Which PHP Development Approach Should IBM i Programmers Use?

BCD
A Quadrant Software Company

QUADRANT SOFTWARE

The Global *PHP* Conference
zendcon 2015
October 19-22, 2015 • Las Vegas, NV

Stephanie Rabbani – BCD Professional Services

- I've been doing web application development on the IBMi for 12 years, 7 of those have been PHP
- ZF2 certified architect

- Contributed to the PHP Toolkit
- I have given PHP training courses
- 4th Zendcon, 1st time speaker

Agenda

- **Why PHP?**
- **PHP compared to RPG**
- **A look at procedural PHP**
- **An introduction to object oriented PHP**
- **A look at frameworks**
- **Which should you choose?**

PHP VS RPG

RPG	PHP
Procedural	Procedural/ OO
Modular – bound modules	Modular (via “includes”)
Compiled	Interpreted/Cached code (can be compiled on-the-fly)
Columnar coding style +free-format (or pseudo-free format)	100% Free format
IBM i only	Cross-platform
Business language-green screen, batch, web	Web, batch
Record Level Access/ DB2 SQL	SQL (MySQL, DB2, etc.)
*Popularity: < #51 (< 0.2 %)	*Popularity: #7 (2.94%)

* Source: www.tiobe.com/index.php/content/paperinfo/tpci/index.html, October 2014

Similar concepts

RPG	PHP
Programs	Scripts
Modules, Service programs	Includes and Objects
Subroutines and Subprocedures	Functions and Methods
Variable scope –Global, module, subprocedure	Global, function, object, object method
*INZSR	__construct (objects)
Embedded SQL	Same

Procedural Programming

- **It's what we've been doing in RPG for 35+ years**
- **All fields used to be global until procedures were introduced in 1994**
- **Allowed for parameters to be passed in and return values to be passed back**
- **With procedures we got modules, service programs and the ILE environment**
- **It allowed us to have more modular code, and avoid code duplication**
- **But it's still procedural… no instantiation, no inheritance, etc.**

Procedural PHP

- **Uses mainline code**
- **Can share code through the use of the include() function**
- **Has functions with local variables, parameters and return values**

```
<?php
$i = 0;
addOne($i);
echo $i; //1 appears on the screen

function addOne($number)
{
$number = $number + 1;
return $number;
}
?>
```

Pros & Cons of Procedural Code

- **PROS:**
  - Easy to start coding
  - Similar concepts to RPG
- **CONS:**
  - Does not promote modularity
  - Can grow to be unmanageable
- **Advice for procedural code**
  - Create function includes
  - Try to avoid global variables (pass variables as function parameters instead)

Intro to Object Oriented PHP

- **Objects conceptually mimic real world objects**
- **They have properties and can do actions**
- **Example (real world)**
  - An Appointment
    - Properties: start time, duration, a doctor and a patient
    - Methods: is appointment time available, register for appointment, add a note to the appointment

```
Class Appointment
{
    public $startTime, $duration, $doctor, $patient;
    public function startAppointment()
    {
        $this->startTime = date("His");
    }
}
```

Class vs. Instance

- **A class is an abstract definition of an object**
- **You create instances of that class**

- **Appointment class:**

```
$firstAppt = new Appointment();
$firstAppt->doctor = 'Jenken';
$secondAppt = new Appointment();
$secondAppt->doctor = 'Eudora';
$firstAppt->startAppointment();

echo $firstAppt->startTime;
```

Scope

- **Scope defines availability of properties/methods within and outside of an object**
  - Private - only the object itself can access a property or method
  - Public – other objects can access the property or method
  - Protected – applies to inherited classes
  - Method – scope local to method
- **Scope provides safety and clarity**
  - Helps prevent name collisions and unintended overwriting of memory
  - Names are prefixed with object name
    - $appointment->startTime

Starting Object Oriented Programming

- **Start by creating a simple object, think of it as an RPG module**
- **A customer object or a user object – I think almost everyone here will have customers**
- **You can use it in your procedural programs**
- **Move on from there (orders?)**

PROS of Object Oriented Programming

- **Promotes Good Code Organization**
  - Objects reflect their real-world counterparts
    - Order, Customer, Invoice, Inventory Item, etc.
- **Easier to maintain**
  - Structures and logic for objects located in one place
- **Safer**
  - Explicitly expose or hide data and logic (public versus private variables and methods)
- **DRY Principle (Don't Repeat Yourself)**
  - Cut down on repetitive code
  - Allows for more powerful coding patterns and abstractions
    - Patterns of Enterprise Application Architecture

Cons of Object Oriented Programming

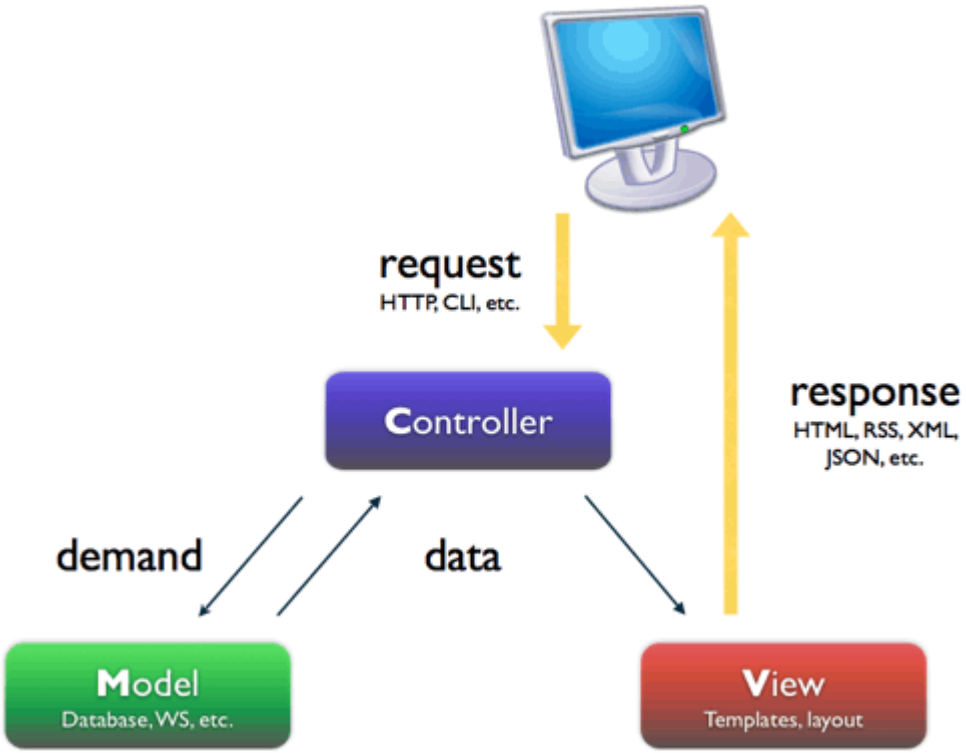
- **Longer learning curve than Procedural**
- **Requires you to think differently**
- **Requires more initial effort building infrastructure**
  - Consequence of keeping to the DRY principle
  - This would hopefully save you time down the road
- **The Toolkit is OO**

Where frameworks come in

- **A frameworks is an organized structure of code that allows you to develop web applications**
  - Many classes already designed to help you with many different aspects of your application
  - Database access, validation, caching, rendering your page, etc. etc.!
- **All Frameworks are Object-Oriented and most use a form of MVC**
- **Difference between 'Framework' and 'Code Library'**
  - Framework surrounds your code
  - A code library gets referenced by your code

Coding in a framework

- Semantics: Procedural and OO are language models, and MVC (along with ADR) is an architecture pattern.
- Frameworks can use either MVC (model/view/controller) or ADR (action/domain/responder) architecture patters.



MVC (ADR)

- **Let's start with 'C':**
  - Controller (can also be Action in ADR)
  - Manages conversation between User (via the View) and the code (the Model)
  - Routes user requests and provides responses to the user
- **Model = Database + Business rules (Domain in ADR)**
  - Eg: A 'Customer' could be a Model
  - Program calls would be included here
  - Typical RPG programs integrate both the model and the controller
- **View = User Interface (can also be Responder in ADR)**
  - It's the HTML incorporated with the response from the controller
  - Accepts input from user (links, forms, AJAX calls, etc).
  - This would be like our display files

Which framework?

- **Zend Framework 2**
  - The only framework I know of with native IBM DB2 support
- **Laravel**
- **Symphony**
- **Code Igniter, YII, CakePHP**

- **You can usually use pieces of these in your application**

The PROS of a Framework

- **Improves speed of development**
- **Have dealt with common problems so you don't have to:**
  - User-based security – Login, etc.
  - 'Clean' URL's
  - Sanitize input and output
  - Graceful error handling
  - Emails
  - Date manipulation and arithmetic
  - Consistent and safe database interaction
  - Template engines for Views (UI)
  - Etc.

The PROS cont.

- Separation of Concerns (SOC)
  - Model deals with database, business rules
  - View deals with UI
  - Controller deals with program flow and user interaction with code
- Easy to change pieces without impacting rest of app – eg: you can replace Views with a new design
- Great for large projects

The Cons of a framework

- Steeper learning curve
  - Need to learn the ‘language’ of the Framework
- Coding can be more verbose
  - Long variable names, or object and method names
- Documentation and Tutorials may be lacking
- Possible performance issues (ZF1 used to load many classes in the background, ZF2 took care of that)
- Dependence on Code Quality of Framework

Some ZF2 code:

- **IndexController.php**

```
namespace Application\Controller;
class IndexController extends AbstractActionController{
   public function indexAction()
   {
            $serviceLocator = $this->getServiceLocator();
            $homePageTable = $serviceLocator->get('advertising-header-table');
            $info = $homePageTable->getData($pageType);
            $model = new ViewModel();
             return $model->setVariables(array( ‘info' => $info, ));
    }
}
```

Any questions?

- **Come see me tomorrow for introduction to ZendFramework 2 on the IBM i**

- **Contact me by email: stephanie@excelsystems.com**
- **Twitter: @jordiwes**

- **Please provide feedback for this talk: https://joind.in/15526**
- **Thank you!**

- **Bonus Toolkit material**

Toolkit calls

1. **Include Toolkit file:**

```
require('ToolkitService.php');
```

2. **Create instance:**

```
$tkObj = ToolkitService::getInstance($conn,$namingMode);

$tkobj->setOptions(array('stateless'=>true));
```

3. **Array of params:**

```
$params[] = $this->tkobj->AddParameterChar($type, $parmlength, $rtnField,
$rtnField, $parmValue, $varying);

$params[] = $this->tkobj-
>AddParameterPackDec($type,$length,$dec,$rtnField, $rtnField, $parmValue);
```

4. **Call the program:**

```
$result = $this->tkobj->PgmCall($program, $lib,$param, null, null);
```

5. **Check out the return values**

```
$returnArray = $result['io_param'];
```