Play-Slick:
Integrating Play-Slick with Play Framework

Himanshu Gupta
Software Consultant
Knoldus Software LLP

Agenda

- What is Slick ?
- Introduction to Play-Slick.
- Why Play-Slick ?
- Setup
- DBAction
- DDL Plugin
- Modular Models
- Live Demo

knólx

What is Slick ?

Image source - http://invasivespeciesireland.com/what-can-i-do/

knólx

What is Slick ?

Scala Language Integrated Connection Kit

.

source - http://blog.knoldus.com/2014/12/12/knox-session-working-with-slick-2-1-0/

knólx

Introduction to Play-Slick

- Provides a wrapper DB object that uses the data sources defined in Play framework's conf files.
- A DDL plug-in, that reads Slick tables and Automatically creates schema updates on reload.
- In addition it contains a wrapper to use Play Enumeratees together with Slick.

source - https://github.com/playframework/play-slick

knólx

Why Play-Slick ?

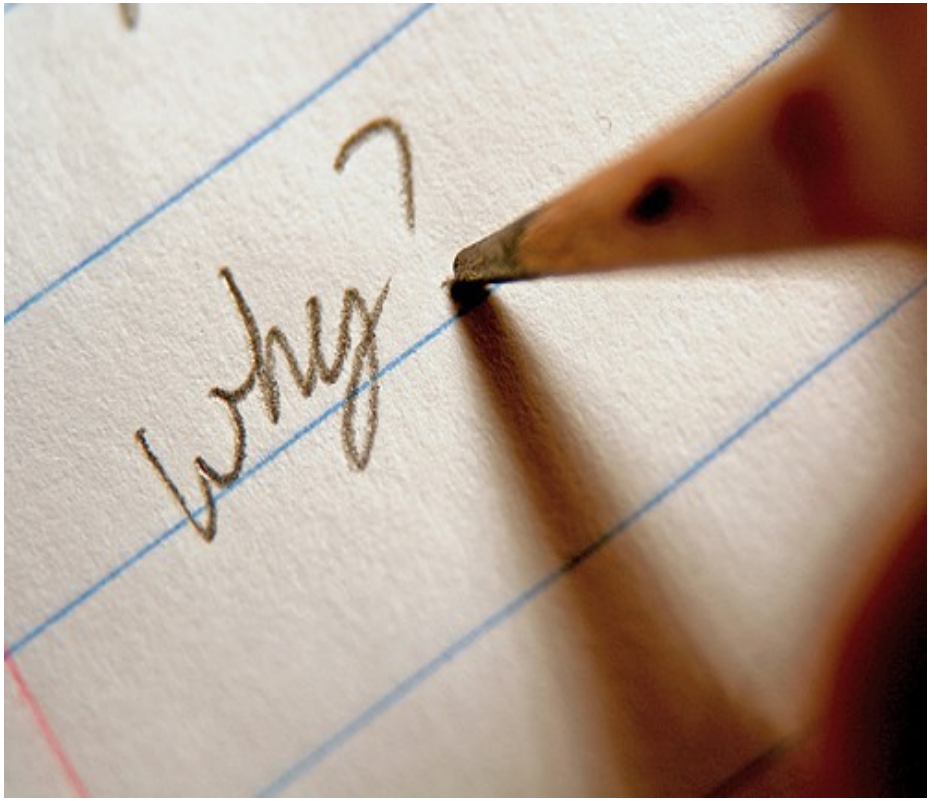

Image source - http://mollygalbraith.com/wp-content/uploads/2012/02/why.jpg

knólx

Why Play-Slick ?

- Easy to Create and Update Schema.
- Implicit Load-Balancing of Traffic.
- No need of providing Driver to DB object explicitly.
- Provides functionality to use Play Enumeratees together with Slick.

Setup

- Add following dependency in build.sbt file:

"com.typesafe.play" %% "play-slick" % "0.8.1"

- Add following configurations in application.conf:

db.default.driver=org.postgresql.Driver
db.default.url="jdbc:postgresql://localhost:5432/slickdemo"
db.default.user="postgres"
db.default.password="12345"

DBAction

Create:

```
  def create = DBAction { implicit rs =>
    Ok(html.createForm(computerForm, Companies.options))
  }
```

Read:

```
def list(page: Int, orderBy: Int, filter: String) = DBAction { implicit rs =>
    Ok(html.list(
      Computers.list(page = page, orderBy = orderBy, filter = ("%"+filter+"%")),
      orderBy, filter
    ))
  }
```

DBAction (continue...)

Update:

```
def update(id: Long) = DBAction { implicit rs =>
    computerForm.bindFromRequest.fold(
      formWithErrors => BadRequest(html.editForm(id,formWithErrors,Companies.options)),
      computer => {
        Computers.update(id, computer)
        Home.flashing("success" -> "Computer %s has been updated".format(computer.name))
      }
    )
  }
```

Delete:

```
def delete(id: Long) = DBAction { implicit rs =>
    Computers.delete(id)
    Home.flashing("success" -> "Computer has been deleted")
  }
```

DDL plugin

Enables DDL Schema Generation

```
slick.default="models.*"
```

Modular Models

Example

Modular Models (continue...)

Pros:

- Compared to Slick there is no Risk of getting Runtime Errors.
- No need of “**implicit session: Session**” on all methods.
- An arguably clean way of separating the definition of the table in DB and the methods you need to use.

Modular Models (continue...)

Pros:

- Compared to Slick there is no Risk of getting Runtime Errors.
- No need of "**implicit session: Session**" on all methods.
- An arguably clean way of separating the definition of the table in DB and the methods you need to use.

Cons:

- We have an extra BusinessObjects object.
- We have to specify the Session or Transaction elsewhere.

Live Demo

knólx

Q & Option[A]

References

- https://www.playframework.com/documentation/2.3.x/Modules
- https://github.com/playframework/play-slick
- https://github.com/playframework/play-slick/blob/master/docs/manual/working/scalaGuide/main/sql/slick/ScalaSlick.md

Thanks