step 4 - Test it out

- Test out your code
- At this point, your app can usually run - in just a V mode
- Soon we'll add the Controllers and then the Models last

Step by Step Conversion

Step 5 - Routes

- How do routes work?
- What if we need something special?

Step by Step Conversion

Step 6 - Controllers

- Let's make our Main.cfc Controller
- Let's add an Action
- How Controllers pass data
- Pulling input output & business logic into the Controller

Step by Step Conversion

Step 6b - Controllers - How did it work?

- How come the MVC works without Controllers?
  - ColdBox (and FW/1) are Convention over Configuration so it works on conventions
  - But you can override them
  - Example: Controllers - ColdBox calls them handlers, but we called them Controllers here for simplicity.

Step by Step Conversion

Step 7 - Models

- Models can hold your Business Logic, shared between controllers
- They are Framework unaware
- Let’s extract our Queries

- Dependency Injection - wait what?

Step by Step Conversion

Step 8 - Controller Power

- Controllers are traffic cops
  - Security
  - Relocation
- Power of Controllers
  - PreHandler
- Why use views when there is nothing to display?
  - Saving new rants
-

Step by Step Conversion

Step 9 - Routing Power

- Use routes to make your URLs even more readable
- Use routes to make your actions consistent
- Idempotency
  - How to say it
  - What does it mean?
- Building Links vs hard coding them

Step by Step Conversion

Step 10 - Cruddy by Design

- Simplify Controllers
- Names and Verbs
- Routes allow you to separate urls from site structure

Cruddy by Design

Adam Watham (TailWindCSS) at LaraCon 2017

https://www.youtube.com/watch?v=MF0jFKvS4SI

Step by Step Conversion
Step X+ - but Wait, they're more!

- Hierarchical MVC with ColdBox Modules
  - Uncle Bob's Clean Architecture talks about Horizontal vs Vertical Separation and Modules the ways to achieve that.
- Pluggability - ForgeBox.io
- Extendability
- Customizable
- Security

Conversion Guide Blog Posts

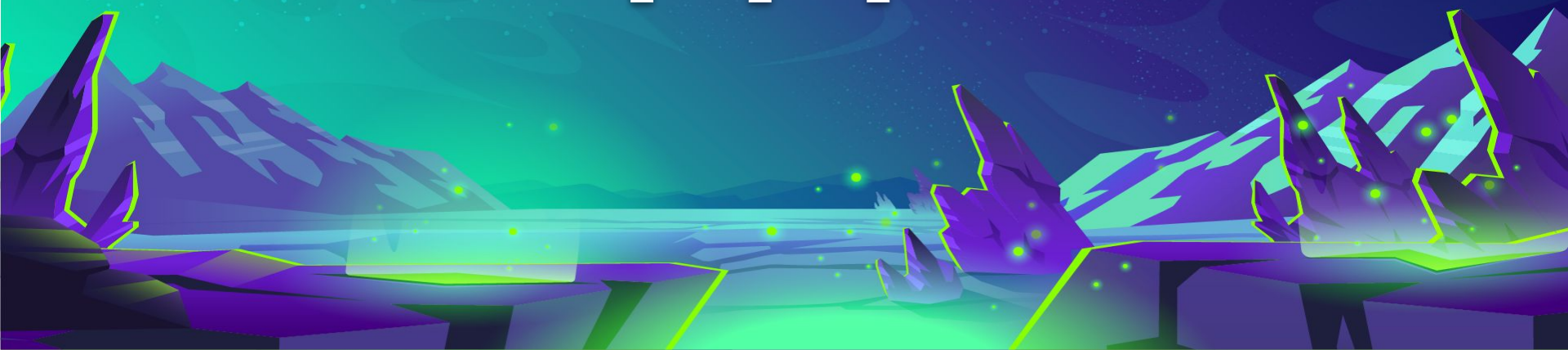
- Converting Fusebox 3 Sample App to ColdBox MVC
- Converting Fusebox 5.1 to ColdBox MVC (part 1 & part 2)
- Integrating ColdBox with Existing Code Series 5: Using Wirebox (Part 1, 2, 3, 4, & 5)

ColdBox Legacy App Demo
https://github.com/bdw429s/coldbox-legacy-app-demo

Ortus Solutions
A Software Revolution

Questions

???



Thanks

Ortus Solutions
A Software Revolution

https://github.com/gpickin/2022-cf-summit-mvc

- https://www.ortussolutions.com/
- https://intothebox.org/
- https://cfcasts.com/
- https://www.youtube.com/ortussolutions
- https://cfmlnews.modernizeordie.io/