# Java Play Framework là gì?

* Play! là Java Web Framework duy nhất không liên quan gì đến Servlet Specification, rất lightweight, đơn giản dễ hiểu. Học ko loằng ngoằng như một số Java Web Framework khác như JSF, Struts. Chạy ko cần Tomcat tôm củng, ko cần GlassFish hay JBoss gì hết. Framework có luôn built-in App-Server cực kỳ nhẹ + performant.
* Nó nhằm mục đích tối ưu hóa năng suất phát triển bằng cách sử dụng quy ước cấu hình.

# Điểm khác nhau giữa java play và các framework khác.

* **Stateless**: Play 2 is full RESTful.
* **Intergrated unit testing**: Junit and Selenium support is include in the core.
* **API** comes with most required elements built-in.
* **Static methods**: all controller entry points are declared as static. After requests were made for this to be customisable, Play 2.1 now supports other styles of controllers, so controller need not be staticl however this is still the default.
* **Asynchronous I/O**: due to using JBoss Netty as its web server, Play can service long requests asynchronously rather than tying up HTTP threads doing business logic like Java EE frameworks that don't use the asynchronous support offered by Servlet 3.0.[17]
* **Modular architecture**: like Ruby on Rails and Django, Play comes with the concept of modules.
* **Native Scala support**: Play 2 uses Scala internally, but also exposes both a Scala API, and a Java API that is deliberately slightly different to fit in with Java conventions, and Play is completely interoperable with Java.

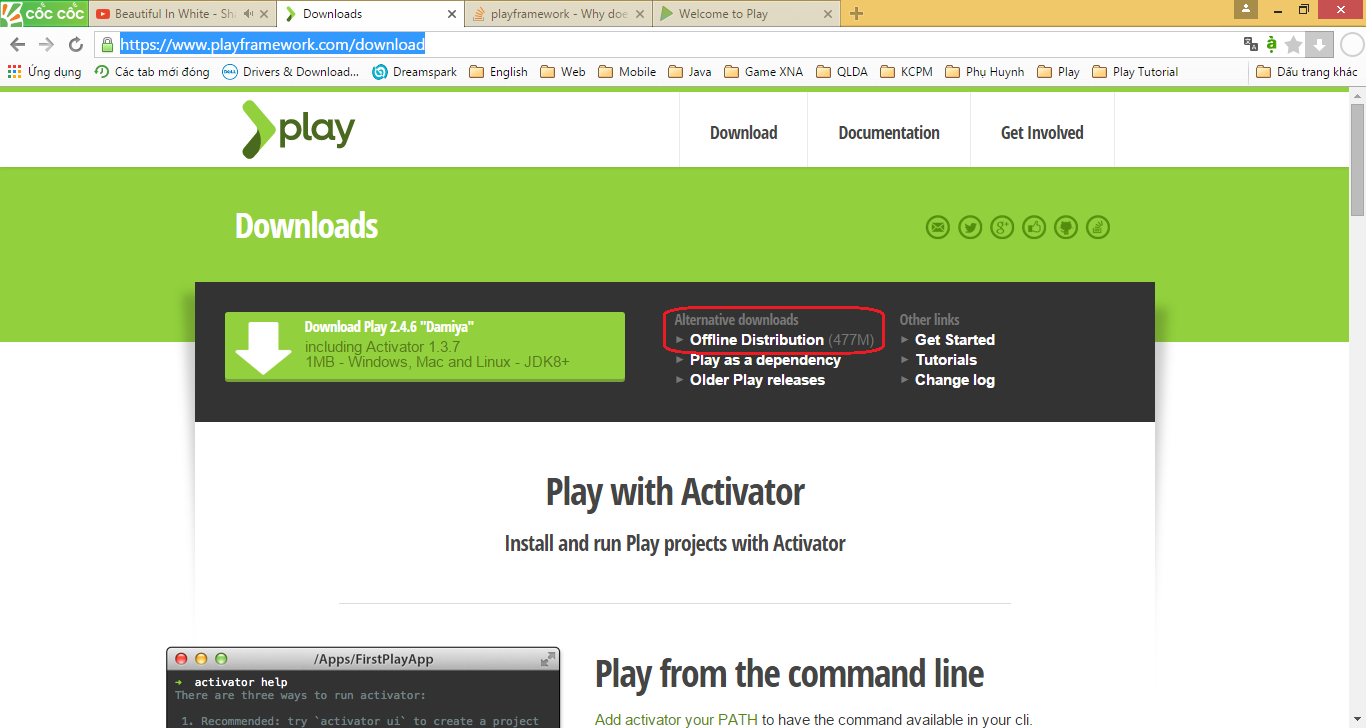
# Version 2.4

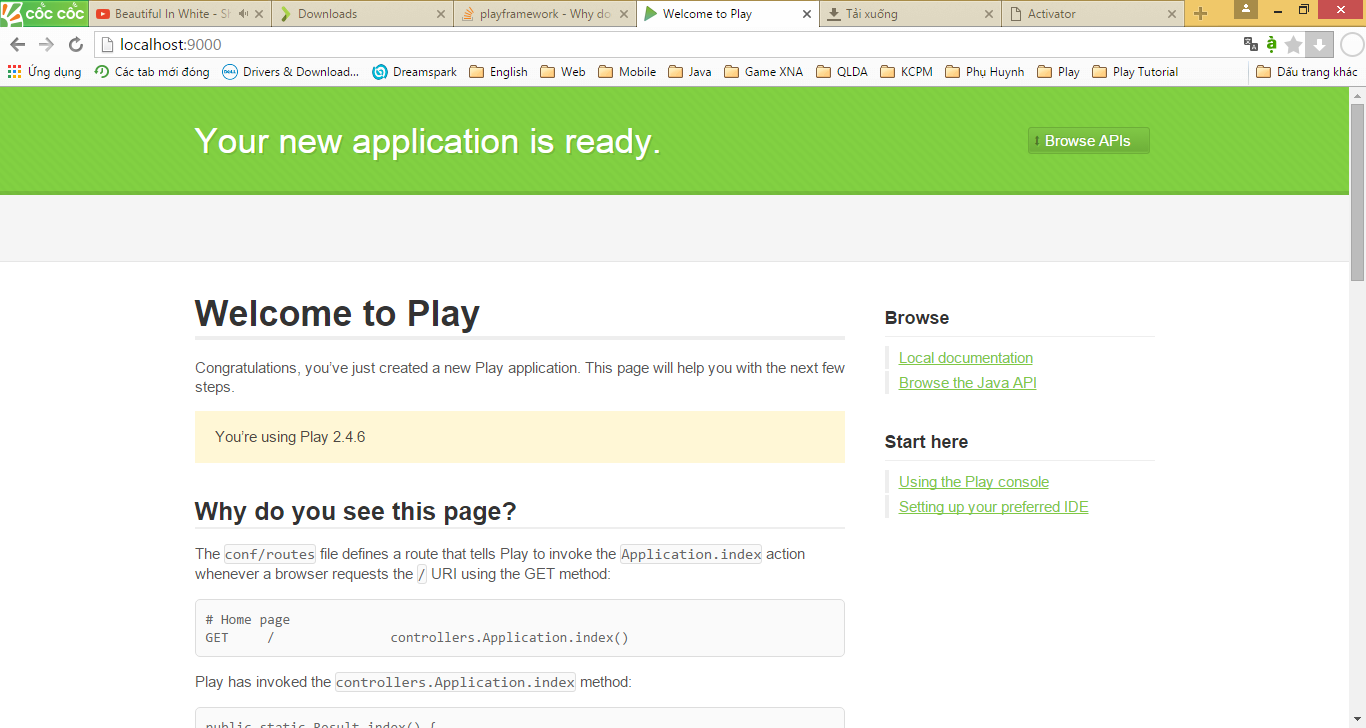
* Dependency Injection.
* Testing.
* Embedding Play.
* Aggregated reverse routers.
* Java 8 support.
* Maven/sbt standard layout.
* Anorm.
* Ebean.
* HikariCP.
* WS.
* Experimental Features

# Hướng dẫn tạo project play framework.

## Tạo project và chạy trên localhost.

* Download play version mới nhất. Hiện tại bản mới nhất là “**2.4.6**”. Chú ý là bản offline distribution theo đường link. <https://www.playframework.com/download>



* Ở đây mình sẽ giải nén gói “typesafe-activator-1.3.7” được tải về và để ở thư mục ổ **C:\play\2.4.6**.
* Tiếp tục bạn mở command line dòng lệnh lên. Di chuyển đến đường dẫn bạn để gói giải nén vừa tải. Ở đây mình là: C:\play\2.4.6.
* Gõ dòng lệnh: “**activator new**”. Tác dụng của dòng lệnh này sẽ cho bạn thấy các template mà bạn có thể tạo project.
* Ở phạm vi tìm hiểu thì mình chọn số “**5**”.
* Tiếp tục nhập “**name**” cho project: “**play-java-intro-1**”.
* Vậy là bạn đã tạo xong project “**java play framework**”. Tiếp tục, chúng ta sẽ **run** project và xem kết quả. Trước hết bạn phải di chuyển vào project vừa tạo. Ở đây mình là: “**cd play-java-intro-1**”.
* Nhập câu lệnh “**activator run**” để chạy project. Mở trình duyệt và nhập địa chỉ: “**localhost:9000**” để xem kết quả.
* 

## Mở project bằng eclipse kelper.

* Để mở project play framework bằng “**eclipse kelper**” bạn cần phải thêm dòng lệnh: “**addSbtPlugin("com.typesafe.sbteclipse" % "sbteclipse-plugin" % "4.0.0")**” này vào file “**plugins.sbt**” trong thư mục “**project**” của project vừa tạo. Ở đây là project “play-java-intro-1”.
* Mở command line, di chuyển vào thư mục project nhập câu lệnh “**activator eclipse**” để chuyển project play sang eclipse.
* Xong bạn mở eclipse: vào “**File🡪Import**” chọn “**Existing Project into Workspace**” trong mục “**General**”.
* Tiếp tục bạn chọn đến đường dẫn chưa project vừa tạo.
* **Finish**.

## Mở project băng IntelliJ IDEA 1.4.6

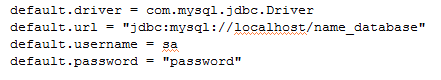
## Hướng dẫn sử dụng LOGGING.

* Mở file “**logback.xml**” trong thư mục “**conf**”. Thêm câu lệnh này vào “**<logger name="homecontroller" level="DEBUG" />**” bạn cần quan tâm đến **name** của logger để ta khai báo chính xác trong file controller.
* Mở một file controller mà bạn muốn ghi log. Ở đây là file “**HomeController**, ta khai báo logger vào: “**private final Logger.ALogger logger = Logger.of("homecontroller")**”.
* Giờ bạn có thể ghi log được rồi!
* Trong file “**logback.xml**”, ở thẻ root, các bạn chú ý:
  + <appender-ref ref="**ASYNCFILE**" /> : ghi lò ra file.
  + <appender-ref ref="**ASYNCSTDOUT**" />: ghi lò ra console.
* Java Play Framework cung cấp 6 mức độ logging.

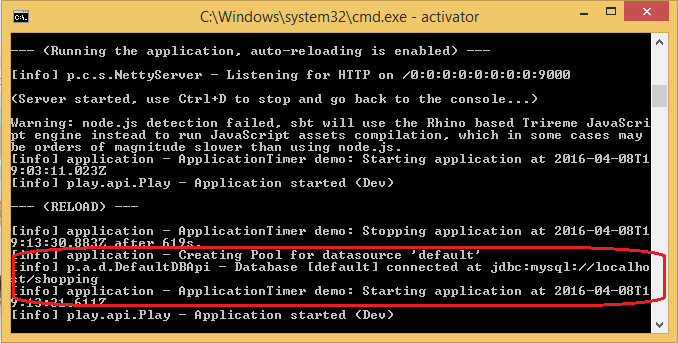
## Kết nối database với MySql.

Các bước để kết nối đến database MySql:

* Thêm dependency trong file “**build.sbt**”: “**"mysql" % "mysql-connector-java" % "5.1.18"**”.
* Cấu hình kết nối đến database trong file “**application.conf”**



* Vậy là project của bạn đã kết nối đến với CSDL MySql thành công.



## Managing database evolutions.

1. Tại sao lại phải quản lý database bằng evolutions.

* Giúp người lập trình tổ chức, lưu vết hoặc upgrade database dễ dàng.
  + Khi bạn làm việc mới nhóm, thì mỗi thành viên sẽ cần phải nắm rõ rằng database có gì thay đổi trước khi làm tiếp. Evolutions sẽ giúp nhà lập trình mô tả những thay đổi trước đó.
  + **Khi bạn deploy lên server, bạn cần có một cách để upgrade database của bạn khi cần.**
  + Khi bạn làm việc trên nhiều máy tính, nhiều nơi, bạn cần đồng bộ database.

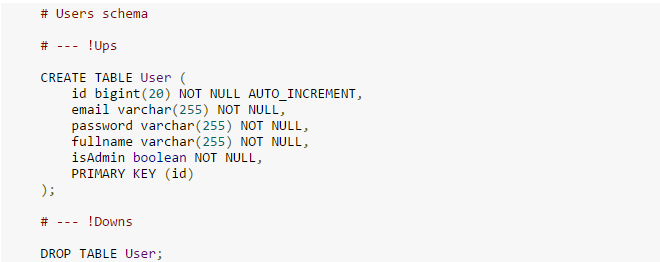
1. Enable evolutions.

Chỉnh sửa trong file “**build.sbt**”, thêm dependencies list:

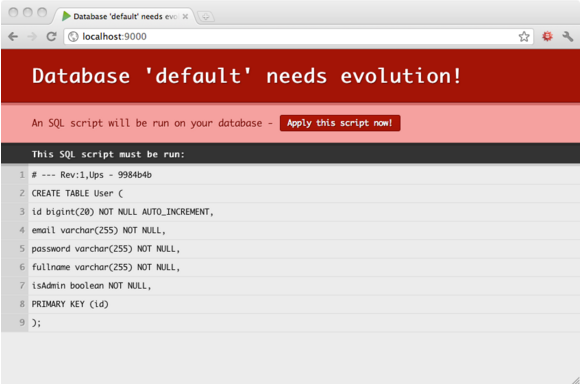
“**libraryDependencies += evolutions**”

1. Evolutions scripts.

* Play theo dõi database của bạn bằng một file scipt. File script này được viết băng ngôn ngữ SQL ở trong thư mục “**conf/evolutions/{default name database}\**”, mặc định là đường dẫn “**conf/evolutions/default/**”.
* File này được đặt tên theo số thứ tự, ví dụ: 1.sql, 2.sql,…
* Mỗi file đều có 2 phần.
  + Ups: mô tả những thay đổi cần thiết.
  + Downs: mô tả làm sao để phục hồi chúng.
* Ví dụ file 1.sql được mô tả như sau.



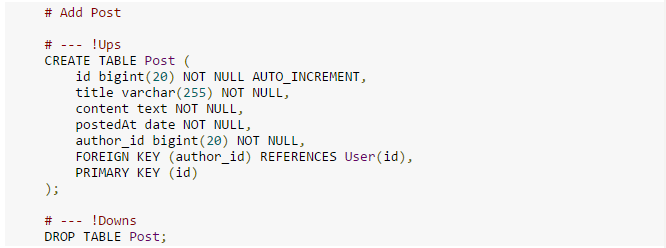
* Evulotions sẽ được tự động active trong Play. Bạn có thể disable evolutions bằng cách: chỉnh sửa dòng lệnh này “**play.evolutions.enable = true**” thành “**play.evolutions.enable = false**” trong file “**application.conf**”.
* Khi evolutions được active. Play sẽ kiểm tra trạng thái database schema của trước mỗi request in DEV mode, hoặc trước khi start ứng dụng in PROD mode. Ở DEV mode, nếu database chưa được cập nhật thì Play sẽ thông báo lỗi và đề nghị đồng bộ database bằng cách chạy script sql được được xây dựng hoặc thay đổi trước đó.



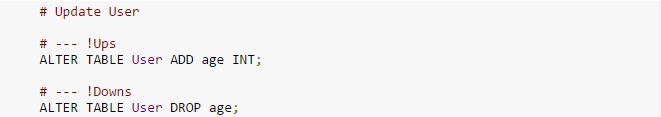
* Evolutions có thể được cấu hình hoàn toàn bằng config trong file “**application.conf**”. Cấu hình cho application, sử dụng từ khóa với prefix “**play.evolutions**”, cấu hình cho database sử dụng từ khóa “**play.evolutions.db.<datasourcename>**” ví dụ là: “**play.evolutions.db.default**”. Có các tùy chọn cấu hình hỗ trợ sau:
  + **Enable**: Cho phép active or disable evolutions. Mặc định là “**true**”.
  + **Schema**: Tên CSDL, mặc định là “**default**”
  + **Autocommit**: Mặc định là “**true**”.
  + **useLocks**:
  + **autoAply**: Mặc định là “**false**”. Ở DEV mode, có 2 scripts UPS và DOWNS sẽ được tự động apply. Ở PROD mode, **chỉ** có UPS evolutions được tự động apply.
  + **autoAplyDowns**: Mặc định là “**false**”, không tác động đến DEV mode. Nếu trở thành “**true**” thì scripts ở DOWNS sẽ tự động apply.

1. Đồng bộ database.

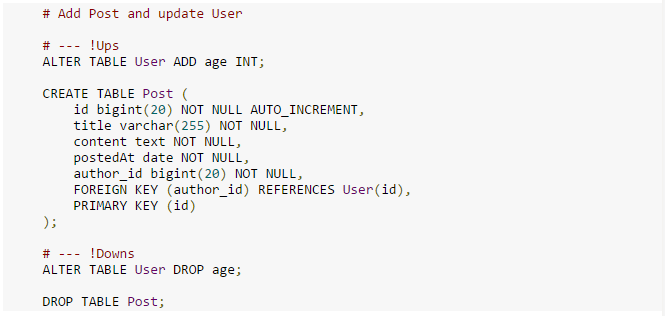
* Giả sử có 2 developers làm việc chung một project. Developer A làm việc yêu cầu bảng mới và A tạo file “**2.sql**”



* Tới developer B lại muốn chỉnh sửa lại bảng trong file “**1.sql**”.



* Khi B hoàn thành và commit lên github (giả sử A và B sử dụng github để lưu project). Thì khi A tiếp tục làm việc, thì Play sẽ merge sự thay đổi của database trước khi làm việc.



* So, Play sẽ phát hiện và yêu cấu A đồng bộ database lại với nhau.

## Intercepter trong Java Play Framework.

Các bước để bắt các event như: start application, stop application, error,.. được thực hiện như sau:

* 1. Tạo class “**Global**” trong pakage “**intercepter**”.
  2. Class “**Global**” kế thừa lớp “**GlobalSettings**” và override lại các hàm Start, Stop, error.
  3. Thêm vào file “**application.conf**” dòng “**application.global=intercepter.Global**”

<https://www.playframework.com/documentation/2.5.x/JavaErrorHandling>

## Tại sao lại sử dụng Result, CompletionStage<Result> và cách sử dụng?