Кратко Представяне на RoR

Ruby on Rails (често съкращавано на Rails или RoR) е популярен фреймуърк за разработка на уеб приложения, написан изцяло на програмния език Ruby. Характерно за RoR е, че дава на разработчиците изключителна бързина и гъвкавост при изграждането на уеб приложения. Мотото на RoR е – „Web development that doesn’t hurt!”.

RoR налага архитектурата Model-View-Controller (съкратено MVC) върху приложенията създавани чрез него. Този тип архитектура допринася за 3 основни неща:

* Изолиране на бизнес логиката на приложението от потребителския интерфейс
* Спомага за избягване на повторения в кода
* Отделяне на различните видове код, което от своя страна улеснява поддръжката



Ще се спрем на основните елементи на тази архитектура:

Моделите олицетворява данните, с които работи приложението и правилата за достъп до тях. За нуждите на RoR, моделите основно се използват за да управлява взаимодействието на приложението с база от данни. В повечето случаи един модел отговаря на една таблица в базата данни.

Изгледите представлява потребителския интерфейс на приложението. В RoR, изгледите основно са HTML файлове с внедрен Ruby код, където изпълнението на Ruby е свързано с изобразяване на данните. Главната задача на изгледите е да са посредник на информация между потребителя и останалата час от приложението.

Контролерите представлява връзката между изгледите, моделите и потребителя. Работата им се състои в това да получават заявки от потребителя, да извличат данни от моделите и да ги обработват ако е необходимо, и накрая да изпращат данните на изгледите.

Освен с гъвкавата си архитектура, RoR улеснява живота на разработчиците на уеб приложения и с големият брой инструменти, които той предлага. С тяхна помощ уеб програмирането става още по-бързо и приятно.

Увод в Тестването с RoR

Разработването на софтуер е сложна интелектуална дейност. Ето защо грешките, допускани при разработката на програми, са неизбежни, независимо от опита, който имат проектантите и програмистите. Не съществува методология на програмиране, която да гарантира, че създадените чрез нея програми са без грешки. Чрез тестването се прави опит да се въведе стил на изграждане на софтуер, при който както отделният програмист, така и целият екип като цяло да допуска по възможност по-малко грешки. Презумпцията, че всяка новосъздадена програма съдържа грешки, налага щателна проверка преди да се пристъпи към използването й.

Тестването е специална процедура, с изпълнението на която се цели откриване на грешки в **програмите**. То е едно от най-ефикасните неща, които могат да се направят за да се подобри качеството на софтуера. Честото тестването на софтуера е от изключителна важност, защото помага за ранното откриване на грешки в кода, което спестява много бъдещи разходи и усилия.

Практиката показва, че повечето разработчици тестват своите приложения. Проблемът е, че някои от тях ги тестват ръчно, а не автоматизирано. Недостатъкът на ръчното тестване е, че при всяка промяна по софтуер-а, тестващият трябва да повтаря едни и същи стари тестове, за да е сигурен, че новата добавка не поврежда старата функционалност. При една такава монотонна и повтаряща се дейност, възможността за човешка грешка или пропуск става много голяма. Когато софтуера нараства на големина, тази вероятност се увеличава още повече. Именно затова е препоръчително тестването да става по автоматизиран начин, тоест тестовете да се изпълняват от програма.

Автоматизираното тестването е заложено в основите на RoR. То е силно препоръчително и е почти неотлъчна част от всяко приложение разработено с RoR. Основната идеята на тестването е проста: написва се код, който проверява дали уеб приложението работи според очакванията. Веднъж написани, тестовете могат да се изпълняват многократно и да се използват всеки път когато има нужда да се провери дали всичко работи както трябва. По този начин процесът се автоматизира.

В RoR присъстват три основни вида тестове, като всеки от тях е насочен към различна част от дадено приложение.

* Unit тестове – използват се за тестване на моделите
* Functional тестове – използват се за тестване на функционалността на контролерите
* Integration тестове – използват се за тестване на взаимодействието между отделните контролери

Всяко приложение разработено с RoR имa 3 основни режима, в които може да работи.

* production
* development
* test

Един от тези режими, именно „test”, е предназначен специално за тестване. Този режим на работа изисква и специална база данни, различна от основната за приложението. Причината за това е, че често при тестване се откриват грешки и ходът на програмата не е изцяло предвидим, което води до голям риск от заличаване на данни. Именно затова се използва друга база от данни, в която няма ценни данни.

При създаване на скелет на ново приложение, RoR автоматично създава директория - test. Нейното предназначение е да съхранява всички файлове и директории свързани с автоматизираното тестване. Тя има следното съдържание:

* test
* fixtures - съдържа файлове с тестови данни
* functional - съдържа файлове с functional тестове
* integration - съдържа файлове с integration тестове
* performance – съдържа файлове с performance тестове
* unit - съдържа файлове с unit тестове
* test\_helper.rb – съдържа настройки на тестовете

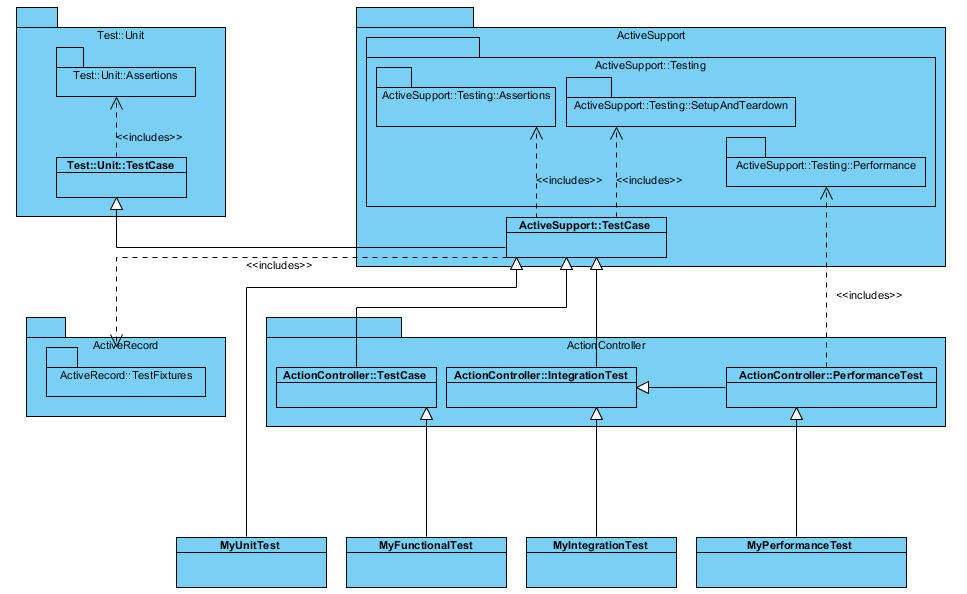
Реализация на Тестването с RoR

Модулите са широко използвани в езика за програмиране Ruby. Те много приличат на класове с изключение на две неща: не могат да се създават обекти от тип даден модул и модулите не могат да бъдат наследявани. Има две основни неща, за който модулите се ползват. Първото е, за да се групират методи, класове, константи и др. на едно място. По този начин те се капсулират и изолират, с което се предотвратяват евентуални противоречия между тях и останалия код. Второто е, за да се реализира вид множествено наследяване, което по принцип не е заложено в езика. То се получава, когато клас включи в дефиницията си даден модула. Тогава всички елементи на модула стават част от класа.

Сега ще се спрем по-подробно на класовете и модулите, който имат връзка с тестването в RoR. Тестовете в RoR се създават като самостоятелни класове, който наследяват определени класове от ядрото на RoR. По този начин те се сдобиват с допълнителна функционалност, която е от голяма полза за изграждането им.

* Unit тестовете наследяват класа TestCase, който се намира в модула ActiveSupport
* Functional тестовете наследяват класа TestCase, който се намира в модула ActionController
* Integration тестовете наследяват класа IntegrationTest, който се намира в модула ActionController
* Performance тестовете наследяват класа PerformanceTest, който се намира в модула ActionController

От своя страна класовете ActionController::TestCase, ActionController::IntegrationTest, ActionController::PerformanceTest наследяват класа ActiveSupport::TestCase. Той пък включва в себе си няколко модула и е наследник на класа Test::Unit::TestCase. По този начин можем да проследим цялата йерархия на класовете и модулите свързани с тестването в RoR. На следващата диаграма е илюстрирана една част от тази йерархия.



Заслужава си да споменем модула Test::Unit. Той представлява framework за unit тестване, създаден за езика Ruby. В последствие с развитието на RoR, този модул се е превърнал в основа, върху която да стъпят тестовете и тестването в новия framework. Класът Test::Unit::TestCase има обединяваща роля за Test::Unit. В него се събира цялата функционалност на модула. Една от най-важните функционалности, които той включва е тази на модула Test::Unit::Assertions. Test::Unit::TestCase се наследява от класа ActiveSupport::TestCase, който има централна роля в йерархията на класовете и от когото всички видове тестове в RoR наследяват функционалност по един или друг начин.