The best way to learn to use Prails is by practice. You will find that a lot of tasks are very simple and straightforward. That’s the idea of Prails.

To start, lets understand how Prails is designed. Prails itself is not the core class of the application, it actually seats in the middle of the Data and the Controllers. That’s because Prails relies on data (Not necessarily a database) to work (Like any other application). So the base class is the Context Class. The context class extends whatever is the data layer selected by the user. Can be MySQL, MSSQL, Oracle or none. So looking at the code you will see that the Context Class extends the db\_driver class (Yes, is named db, but it doesn’t need to be connected to a database). Prails then extends the Context class and enriches it with an entire MVC set of methods. These methods will eventually rely on a helper class called routes. Routes will convert the URI in commands that Prails can understand. On top of Prails goes the Model. The model will use (If required) the data layer to perform data manipulation. The data manipulation is contained on each DbDriver independently and it has easy to use commands to avoid using SQL Queries (But yes, you can use your own queries). On top of the Model goes the Controller. The controller inherits all the previous inheritances plus the data model in order to run the actions. One of the many capabilities (Flexibilities) of Prails is that you can avoid the Model and inherit directly from Prails. Or avoid the context and inherit directly from the db\_driver. It all depends on what you want to do. The reason to use the context is that it gives you tools for Testing that mocks data (You will see this in the Testing Chapter). If you avoid the Model, you can use Prails features without a model attached, but you may not be able to access the data so easily (Because the model gives the application an abstraction of your source of data). The way everything gets connected and executed is based on the URI Call. The Routes helper will decompose the URI in three parts: The Controller, The Action and The Query String. If you have a Uri like this:

http://your.domain/?users/edit/1

It means: Load the controller “Users”, execute the action “edit” on the record with ID “1” and display it in the view “Users”. Prails html rendering is very standard and easy to understand. By default the action name is the same view name (You can change this whenever you want), and you can have public and private pages (Private pages can be like registered users sections or administrator sections).

Lets try to do the most basic example step by step.

For this example we wont use a database, but still we will create a model to understand the entire MVC concept behind Prails.

Step 1: Create the Model.

Lets go to the directory “/app/models” and create the file “users.model.php” (You can name it in any form, we used this notation for easy understanding). Type the following code:

class users\_model extends prails{

var $name;

}

This models means that you have a “name” property that can hold data.

That’s the most basic model you can have. In further chapters we will see how to declare models that are attached to databases.

Lets go to the directory “/app/controllers” and create the file “users.controller.php” (You can name it in any form, we used this notation for easy understanding). Type the following code:

class users extends users\_model{

function ask()

{

$this->RenderView();

}

}

This controller means that you have an action called “ask” and only displays the view of the same name (ask).

Because we have a controller that renders a view, we need a view. Lets go to the directory “/app/views/public” and create a folder called “users”. In users create a file called “ask.php”. Here the naming is strict, because the controller looks for the views inside the folder of its name and the file with the same name of the action. Type the following code inside “ask.php”:

Hi there!

Now, type the following URL (Depending on where you have Prails running):

http://localhost/prails/?users/ask

You must be able to see

Hi there!

No big deal. Seems too much to just make the view show up. Well, lets make it better so we can understand how this works. Lets open the “ask” view we created and type this code instead:

<form method="post">

Please type your name: <input id="name" name="name" type="text"> <input type="submit" value="Send">

</form>

<? if($this->name != "") print "Hi ".$this->name?>

If you reload the page you will see the input field and the button. If you click it nothing happens. Make something happen.

Open the controller file and add this method:

function ask\_post()

{

$this->name .= ", it is a beautiful day!";

$this->RenderView();

}

Now reload the page, type something and submit the form. This time you see a text at the bottom formatted as you indicated in the controller.

What we did?

Prails understand that you may need to submit forms from a controller. To avoid creating actions for each submit (You can if you want), you can add the “\_post” action and the controller will load this action when is submitted. Because the controller inherits from the model (And the model from Prails), it sets the value of “name” with the value from the form. Prails matches the Post values with the properties of the model automatically. That’s why when you appended the “it is a beautiful day”, it gets appended to the already set value of the name.