Play Framework and Evolutions

Produced by:

Eamonn de Leastar (edeleastar@wit.ie)

Dr. Siobhán Drohan (sdrohan@wit.ie)

Waterford Institute of Technology
INSTITIÚID TEICNEOLAÍOCHTA PHORT LÁIRGE

Department of Computing and Mathematics
http://www.wit.ie/

Database Evolution

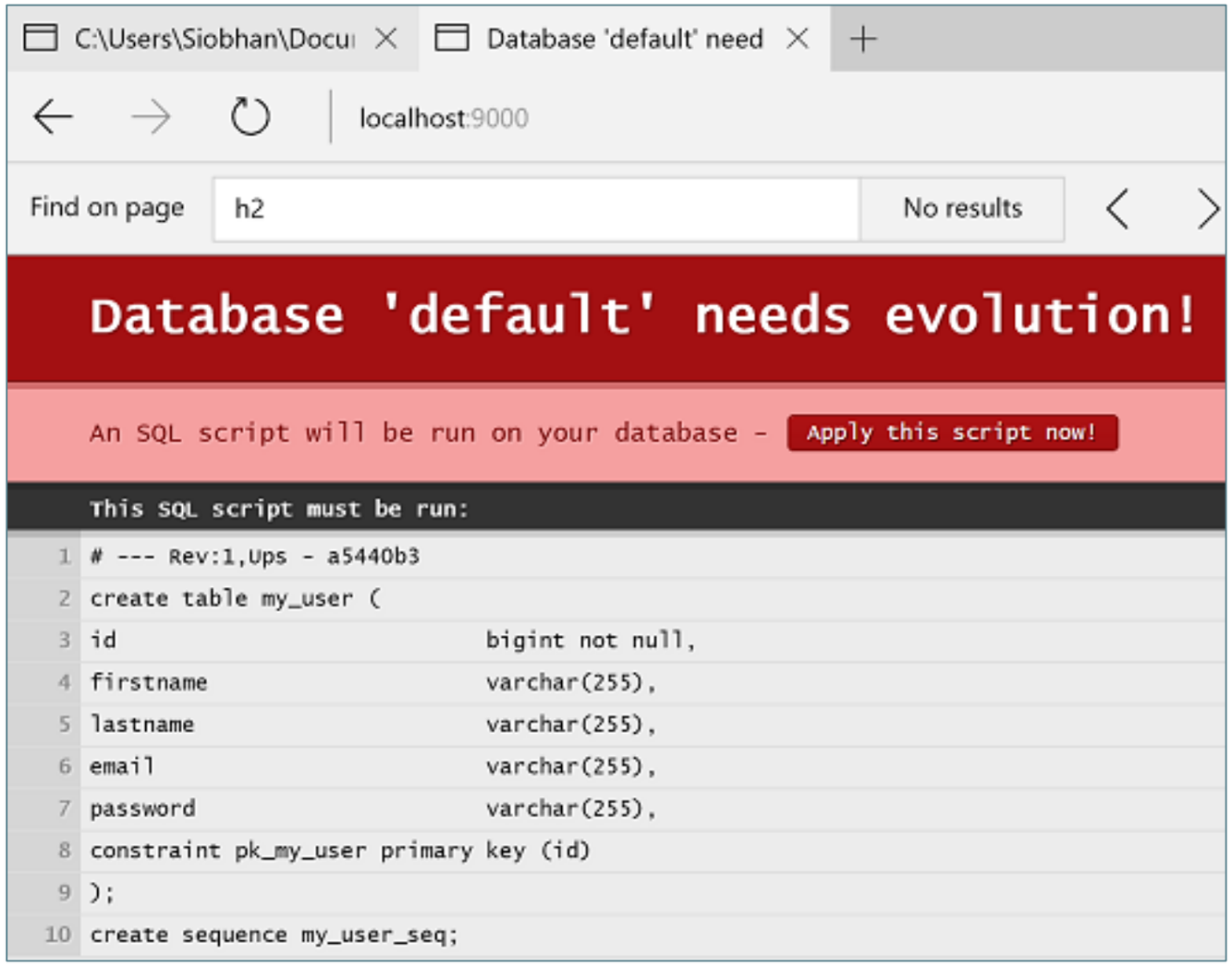
- Database evolution (sometimes called schema evolution) refers to the problem of evolving a database schema to adapt it to a change in the modeled reality.

- The problem is not limited to the modification of the schema, also affecting the data stored under the given schema

> *"The problem has been recognized as a very pressing one by the database community for more than 12 years …* support for Schema Evolution, is a difficult problem involving complex mapping among schema versions, the tool support has been so far very limited. The recent theoretical advances on mapping composition and mapping invertibility, which represent the core problems underlying the schema *evolution remains almost inaccessible to the large public"*

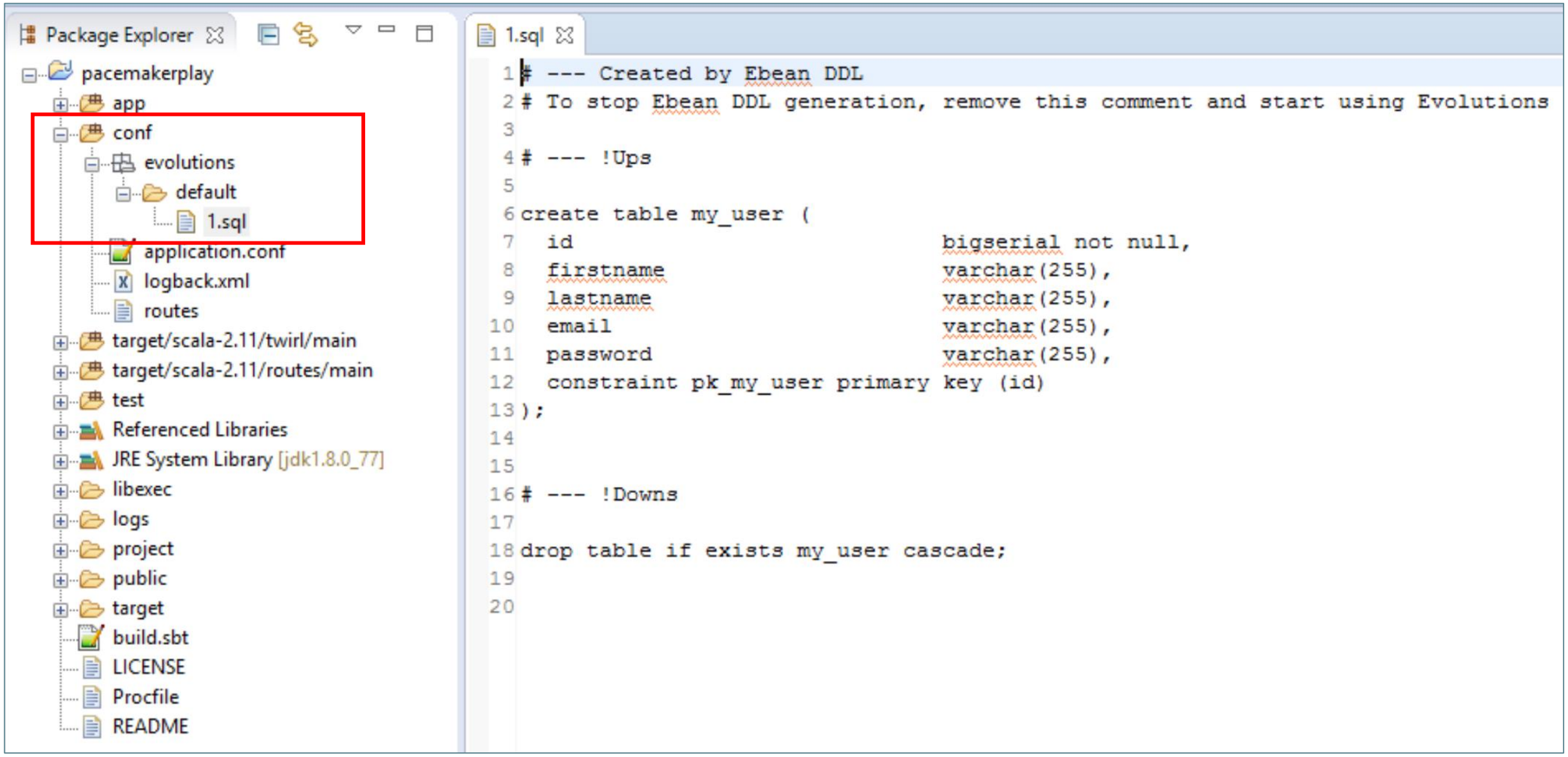
http://en.wikipedia.org/wiki/Schema_evolution

Database needs Evolution!

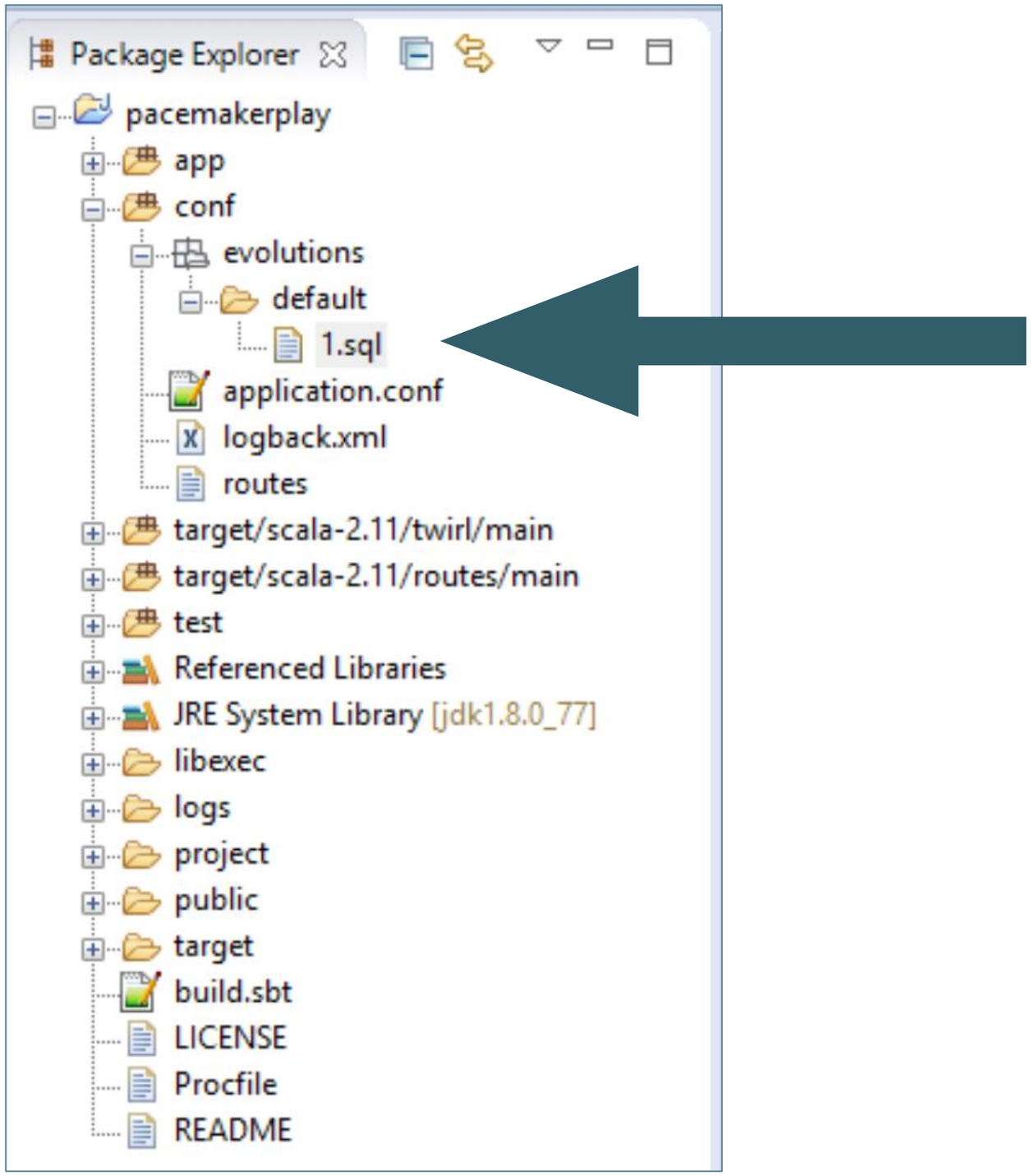


- An **evolution** is a SQL script that migrates the database schema from one state to the next in your application.
- Every time to make a change to the model, the database must be ‘evolved’.
- This is done via play generated evolution scripts.
- These scripts must be run before application starts.

Evolution SQL Script



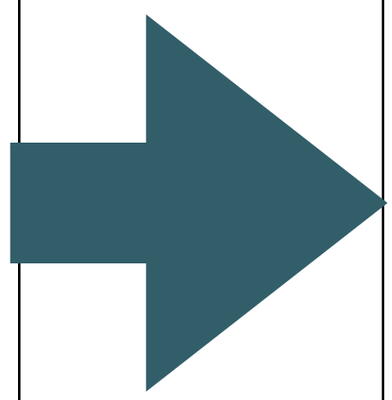
Database Evolution in Play (2)



- Play monitors model classes and generates a new SQL script if it detects a change in the model from the pre-existing script.
- This script can also be manually updated and maintained.

Evolution Script (1)

```java
@Entity
@Table(name="my_user")
public class User extends Model
{
  @Id
  @GeneratedValue
  public Long   id;
  public String firstname;
  public String lastname;
  public String email;
  public String password;
  …
```



```
# --- Created by Ebean DDL

# --- !Ups

create table my_user (
  id                          bigint not null,
  firstname                   varchar(255),
  lastname                    varchar(255),
  email                       varchar(255),
  password                    varchar(255),
  constraint pk_my_user primary key (id))
;

create sequence my_user_seq;

# --- !Downs

drop table if exists my_user cascade;

drop sequence if exists my_user_seq;
```

Evolution Script (2)

```
@Entity
@Table(name="my_user")
public class User extends Model
{
  @Id
  @GeneratedValue
  public Long   id;
  public String firstname;
  public String lastname;
  public String email;
  public String password;
  public String nationality;
  …
```

```
# --- Created by Ebean DDL

# --- !Ups

create table my_user (
  id                          bigint not null,
  firstname                   varchar(255),
  lastname                    varchar(255),
  email                       varchar(255),
  password                    varchar(255),
  nationality                 varchar(255),
  constraint pk_my_user primary key (id))
;

create sequence my_user_seq;

# --- !Downs

drop table if exists my_user cascade;

drop sequence if exists my_user_seq;
```

More interesting model

```
@Entity
@Table(name="my_user")
public class User extends Model
{
  @Id
  @GeneratedValue
  public Long   id;
  public String firstname;
  public String lastname;
  public String email;
  public String password;

  @OneToMany(cascade=CascadeType.ALL)
  public List<Activity> activities = new ArrayList<Activity>();
  //…
}
```

```
@Entity
public class Activity extends Model
{
  @Id
  @GeneratedValue
  public Long   id;
  public String type;
  public String location;
  public double distance;
  //…
}
```

`CascadeType.ALL:` persistence will propagate (cascade) all `EntityManager` operations (`PERSIST, REMOVE, REFRESH, MERGE, DETACH`) to the relating entities.

application.conf

- Database URL different for local/Heroku database
- Database Driver also different!
- This implies the syntax of the evolution script may differ depending on which driver is loaded.

```
#Remote Heroku PostgresQL database
default.driver=org.postgresql.Driver
default.url=${DATABASE_URL}

#Local h2 database
#default.driver = org.h2.Driver
#default.url = "jdbc:h2:mem:play"
#default.username = sa
#default.password = ""
```

Different Evolutions!

```
create table activity (
  id                    bigint not null,
  user_id               bigint not null,
  type                  varchar(255),
  location              varchar(255),
  distance              float,
  constraint pk_activity primary key (id))
;

create table my_user (
  id                    bigint not null,
  firstname             varchar(255),
  lastname              varchar(255),
  email                 varchar(255),
  password              varchar(255),
  constraint pk_my_user primary key (id))
;

create sequence activity_seq;
create sequence my_user_seq;

alter table activity add constraint fk_activity_my_user_1 foreign key
(user_id) references my_user (id);
create index ix_activity_my_user_1 on activity (user_id);



drop table if exists activity cascade;
drop table if exists my_user cascade;
drop sequence if exists activity_seq;
drop sequence if exists my_user_seq;
```

Postgres

```
create table activity (
  id                    bigint not null,
  user_id               bigint not null,
  type                  varchar(255),
  location              varchar(255),
  distance              double,
  constraint pk_activity primary key (id))
;

create table my_user (
  id                    bigint not null,
  firstname             varchar(255),
  lastname              varchar(255),
  email                 varchar(255),
  password              varchar(255),
  constraint pk_my_user primary key (id))
;

create sequence activity_seq;
create sequence my_user_seq;

alter table activity add constraint fk_activity_my_user_1 foreign key
(user_id) references my_user (id) on delete restrict on update
restrict;
create index ix_activity_my_user_1 on activity (user_id);


SET REFERENTIAL_INTEGRITY FALSE;
drop table if exists activity;
drop table if exists my_user;
SET REFERENTIAL_INTEGRITY TRUE;
drop sequence if exists activity_seq;
drop sequence if exists my_user_seq;
```

MySql

Switching Drivers

- The remote configuration will not run locally.
- ${DATABASE_URL} is only valid inside the Heroku environment:

  i.e. it indicates the database connection string is to come from the environment variable on Heroku.

```
#Remote Heroku PostgresQL database
default.driver=org.postgresql.Driver
default.url=${DATABASE_URL}

#Local h2 database
#default.driver = org.h2.Driver
#default.url = "jdbc:h2:mem:play"
#default.username = sa
#default.password = ""
```

- → Evolution will not be generated unless:
  - Use Postgres database locally
  - Connect to Postgres in Heroku

Connecting Local App to Postgres on Heroku (1)

- Locate the JDBC connection string for your database on Heroku by entering the following command in your Git shell:

```
heroku pg:credentials DATABASE
```

- Your connection info string similar to the one below will be returned:

```
Connection info string:
  "dbname=d5aesl5qn4beho host=ec2-107-21-222-62.compute-1.amazonaws.com port=5432 user=liynenxndfmqqz password=JpYRkxeLpMV3pItfCID3ZjVIf7 sslmode=require"
Connection URL:
  postgres://liynenxndfmqqz:JpYRkxeLpMV3pItfCID3ZjVIf7@ec2-107-21-222-62.compute-1.amazonaws.com:5432/d5aesl5qn4beho
```

Connecting Local App to Postgres on Heroku (2)

However the Postgres JDBC driver uses the following convention:

```
jdbc:postgresql://<host>:<port>/<dbname>?user=<username>&password=<password>
```

Format your connection string accordingly, and place it in your application.conf as your default url:

```
db.default.driver=org.postgresql.Driver
db.default.url="jdbc:postgresql://ec2-107-21-222-62.compute-1.amazonaws.com:5432/d5aesl5qn4beho?user=liynenxndfmqqz&password=JpYRkxeLpMV3pItfCID3ZjVIf7"
#db.default.url=${DATABASE_URL}

#db.default.driver=org.h2.Driver
#db.default.url="jdbc:h2:mem:play"
#db.default.user=sa
#db.default.password=""
```

Connecting Local App to Postgres on Heroku (3)

- The connection string will need one more fragment before it can work - append the following directly to the end of the string:

```
&ssl=true&sslfactory=org.postgresql.ssl.NonValidatingFactory
```

- Restart the local app, it should be using the Postgres database on Heroku.

Apply Evolutions Automatically

- We can choose to have our schema evolutions done automatically by adding **evolutions** as a library dependency to our **build.sbt:**

```
libraryDependencies ++= Seq(
  javaJdbc,
  cache,
  javaWs,
  evolutions,
  "org.postgresql" % "postgresql" % "9.4-1201-jdbc41",
  "net.sf.flexjson" % "flexjson" % "3.3")
```

- and updating the **play.evolutions** setting in **application.conf** to have the autoApply set to true:

```
play.evolutions {
  # You can disable evolutions for a specific datasource if necessary
  #db.default.enabled = false
  db.default.autoApply=true
  db.default.autoApplyDowns=true
}
```

Approach: Evolve Locally - Deploy Remotely

- Evolve the database locally.
- Commit the generated SQL script to Git.
- Push to Heroku.
- This will trigger a remote evolution.