**What is Testing & How it is built in integrated with Rails?**

**Introduction:**

Testing is the process of executing a system to find the gaps, errors or the missing requirements comparing to the requirements. It’s one of the vital components of Software Development. It helps us to validate and verify the product meets the Business and Technical requirement and it’s working properly or not as expected. The quality of a product can be improvised with the help of testing.

Rails has made it very easy to write tests. It creates the skeleton test code behind the scene when the models and controllers are created. Simply running the test we can ensure the application provides desired functionality.

**How it is built in integrated with Rails:**

Every Rails application has 3 sides: production, development and testing. We can find there is three different sections for 3 unique database setups in config/database.yml file. This helps in setting up and interacting with test data without any danger of your tests altering data from your production environment.

Rails create a test folder for you as soon as you create a Rails project using rails new application\_name. The content of the test folder are as follows:

* fixtures/
* functional/
* integration/
* performance/
* unit/
* test\_helper.rb

The unit folder is meant to hold tests for models, the functional folder is meant to hold tests for controllers, and the integration folder is meant to hold tests that involve any number of controllers interacting. Fixtures provide functionality to organize test data; they reside in the fixtures folder. The default configuration for tests is present in the test\_helper.rb file.

**Preparing Application for Unit Testing:**

Before running tests, we need to ensure that the test database structure is current. For this following rake commands are used:

*rake db:migrate*

*. . .*

*rake db:test:load*

The rake db:migrate updates the db/schema.rb and the rake db:test:load recreates the test database from the current db/schema.rb

**Running Test:**

Running test is done by invoking the file containing the test cases. This is done by the following command:

*ruby -Itest test/unit/comment\_test.rb*

**Functional Test:**

Controllers process the incoming web requests to the application from client and they respond with a formatted view. Various actions of a single controller is tested under Functional Test. Functional test also involves User authentication and Page redirection. Following code is an example of functional test.

|  |
| --- |
| *test "should get index" do*  *get :index*  *assert\_response :success*  *assert\_not\_nil assigns(:comments)*  *end* |

In the given example, Rails simulates a request on the action called index, tests that the request was successful and checks the assignment of a valid comment was done or not.

**Integration Testing:**

Integration tests are useful for testing the interaction when more than one controller is there. Integration testing are vital because they handle the important work flow within the application. Integration tests have to be explicitly created under test/integration folder. A generator is provided to create Integration test skeleton by Rails. Here is an example:

*rails generate integration\_test visitor\_flows*

*exists  test/integration/*

*create  test/integration/visitor\_flows\_test.rb*

*require 'test\_helper'*

*class VisitorFlowsTest < ActionDispatch::IntegrationTest*

*fixtures :all*

*# Replace this with your real tests.*

*test "the truth" do*

*assert true*

*end*

*end*

**Testing Routes:**

Like everything it is also important to test all the routes. Here is an example test for a route:

|  |
| --- |
| *test "should route to comment" do*  *assert\_routing '/comments/1', { :controller => "comments", :action => "show", :id => "1" }*  *end* |
|  |

**Conclusion:**

As human are bound to make mistakes, so the programs written by human is supposed to have some mistakes or errors. To overcome this and more importantly to maintain the product quality testing is vital in software development. Testing provides many processes and control procedure to control the bugs. Rails provide built-in testing which is one of the benefits of using Ruby on Rails. Rails create automated test cases and provide harnesses and fixtures that makes test case easier to write and run. We can execute all tests using the rake utility.