

Readme:

PHP Code sample of Monir

By

M Moniruzzaman

For

Rosen Ivanov

2014-12-14

Contents

README:	1
PHP CODE SAMPLE OF MONIR	1
CONTENTS	2
CONTEXT	3
SCOPE	3
LIMITATIONS	3
KNOWN ISSUES	4
TOOLS AND METHODOLOGIES	4
WHAT THE SAMPLE CODE IS ABOUT	5
COMPONENTS DEVELOPED IN THE SAMPLE	5
THE BASIC FLOW	7
DIRECTORY STRUCTURE	7
NAVIGATING	9
QUESTIONS OR ISSUES	10

Context

As a part of the interview process, Rosen Ivanov asked Monir to send a sample of his code. Hesitating over whether coding done for clients can be given as sample quickly without permission of client and companies under NDA agreements, Monir decided to code a small miniature framework that can be relevant to questions that Rosen was asking in the interview.

Scope

Rosen did not specify any scope or type of code, saying only that the code should be around 300 lines.

Monir thought that additionally he can make the code and the task relevant to some object design pattern questions that Rosen was asking in the interview. He remembered the following major questions and tried to use/implement these concepts in the miniature framework he coded to showcase his coding style.

- Encapsulation of functionality
- Abstract classes
- MVC design pattern
- Visibility and scopes: private, protected and public etc.
- Inheritance etcetera

Limitations

Monir faced following limitations in the coding –

1. Staying on the verge of Year End project deliveries and related sprints in his current workplace, he could not make time for it, except for the weekend (Saturday)
2. Not being sure what aspects or technologies would be judged in the coding, Monir chose to develop a technology-agnostic small miniature PHP-based pseudo-framework, which does not talk to operating systems and databases, but with objects on-memory.
3. Since the PHP coding skill was to be tested, and the position was *PHP Backend Developer*, Monir decided to not worry regarding UI/UX and interactivities much, rather he focused on the

application design questions that were discussed during the interview, since those aspects may be crucial for the job role.

Known Issues

This one day prototypical sample of coding style does not:

1. Reflect a full-fledged working solution
2. Scale
3. Have much of UI / UX design
4. Database / File System / Network I/O operations
5. Unit test coverage

Additionally, documentation was not greatly up to the mark, few errors exist with PHPDocumentor compilation, although the documentation markup does not reflect the errors. Not each and every line was commented, though sometimes overcommenting is considered as a code smell.

Tools and Methodologies

Monir used the following tools to design and develop the coding sample:

1. Netbeans 8 for PHP coding
2. StarUML for UML design
3. PHPDocumentor for the documentation, MS Word and PDF printer for this readme
4. Code is shared via email attachment
5. Monir started in TDD approach with PHPUnit, but later abandoned it altogether, thinking it may be unasked for.

Rather later he tried to follow minimalistic, zero upfront and rapid prototyping XP principles since he did not have time much for submission of the code, and he thought he was running late.

What the sample code is about

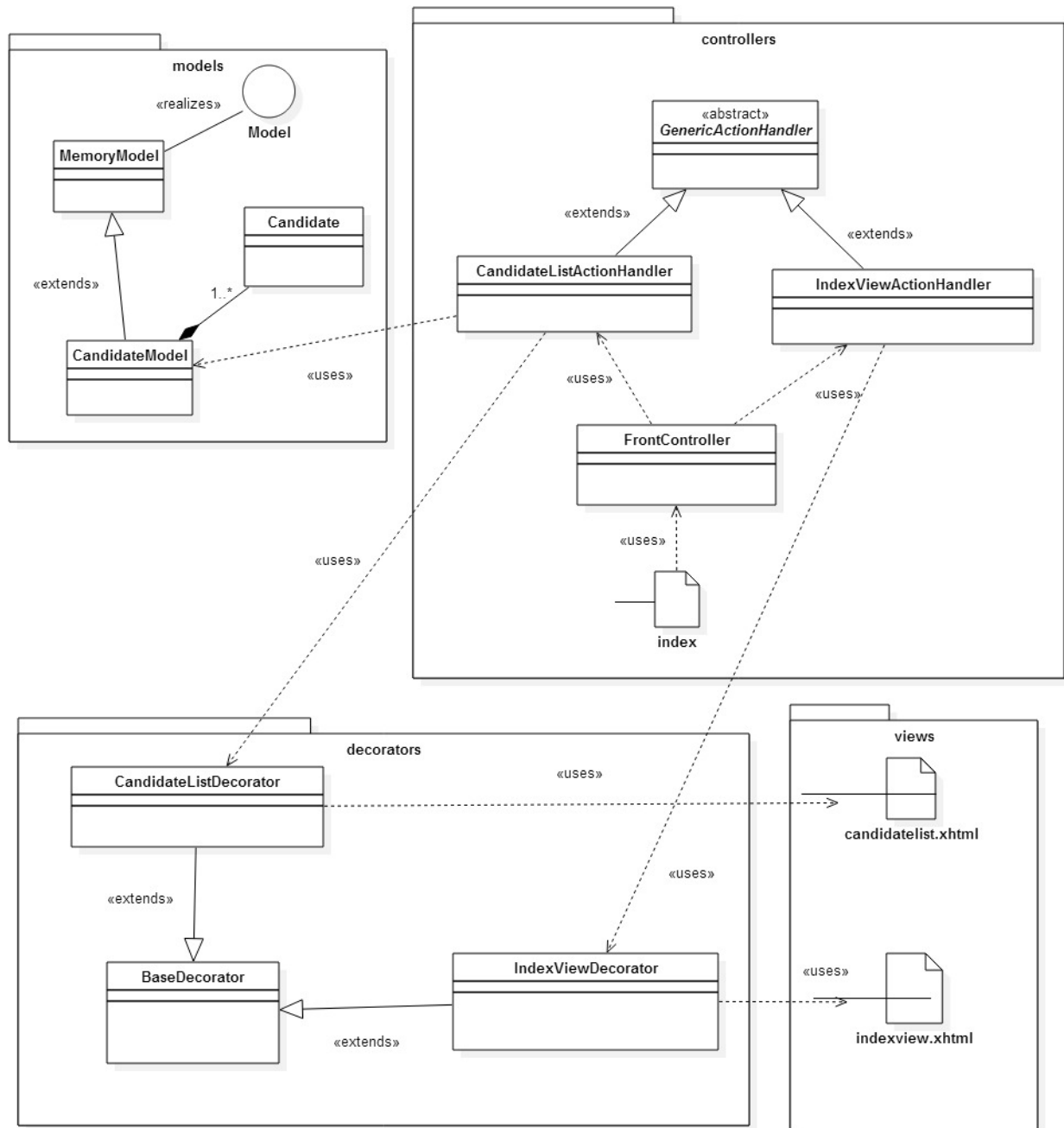
Rosen interviewed candidates, and from a hypothetical on-memory data source he is seeing the scores that candidates have got.

With a coding limit of 300 lines, Monir thought of focusing on ONLY this ONE use case – view list of scores of candidates, from a memory data source.

A simple index page welcomes the guest, the only role apart from the system, and gives him option to view the scores and also to view the documentation of the source code.

Components developed in the sample

The following components were developed in the sample:



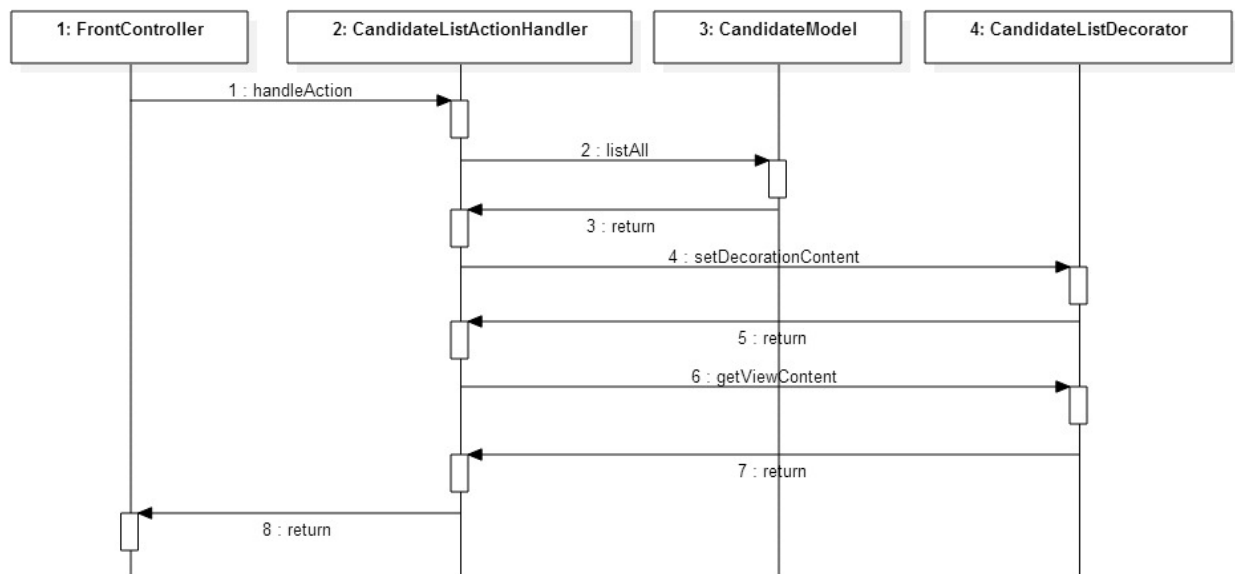
The components are layered into four packages. Not being sure about which environment it will be run or tested on, Monir decided not to use Namespaces. The packages are namely:

1. Models
2. Controllers
3. Decorators
4. Views

Additionally there were Exception classes and Documentation files.

The Basic Flow

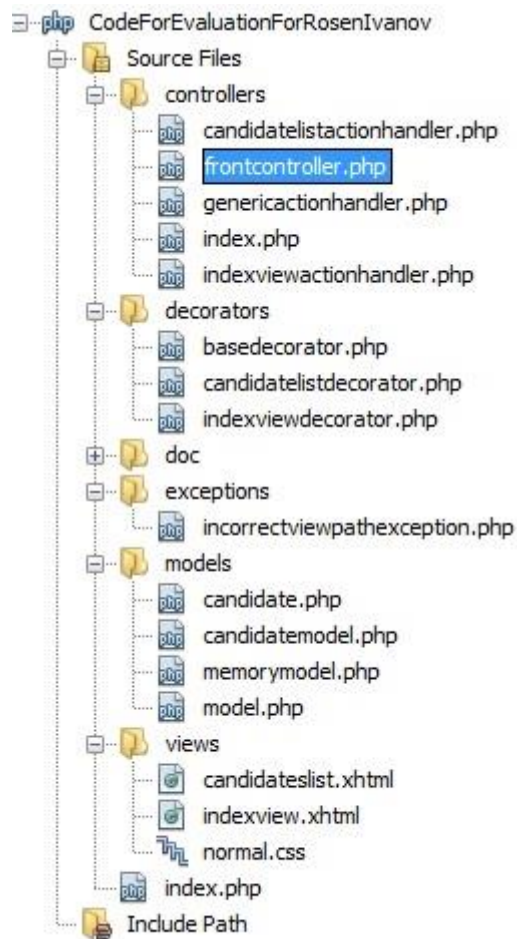
After the request reaches to Index.php, it is redirected to internal index in the controllers. The index transfers the request to FrontController, which then processes the request in a chain of responsibility, and returns that response to be echoed by the Index.



As we see in the above sequence diagram for the single hypothetical use case Candidate Listing, after the FrontController receives request, it finds the Action Handler responsible for dispatching this action. The CandidateListHandler knows its mode and its decorator. So it first fetches the data from the Model with the listAll call, and then it sends the data to be processed and decorated by the view. Once the handler receives its response from CandidateListDecorator, it returns it in the chain to FrontController.

Directory structure

The code have the following directory structure:



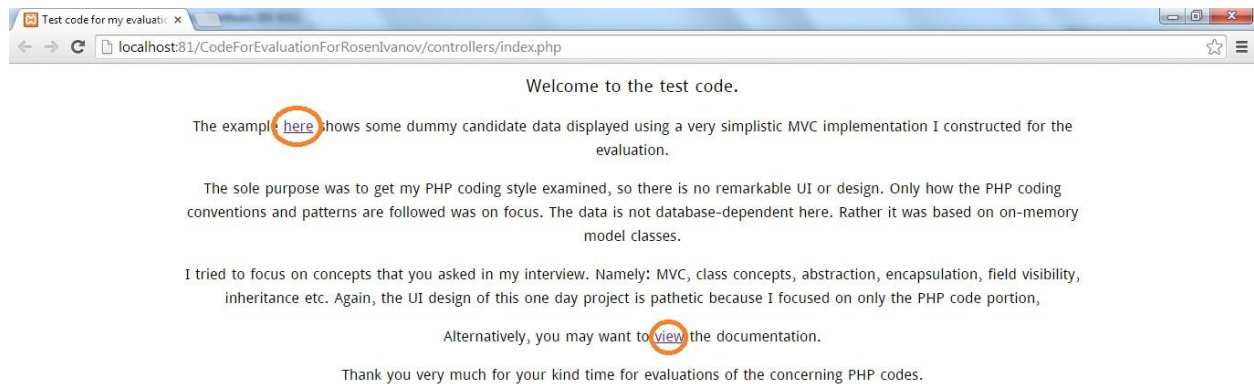
Root folder

- i. Index.php (root)
- ii. Readme.pdf (this file)
- iii. Controllers
 1. candidatelistactionhandler.php
 2. frontcontroller.php
 3. genericactionhandler.php
 4. index.php (internal)
 5. indexviewactionhandler.php
- iv. Decorators
 1. basedecorator.php

- 2. candidatelistdecorator.php
- v. Doc
- vi. Exceptions
 - 1. Incorrectviewpathexception.php
- vii. Models
 - 1. candidate.php
 - 2. candidatemodel.php
 - 3. memorymodel.php
 - 4. model.php
- viii. Views
 - 1. candidatelist.xhtml
 - 2. indexview.xhtml
 - 3. normal.css

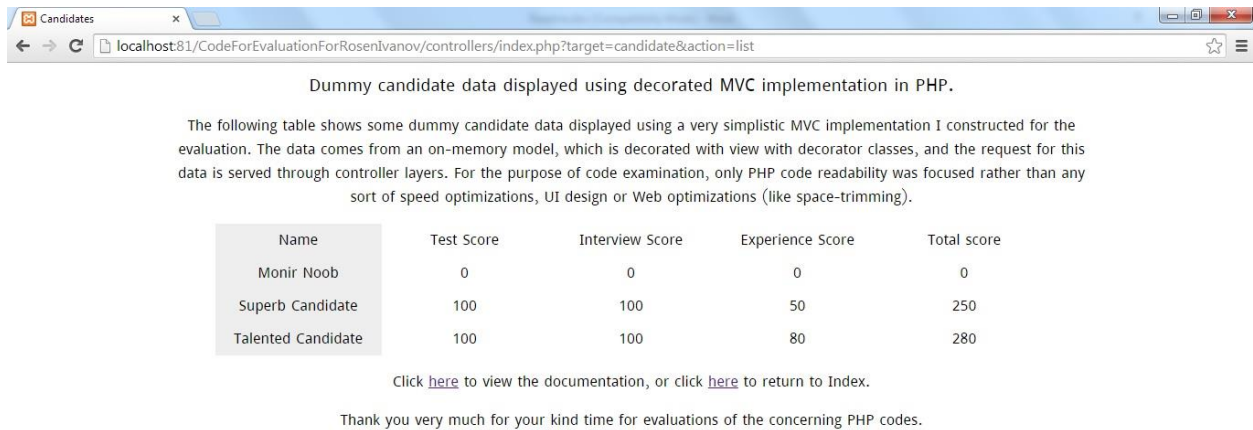
Navigating

When invoked, the index.php looks like the following:



You can go to see the sample use case of listing candidates from there, or you can view the documentations, as shown above.

The candidate listing page may look like the following:



Dummy candidate data displayed using decorated MVC implementation in PHP.

The following table shows some dummy candidate data displayed using a very simplistic MVC implementation I constructed for the evaluation. The data comes from an on-memory model, which is decorated with view with decorator classes, and the request for this data is served through controller layers. For the purpose of code examination, only PHP code readability was focused rather than any sort of speed optimizations, UI design or Web optimizations (like space-trimming).

Name	Test Score	Interview Score	Experience Score	Total score
Monir Noob	0	0	0	0
Superb Candidate	100	100	50	250
Talented Candidate	100	100	80	280

Click [here](#) to view the documentation, or click [here](#) to return to Index.

Thank you very much for your kind time for evaluations of the concerning PHP codes.

You can go to the documentation space from any of the pages.

Finally, the codes are written in Netbeans 8 on Windows, but can be viewed in any editor, and should run on environments running PHP 5.3 or higher.

Questions or issues

Please let know immediately: callmoni@gmail.com

Thank you