# **Procedural to OOP in PHP**

Presented by
Barrett Avery, Data Systems Lead @ IDX, LLC
Antonio Ortega, Biz Dev @ IDX, LLC

# **Welcome Everybody**



## What is IDX?



We are a SaaS product that creates web tools based on aggregate real estate data.

And More Data Provided by a

And More Data Provided by a Multiple Listing Se Vice:

Listing Service Data Multiple Listing Se Vice:

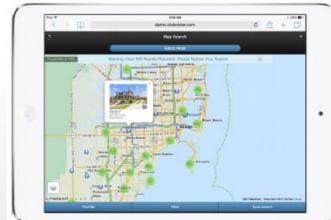
Listing Service Data Provided by a Multiple Listing Service:

And More Data Provided by a Multiple Listing Service:

And More Data Provided by a Multiple Listing Service:

And More Data Provided by a Multiple Listing Service: Multiple Listing Service: cxml><stuffs>asihe234oua wyd7suh<|stuffs><|xml>





## **Objective**

- Where we were and where we are going.
- Some differences between Procedural and OOP
- Give some insights from what we have learned along the way.
- Quick dive into SOLID Principles
- Quick dive into the Laravel Framework

#### Procedural - Where we were

**Definition**: Procedural Programming is a list or set of instructions telling a computer what to do step by step and how to perform from the first code to the second code. Source: Wikipedia.

#### **Procedural Pros and Cons**

#### Pros:

- Quick to develop and implement.
- Easy to learn.
- Simple architecture and overall structure.
- Good for quick and simple applications.

#### Cons:

- Difficult to scale for future needs.
- Does not lend itself well to test driven development (TDD).
- Usually is very flat in terms of design and structure.
- Not good for larger applications that will likely change over time.
- Maintaining can be very challenging.

#### OOP - Where we are

**Definition**: Object Oriented Programming (OOP) is a programming language model organized around objects rather than "actions" and data rather than logic. Source: Wikipedia.

#### **OOP Pros and Cons**

#### Pros:

- Much easier to scale for future needs and development.
- Good for larger more complex applications.
- Lends itself well to Test Driven Development (TDD).
- More dynamic and fluid in terms of the architecture and overall design.
- Maintainable.

#### Cons:

- Can easily become very complicated in terms of design and architecture.
- Takes much longer to develop initially.
- More difficult to learn than Procedural.

# Why did we switch to OOP?

- Scalability
- Testable
- Maintainable
- Follows a more modern methodology of programming (SOLID)
- Laravel

## **SOLID** and Laravel - Quick Dive

Covering a couple of key components we are using in our applications at IDX, SOLID principles and the Laravel framework.

# **SOLID OOP Principles**

Single Responsibility

Open Closed

Liskov substitution

Interface segregation

**D**ependency inversion

### **Laravel Framework**

- Why we chose a framework?
  - Gives a base to build scalable applications
  - Consistency between projects
  - Training is much easier when you have a standard to base your application on
- Why we chose Laravel?
  - Open Source
  - Fully featured
  - Lends itself to TDD
  - Provides built in tools for utilizing SOLID principles
  - Good for front facing as well as backend applications.
  - Good documentation and the framework is being actively developed.

## **Resources and Links**

Wikipedia

Laravel.com

Laracasts.com

PHP.net

Stackoverflow

# Thank you and happy coding!

