Expeditors Backend Academy Labs

Introduction

This document contains the labs for the Expeditors Backend Academy.

The instructions are split up into **Classwork** and **Homework**. The Classwork we will do in class together. You should do the Homework in the **Homework** module. Create a new package for each week's homework if it makes sense. I will go through an example before we start.

Week 9 – Backend Capstone 1

Objectives

1. Put all the knowledge gained so far to design and build a Spring Boot REST application from scratch with minimal instructions.

Requirements

1. You have to build an application for an online Music Information business.
2. The application consists of two independent Spring Boot services:
   a. The Track Service allows users to get information about music tracks and their respective artists.
   b. The other service is a Pricing Service which returns a price for a track.
3. Both services should be exposed with REST apis
4. To make it easier, you might want to work on it in incremental steps.

Step 1: Track Service. The Track Service delivers information about Music tracks and associated Artists.

1. A Track should have at least the following properties. Add any others that you think are appropriate.
   a. Title
   b. Album
   c. One or more Artists
   d. Issue Date
   e. Duration
   f. Media Type. Must be a fixed set. e.g.
      1. Ogg

2. Mp3
3. Flac
4. Wav

2. Artists can have whatever properties you think they should have.

3. One issue your application should handle is that available information about both Tracks and Artists might be incomplete. For example, for one Track you might have only the Title and Duration, and for another you might have only the Title and Issue Date. Similarly with Artists.

4. Functionality
   a. For Tracks:
      1. Basic Create, Update, Delete functionality
      2. Get a Track by id
      3. Get Tracks with a specific media type
      4. Get Tracks for a particular year
      5. Get Artists for a particular Track
      6. Get Tracks longer/shorter/equal to a specific duration

   b. For Artists
      1. Basic Create, Update, Delete functionality
      2. Get an Artist by id
      3. Get Artists by name
      4. Get Tracks for a particular Artist
      5. Get all Artists

Step 2: Pricing Service

1. The Pricing Service returns the current price for a track. In this universe, prices change continuously, so it is important to get the current price every time a Track is requested.
2. The price is **not** a persistent property of the Track. It's value can change at any instant.
3. The Track Service should make a call to the Pricing Service to get the current price a track. It needs to do this **every time** it delivers a track since prices are continuously changing.
4. You can make up prices any way you want. Phases of the moon, the inclination of Saturn, the beating wings of a butterfly in Kuala Lumpur. Or, ThreadLocalRandom could be useful here too.

5. In the initial stages you could implement it as a dummy Java service.
6. But it should eventually be implemented as an independent service that is accessed with a REST call.

Extra Credit

1. Containerize the pricing service application using podman.

Tasks

1. Design the application
   a. You will be split up into teams to do this part collaboratively.
   b. Everyone on the team should have a good understanding of why any particular design decision has been made
2. Build the application
   a. You will build the application individually, to give everyone a chance to work on all the pieces.
   b. But you will have a quick daily meeting with your team every morning to do a status review of where everyone is and talk through any issues or blockers.

Deliverables

1. A working Spring Boot REST application
2. A suite of JUnit tests that provides as large coverage as possible. Aim for the high 80's, specially for controller and service code.
3. A presentation of the final project