**PRAILS**

You can call it Php On Rails. Php Rails. PHPoR. But all those name are taken somehow. So I decided to come  
with a closer, but unused approach: PRAILS.

Prails is a Rails inspired PHP MVC Framework. Instead of bringing a heavyweight  
set of classes and assets, this framework is very simple but works just as any other one.  
You can start with your own predefined models or you can create your own from scratch. Support for MySQL and fixtures  
come out of the box. Support for other databases is coming in the near future. It comes with full support for Fat Controllers or Fat Models. You choose. Support for private and public pages is   
included, so you don't have to mix everything under one view directory. Templates support is  
included, so you can rely on a single page to do your UI.

Since it runs over PHP 5, the compatibility is only limited by PHP itself (In short terms, there is no limit). That also means it runs in Linux, Windows, Mac, etc.

It comes with its own test class to avoid the installation of additional test frameworks. The results  
are jUnit compatible if you want to integrate the project with automated builders. Just point your  
parser to tests/results and your done.

**STRONG DATA FRAMEWORK**

Prails come with a very simple and yet powerful data framework that lets you:

* Connect to MySQL.
* Inject other databases.
* Create visual fixtures in test mode. No database required.
* Edit fixtures on the fly for testing. Again, no database required.
* Database migrations.
* Code-First.
* Own Query Language

**HOW TO SEE IF MY PRAILS INSTALLATION IS WORKING**

Prails comes with its own test class to avoid the installation of additional test frameworks. The results  
are jUnit compatible if you want to integrate the project with automated builders. Just point your  
parser to tests/results and your done.

To see if your system is compatible and if Prails is working fine, just run the tests by calling them via your web browser:

http://localhost/application\_path/tests/

Don't forget to not to publish your tests when going to production.

**IS PRAILS PRODUCTION READY?**

Yes, there are applications in production already running in Prails.

**GETTING STARTED:**

The easiest way to get started with PRAILS is to write the basic “Hello World”, the “Prails” way.

Step 1: Create your model

In the directory “/app/models” lets create the file hello.model.php (The naming is optional):

class hello\_model{  
}

We will only need a basic variable and no methods.

class hello\_model{  
  var $name;  
}

Step 2: Create your controller

In the directory “/app/controllers” lets create the file hello.controller.php (The naming is optional):

class hello\_controller{

public function \_\_construct() {  
    $this->DynamicCall();   
  }  
}

All controllers must be initialized with this construction method in order to inherit the full variable capture of the framework.

Now lets create an action called “world” that sets a variable “response” and then calls the Prails RenderView method:

class hello\_controller{

public function \_\_construct() {  
  $this->DynamicCall();  
}

public function world() {  
  $this->response = “Hello world”;  
  return $this->RenderView();  
}

Step 3: Create your view

In the directory “/app/views” lets create the file world.php (The naming is strict, and matches the action if you expect Prails to render the action result):

<html>  
<body>  
<? print $this->response?>  
</body>  
</html>

Step 4: Run your application

If you have the application running in your server, go to the following URL:

http://localhost/application\_path/?hello/world

You must see:

Hello world

**PRAILS DIRECTORY STRUCTURE**

Prails directory structure is very similar to Rails, but comes with some extra ones to help distribute functionalities:

* app: Contains your application. Your Models, Views and Controllers are located here
* app/assets: contains your static assets like JavaScripts, images and CSS:
  + app/assets/javascripts
  + app/assets/css
  + app/assets/images
* app/controllers: contains your application controllers
* app/models: contains your application models.
* app/views: contains your application views. Is distributed in private and public. Private pages are only visible when the variable ”\_private” is set to true.
  + app/views/public
  + app/views/private
* config: contains your configuration files for all or the current environment
* db: contains your Database drivers and fixtures (For testing).
* doc: contains all Prails documentation.
* lib: contains all Prails core framework.
* tests: contains the VerySimpleTest Testing framework set for Prails.
* tests/results: contains the jUnit XML results from last run tests.
* tests/suite: contains the test suite (Files with the Tests).
* tests/suite/after: these are the tests that are run after all the tests files.
* tests/suite/always\_asfter: these tests are run after every test file.
* tests/suite/always\_before: these tests are run before every test file.
* tests/suite/before: these are the tests that are run before all the test files.
* tests/suite/tests: your actual tests.

TODO LIST:

a) Self-creating command: create the structure of the application with a PHP command.  
b) Model and Controller generation from existing databases.  
c) Scaffolding.  
e) Online documentation.