

Titania

A Quick Preview on Using Titania, a Ruby Web Framework



Aditya Purwa

avaga

2013



Titania

Titania is a ruby web framework developed by Aditya “Avaga” Purwa of ION. I hope that Titania can bring a new future to web development using ruby. Titania is **not aimed to defeat today top framework** like Rails.

Titania is aimed so you can **learn development** using Ruby easily using the XMVC design pattern. XMVC design pattern was also developed by Aditya “Avaga” Purwa. **XMVC stands for Extended Model View Controller.**

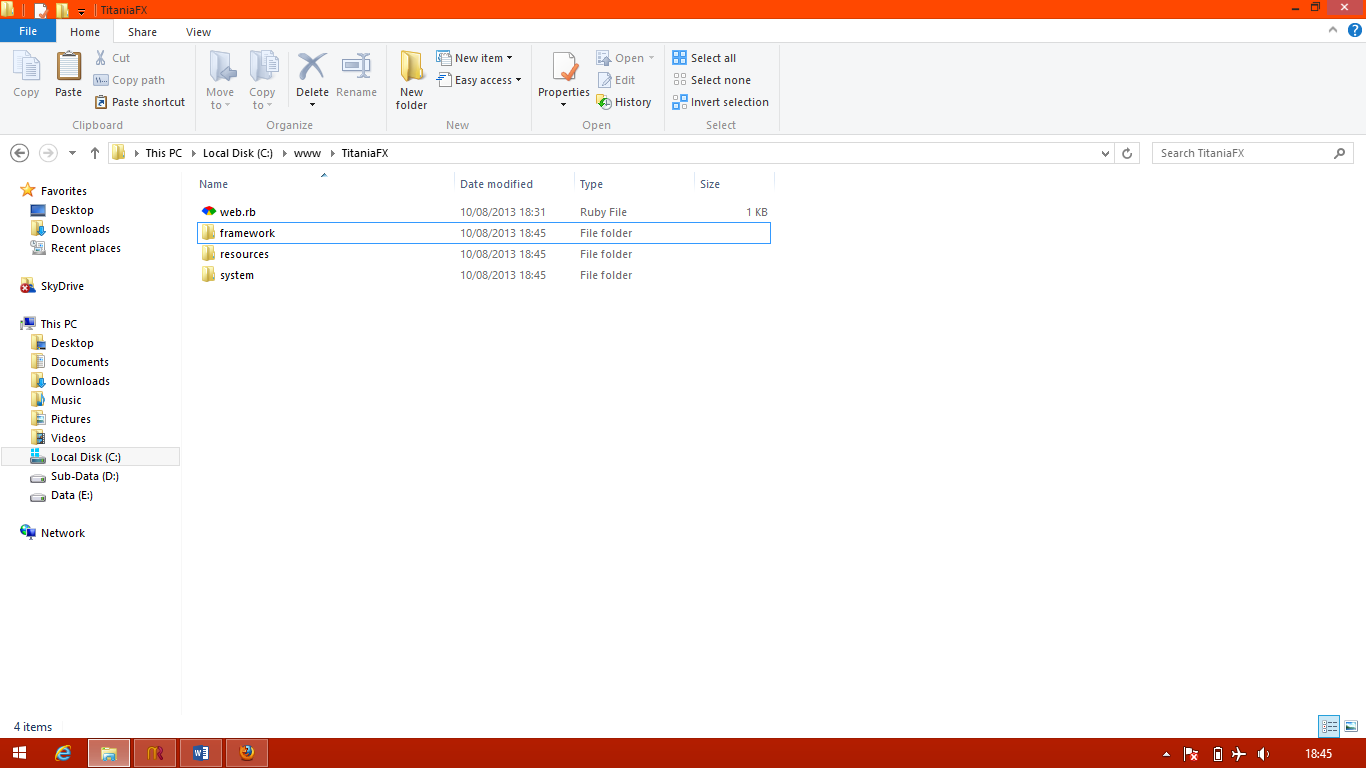
While MVC is so popular today, there is areason why the creator of Titania created its own design pattern to supply Titania.

We’ll be using the **preview version of Titania**. We will aim you to develop a web application using XMVC of Titania. The preview version doenst have native database support, doesnt have http request form data support also. All of these features will be available during final release.

# Installing Titania

You can get Titania on .... (download it as an archive).

Extract Titania archive anywhere on your computer. Use C:\www\TitaniaFX\ for this preview tutorial.

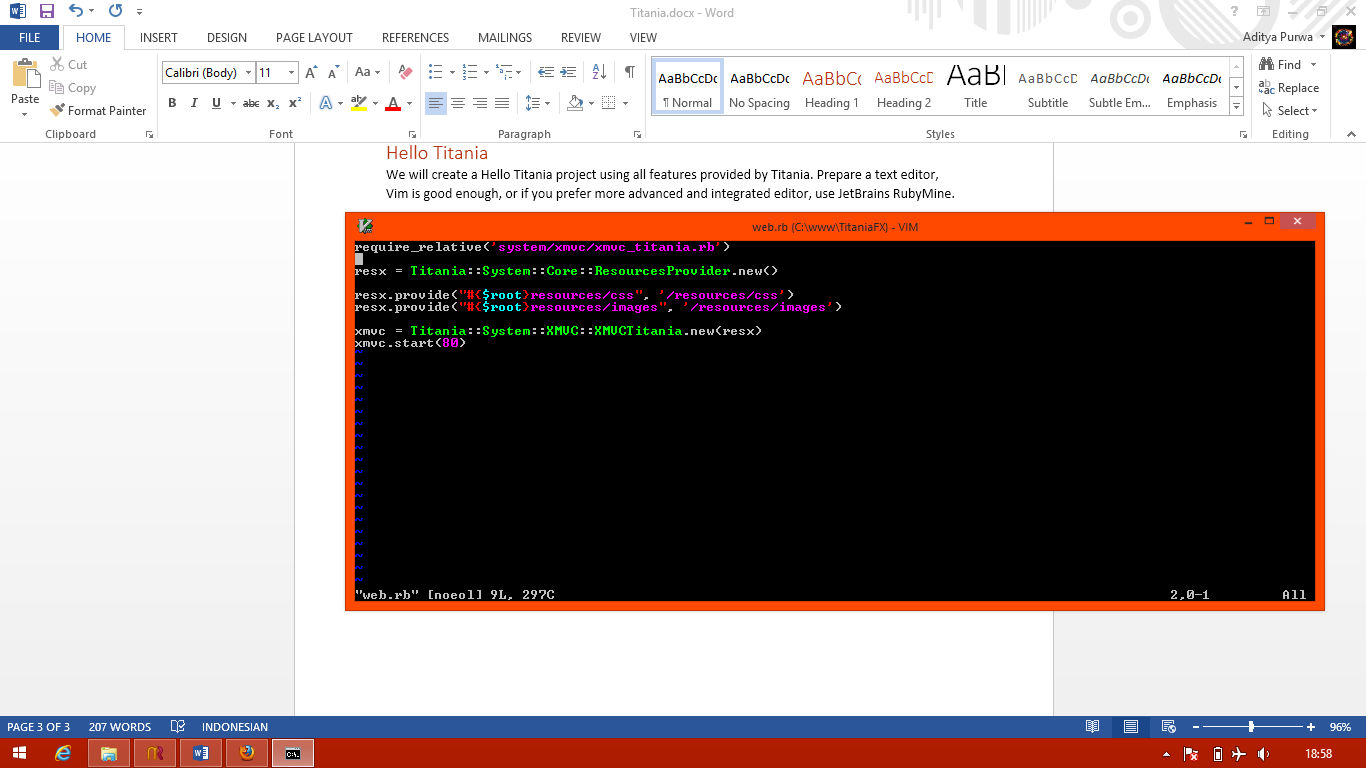


Well done, you have fully sucessfully installed,. Titania on your computer. There is **no messy configurations** to do. XMVC will works well automatically. To test your installation, make sure you didnt have port 80 used by other programs (Apache, Skype, etc). Then, run web.rb. Open your browser, type 127.0.0.1 or localhost at the address bar.

# Hello Titania

We will create a Hello Titania project using all features provided by Titania. Prepare a text editor, Vim is good enough for this tutorial, or if you prefer more advanced and integrated editor, use JetBrains RubyMine.

Open web.rb, you’ll see this:



The first line, is require\_relative. Which loads the core file needed to use the XMVC model for Titania.

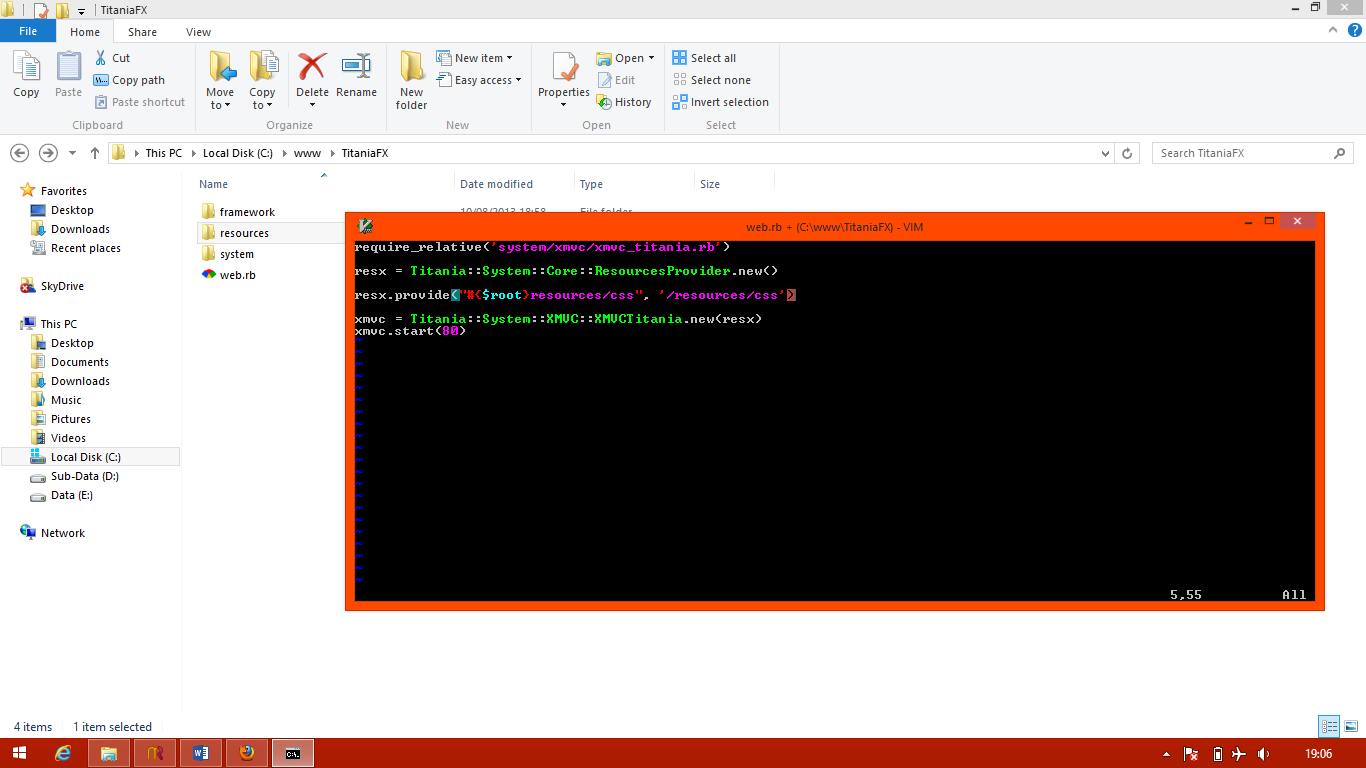
The next line is required to supply resources to our web application. Resources in Titania means a file that is publicy available to the client. No processing ocurred during transfering the resources to the client.

Resources may useful if you want to supply an image, css, or javascript file. Because you dont need to pre-process an image, css, or javascript file before transfer it to the client.

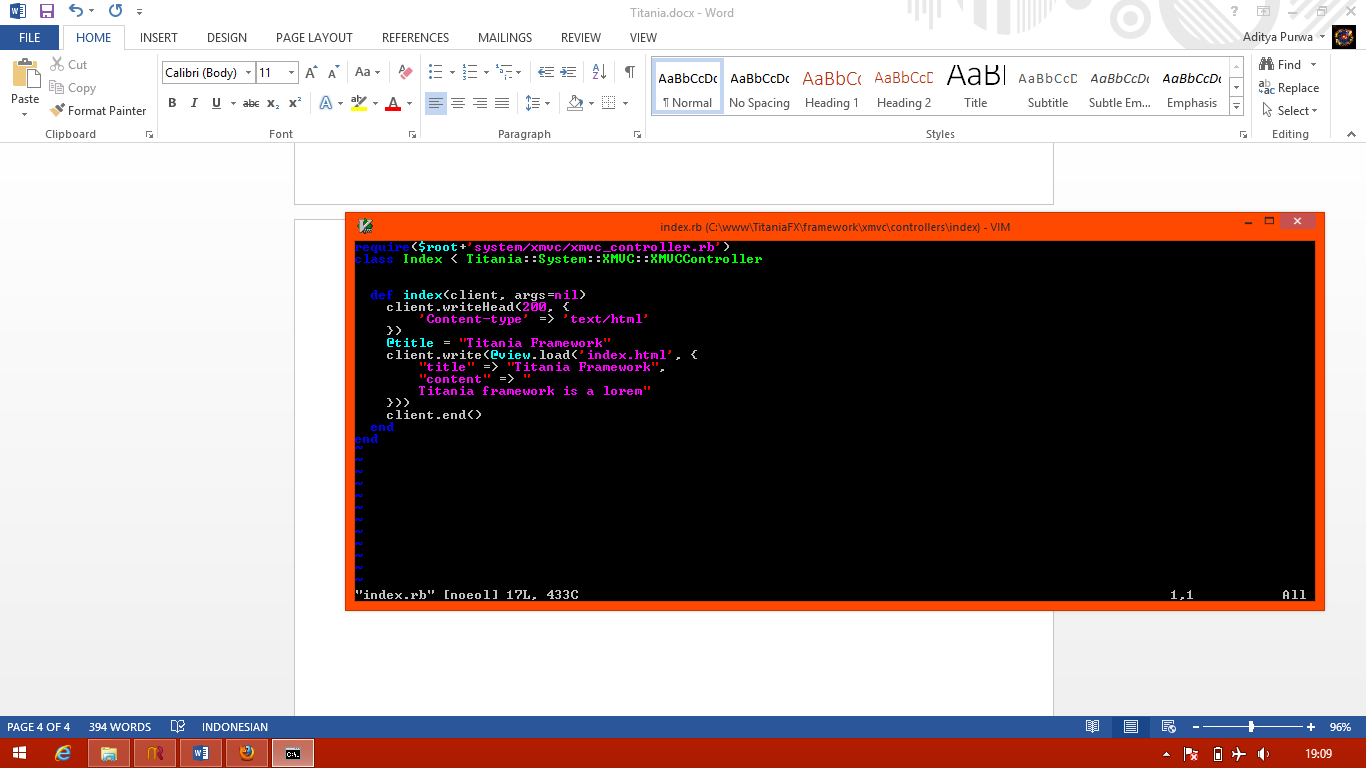
The next line is listing the resources to provide. We wont need the images folder by now, you can delete the line that provide an images to the client. Delete the **/resources/images** folder also.

The next line, is the most important. You create a new XMVC based server and binds it on port 80.

The web.rb file should be look like this after a few editing:



Your file for working with XMVC is stored on **/framework/xmvc**. Open **/framework/xmvc/controllers/index/index.rb**.

XMVCTitania already define its routing format. As an example, look at this url

<http://localhost/index/Index/hello/titania>

!!! TITANIA URL IS CASE SENSITIVE !!!

You wont be able to open <http://localhost/index/index/hello/Titania> (unless you name your class `index`, not `Index`)

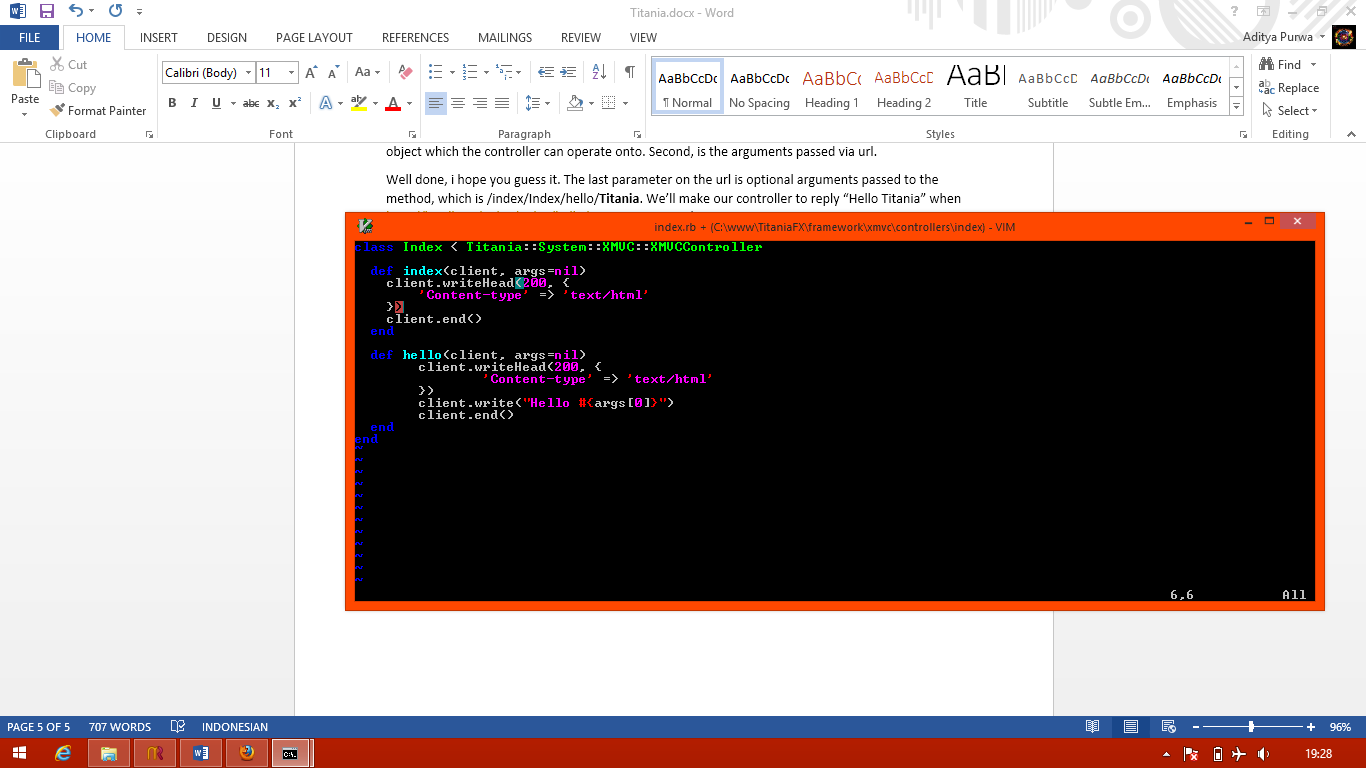
The <http://localhost/> just points to your computer, the first /**index** is the name of the package. You define the package name in the /xmvc/controllers folder. In this case, the index.rb file we opening resides in the `index` package. The default package accessed when no package is specified is the index package.

The second, /index/**Index** is the name of the controller. You define the controller inside your package folder. The name of the file must be all in lowercase, while the class name should follow standard convention, WordCase. The file name and the controller class name should be equal (except for the case, the case doesnt have to be equal). The default controller accessed when no controller is specified is the Index controller.

The third, /index/Index/**hello** is the name of the method called on the controller. Its considered a good practice to name your controller method using under\_line\_case. It will makes your url more prettier. The default method called when no method is specified is the index method.

Controller method should follow this signature, they must accept 2 parameters. The first is the client object which the controller can operate onto. Second, is the arguments passed via url.

Well done, i hope you guess it. The last parameter on the url is optional arguments passed to the method, which is /index/Index/hello/**Titania**. We’ll make our controller to reply “Hello Titania” when <http://localhost/index/Index/hello/Titania> is accessed.

Okay, lets examine this file!

The first line is class declaration. We declare a class named `**Index**` that extends from **XMVCController**. Your controller doesnt need to extends from the XMVCController. But, for integrity and consistency sake, you should extends it from the XMVCController.

Edit the `**index**` method so it doesnt show anything.

Define a new method called `**hello**`, accepting 2 arguments, the client and arguments. You do all output operation directly to the client object. The client object passed to this controller is an instance of **HttpClient**.

**HttpClient.writeHead**, is a method that accepts 3 arguments. The first is the http response code, 200 is a response code for OK, which means the server understand and able to fulfil the client request. The second is additional header to be passed to the client, we specify the content-type of the data we sent to be an html text. The third (optional, default to false) is used to specify wether we remove default header or not. You should call this method before any other output method.

The default header passed is **Transfer-Encoding** as `**chunked**`, and **Connection** as `**keep-alive**`.

**HttpClient.write**, is a method accepting one arguments. The first and only arguments is a string to be sent to the client.

**HttpClient.end**, is a method without arguments. Its end the client stream so the server can release the data to the client. You should call this method when finished outputting data to the client.

Edit **index.rb** so it looks like the image above. Open your browser, type [**http://localhost/index/Index/hello/Titania**](http://localhost/index/Index/hello/Titania)

As your URL, remember that Titania URL is case-sensitive.

# XMVC

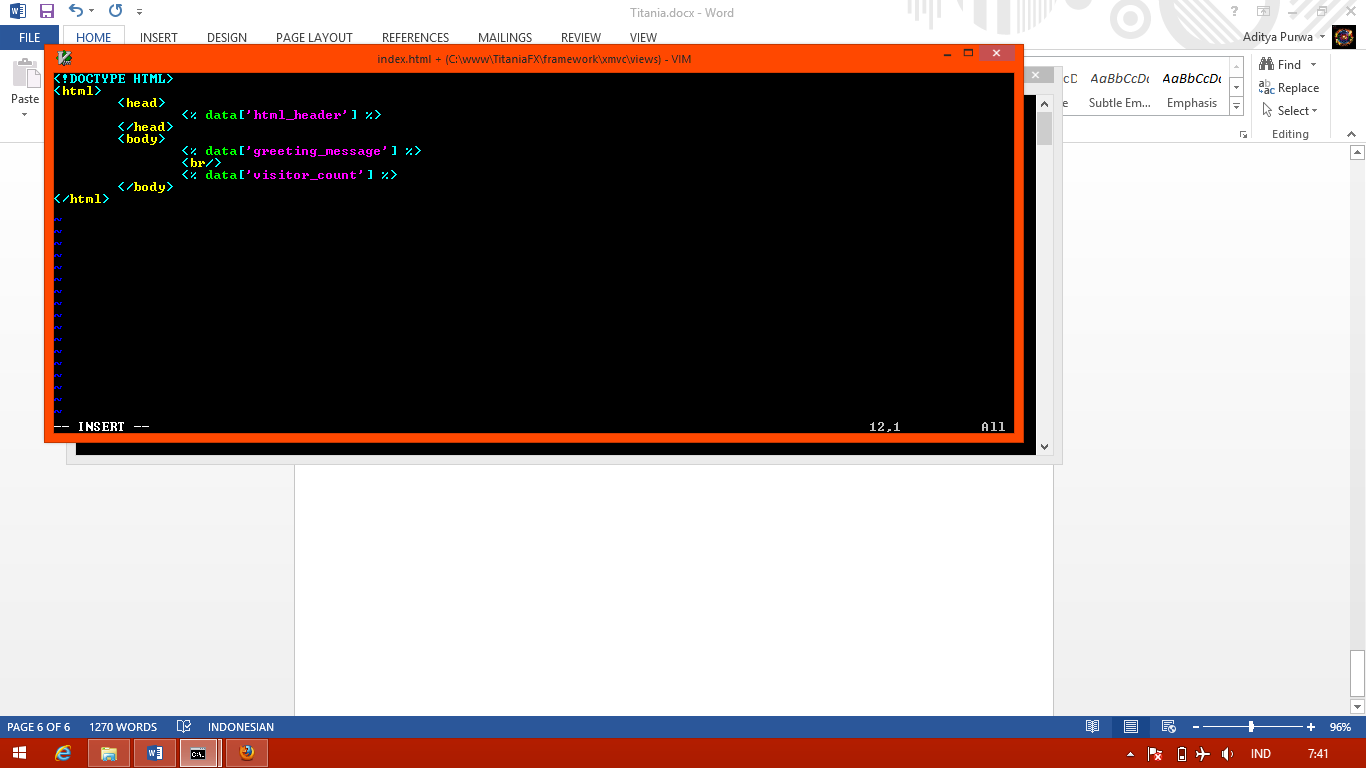
Okay, we assume that you already know MVC. XMVC is just an extended version of MVC. Which contains 3 core extensions. These extensions are:

1. Helper
   1. Helper is a class that helps the primary controller. Helper gives you more simplicity, you can put your controller hardwork here. The author of Titania recommends, that controller should only do 4 steps on each method. Which is,
      1. Take Input (Controller)
      2. Process Input (Controller give the input to Helper, which do all processing)
      3. Formatting (Controller ask Cenerator to format the data)
      4. Give Output (Helper, return the result. Controller, pass it to client)
2. Generator
   1. Generator is like pre-defined control in desktop development. On a web development, it generates a formatted version of the data. You pass a row result, and the generator encapsulate the row result on a table in HTML tag. This has not to be a HTML tag. You can also generate the data as XML, JSON, or whatever you want the data to be formatted.
3. Adapter
   1. During a web development using MVC. A lot of people out there puts query, database connectivity inside a model. This considered as a bad practice, because model should only contain placeholder and methods for client data. Adapter acts like DAO (Data Access Object), its do a separation between your model and database. All queries, database connectivity, or anything related with data extraction from its source should be placed on Adapter.

Okay, lets modify our Hello Titania program so it use these 3 aspects of XMVC.

## Creating View

Go to **/framework/xmvc/views** folder, and open **index.html**. You may see that it is only a greeting message from Titania. Delete everything inside this file, and begin to make your own HTML page here. Lets make it to look like this:



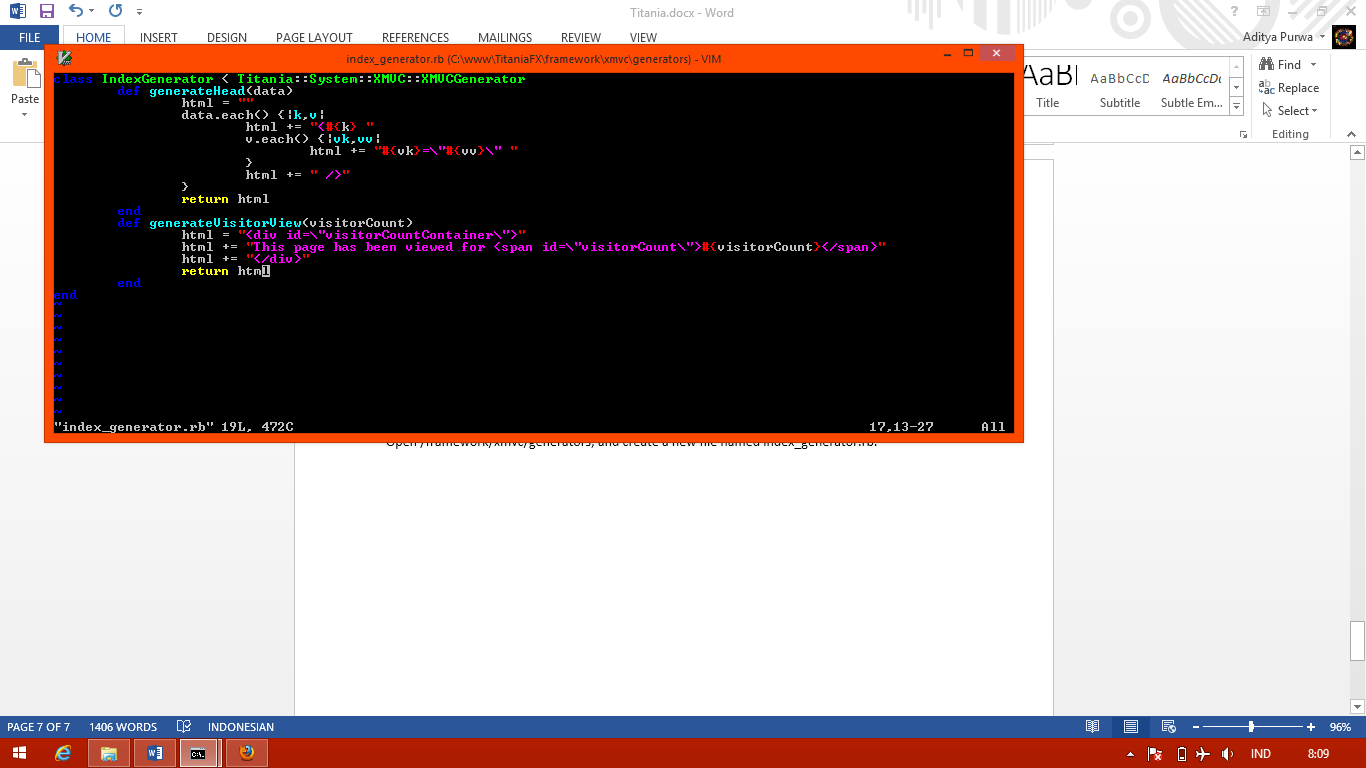
So, what happens? Well, nothing its actually just a HTML code. The only things you have to pay attention is to the ***Ruby Evaluation Context***syntax. Ruby Evaluation Context or simply pronounced as ***Rubec*** is a single line ruby code to indicates an output. Its enclosed between <% and %>. Just like JSP or ASP tag.

Multiline ruby code wont be evaluated, there is a reason why the author doesnt give it multiline support. The only reason is code clarity and integrity. If multiline support is available, your view file will sometimes contains formatting process. Which for xmvc-sake, it should be placed inside a generator.

So, a single line rubec wont hurt at all right? Its also make your code looks better, easier to read.

## Creating Generator

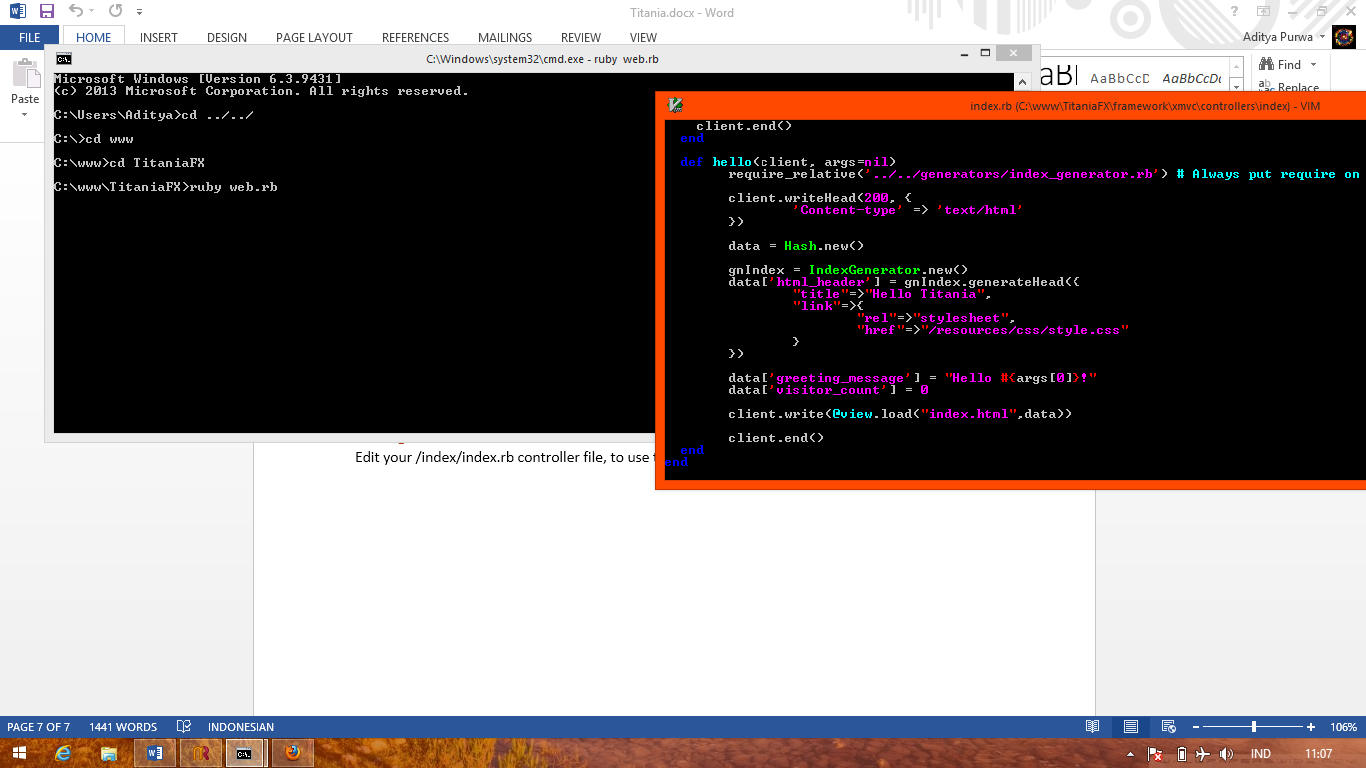
Open **/framework/xmvc/generators**, and create a new file named **index\_generator.rb**.



Another information is, that you dont have to extends your IndexGenerator from XMVCGenerator. But you should do it, remember, for integrity and consistency.

## Editing Your Controller

Edit your **/index/index.rb** controller file, to use the generator.

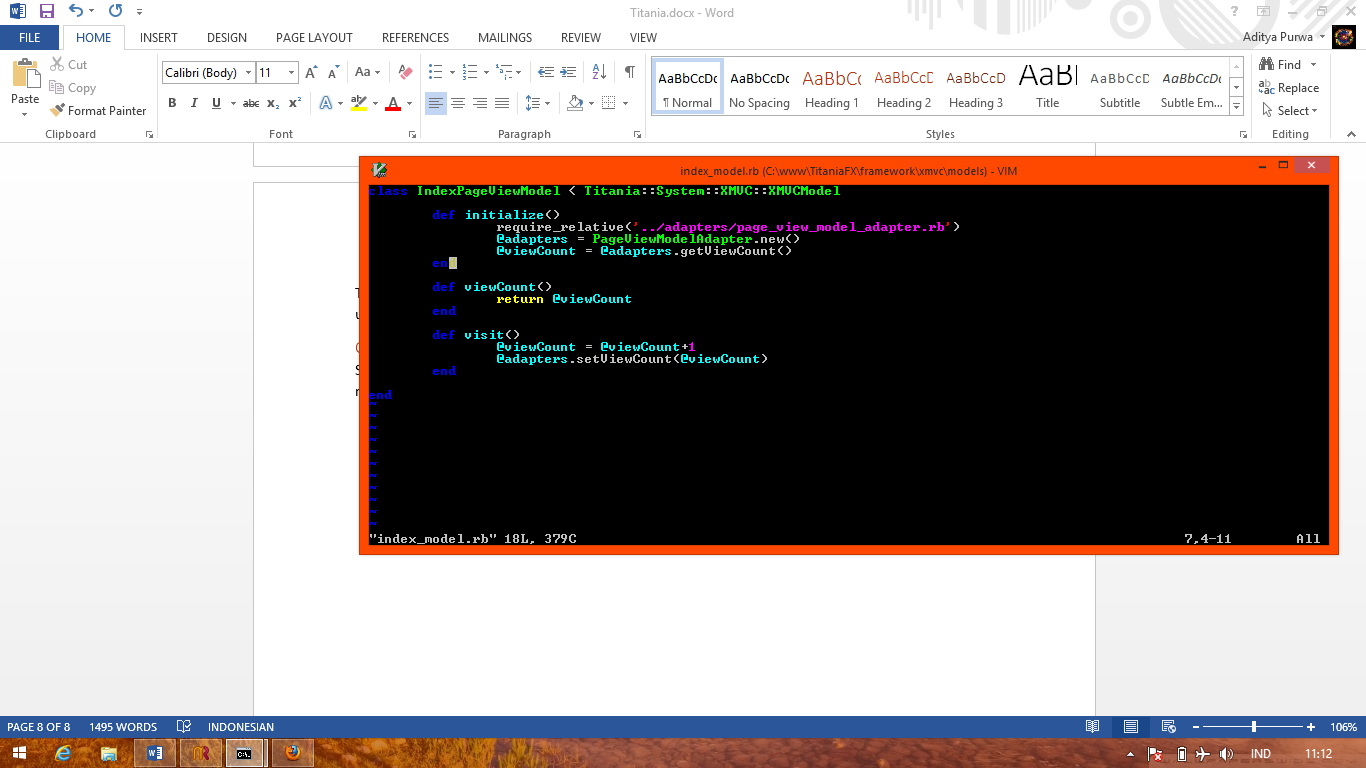


The generator will generate the header, its actually not a good practice to generate a html head using generator. Generator should be used to generate table data, dynamic menu, etc.

## Creating Model

So, the visitor counter always shows 0 till death do us part. We have to create a model that represent a counter.

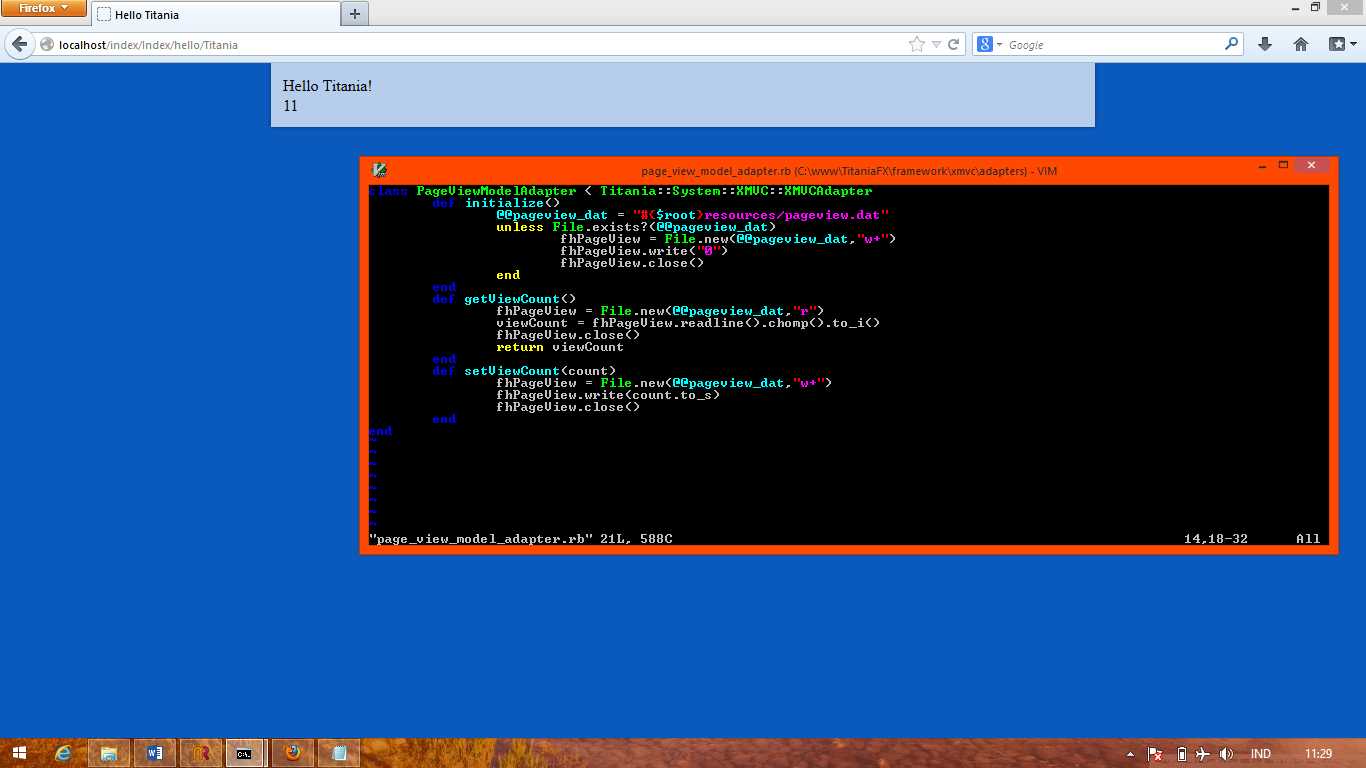
Create a model, **/models/index\_page\_view\_model.rb**



You may notice the first line of **initialize** method. It require an adapters for the model. Remember, model should stores the data only and method related to the data. It must not implement anything related to fetching the data, or anyting related to the model storage.

## Creating Adapter

Create an adapter, **/adapters/page\_view\_model\_adapter.rb**



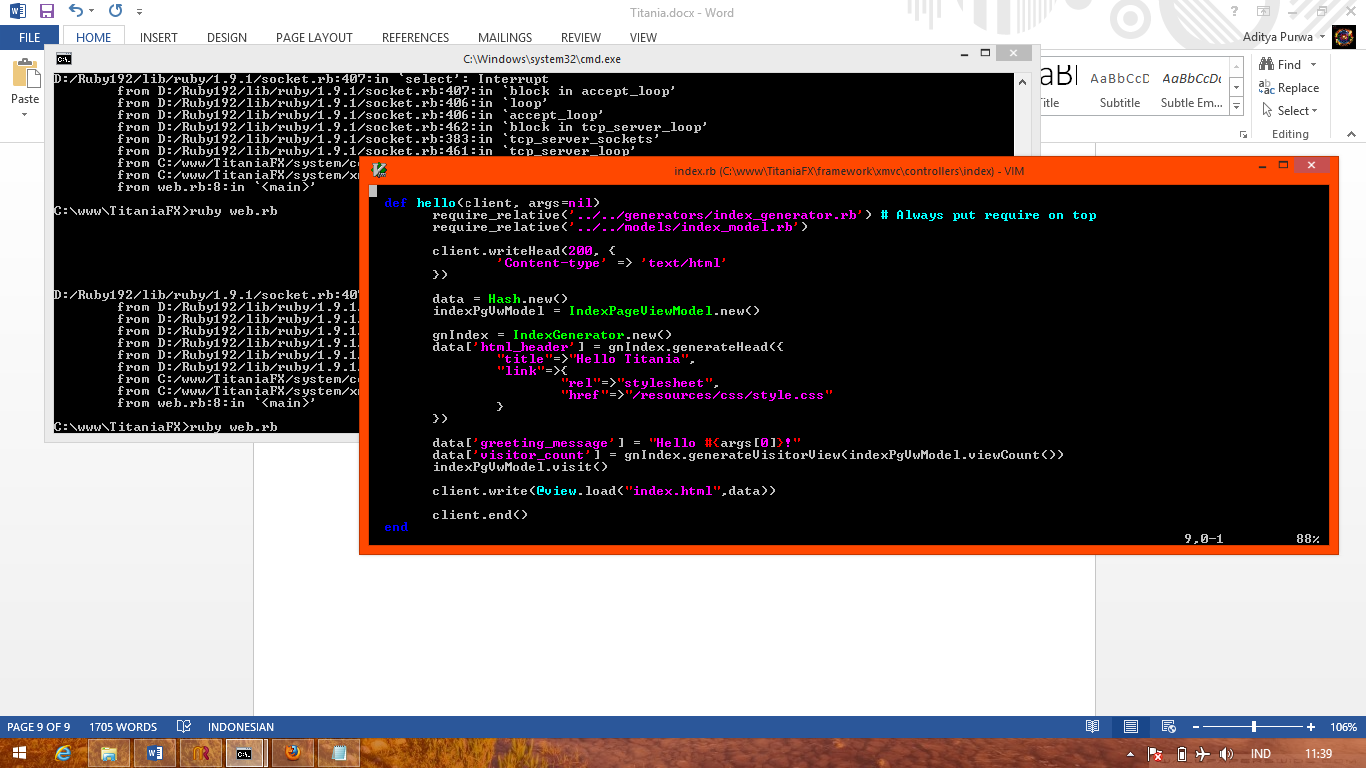
We’re creating an adapter that doesnt need a database. We use a local file storage at the server. Titania preview doesnt have native support for database.

## Editing The Controller

Now, after all XMVC elements has been properly created. Well, i know that you’re exhausted. Working on XMVC force you to work with many files. But, you’ll be thankful because XMVC is more maintainable, and you can do paralel development with your team using the XMVC design.

Our final step, is editing the controller, to use all XMVC elements, generator, model, and view. Helper is just a class that do something, all Ruby standard library classes were considered as a helper too. We dont need to create a helper because our side is really simple.

Edit the hello method inside our lovely **/index/index.rb** controller.



There it is! You have finally created your first XMVC framework, on the first framework that support XMVC, Titania!

You may want to add some styling using css resources. Give some little modification above so it does something cool. Its really up to you.

# XMVC + Titania Properly

Code smells, improper usage of XMVC elements will make your day stormy. Here is some recommendation you could follow if you want to use Titania with its XMVC design peacefully.

1. Use an IDE
   1. I know you love vim so much, but using vim on thousands files on a teamwork is really something.
   2. Use an IDE, RubyMine is enough.
2. Use XMVC elements properly
   1. Generators should be used to generate templated, small view.
   2. View should not contain any formatting. Well, its alread solved by forcing only a single line ruby code for dynamic output on your view. Just make sure you didnt do formatting inside a controller.
   3. Model is a model, let adapters do their hardwork. Controller should not interfere with the adapter, let the model use the adapter.

# !!! IMPORTANT !!!

Remember, that Titania is still on its preview version. Please report bugs, code smells, forgotten debug and test code to [aditya-app@live.com](mailto:aditya-app@live.com)

This document may also contains mispelled word, bad grammars, or anything that makes your reading hard. Report it to [aditya-app@live.com](mailto:aditya-app@live.com) also, we really appreciate your help.

A few minutes giving a feedback will make Titania grow bigger and faster, and we will be proud we can present you Titania in its best performance, for free, and for you only.

Thank you for using Titania,

Aditya Vi-“Avaga” Purwa

## Special Thanks

For you, the one who choose Titania, Ken Marinaris, and also The Light of Hope of 20th of June.