Please read thru the overall exercises overview.

Exercise 10

Your goal is to extend the starter Play application to allow users to create, read, delete (CR/U/D) observations of whales. There is no need to support update or delete in this app.

Learning Objectives

- Using Play framework, and learning about web development frameworks and architecture.
- Implementing a web app using java: routing, controllers, models, views.
- Applying OO design concepts like encapsulation into a web app framework.
- Working with complex frameworks in teams; work assignment and schedule management.
- Front-end development.
- A bit (tiny bit) about sbt, the Scala Build Tool.

Requirements

1. The app shall run locally only. Hosting is not required (but see bonus in the marks.md file).
2. The app shall **display** Observations and Whales.
    1. Whales have species, estimated weight, gender, and id fields.
    2. Observations have a collection of whales observed, location, date/time (to the nearest hour).
3. The app shall enable a user to **create** a new Whale record.
4. The app shall enable a user to **create** a new Observation record.
5. The app shall support the ability to **search** on Whale species (e.g., "show all Orcas") and Observation date (e.g.,"show all Whales seen on 2012-01-01")
6. Following the example here, create a REST api for your app. The REST API only has to implement ONE endpoint: `GET http://localhost:9000/whales` with mime-type set to application/txt+json should return the JSON representation of the whales currently entered. Note that this involves content negotiation since you will also have an endpoint `/whales` that will return HTML if that was requested (ie. by a web browser). See here under "Content". Note that Play has builtin JSON parsing using Jackson at `import play.api.libs.json.Json`.
7. The source code shall have unit tests that test every operation on the model.
8. Create at least 2 ADRs that explain 1) how you have architected the app (base this on the overall Play architecture) and 2) how you have

implemented the front end of the app.

9. The app's internal architecture shall follow principles discussed in the course with respect to code quality and OO principles. This includes following best practices in MVC using Play.

Deliverables

- The source code of the complete web application.
- Tests, showing functionality.
- The ADRs relevant to the 2 major decisions.
- Each team member must document their contribution in the deliverables.md file. *Each team member should own at least one feature.* In a separate `contributions.md` file, list each team member and their contribution in a few bullet points (e.g., 'implemented unit test for whale JSON').

Tips

- This is a more complex project. While technically I think it is less challenging than Ex9 or Ex 8, it has more moving pieces and so coordination with your team is important. **Reach out to your TA if your group is struggling!**
- There are several required features. Start with the smallest ones and get those working completely. Do *not* divide tasks by tier (e.g., back-end/frontend) because in my experience that becomes a nightmare to integrate. Use vertical/feature slices (e.g., implement the "add Whales" function).
- This starter code uses SBT to build the site. There is SBT integration in IntelliJ with the Scala plugin (preferences-> plugins). You can also start SBT at the command line (I've tested using 1.3.13). Then type `compile` and `run` to get the main program working (run depends on compile, so just the one command is enough). `test` is the other command you will need.
- If all goes well you will see `[info] p.c.s.AkkaHttpServer - Listening for HTTP on /0:0:0:0:0:0:0:0:9000` which has started the app for you at http://localhost:9000 (note the port number). If you have other webapp tools or database servers running, that port may not work.
- The file `build.sbt` is where you can manage dependencies etc. See the help file for more on how to update/add dependencies.
- Follow the following Project structure to keep everything organised:

  images at: public/images/

  javascript files at: public/javascripts/

  CSS at: public/stylesheets

all application routes to be mentioned at conf/routes

java files at app/controllers/

HTML files at app/views/ (they should be named as `filename.scala.html`)

- I've included the dependencies for the Bootstrap UI form widgets and H2 in memory database. See the docs for how to use those.
- I've noticed trouble if I don't `sbt clean` periodically, so make sure to try that if there is a problem.

Docs

A big part of this assignment will be learning a new framework and API. The following are just a starting point, and StackOverflow is my other suggestion. The TAs will be available to help answer questions; don't get hung up on silly bugs without asking for help. * Overview and tutorial * Testing * UI Templates with Twirl * A useful set of examples (esp for REST APIs!)

Due

- We will mark the last commit made before **Dec 11 at 11:59pm**. If that last commit was a mess, let us know. Make sure your code compiles!
- By midnight Dec 14, submit your team peer review form via the webapp (link omitted to prevent spam). *You must add text that explains your rating if it is not Very Good or Excellent.*