Kotlin and Data Classes with Spring Boot and Dynamo DB

Repository for Code Reference 🔍

https://github.com/bexway/kotlin-spring-dynamo-example

Self Background

- Name: Bex
- Pronouns: they/them/theirs
- Job: Software Engineer in Data Services at ShopRunner
  - ShopRunner: Ecommerce start-up connecting top retailers with great customers, and providing free two day shipping
- We use Kotlin for data processing and web services for data APIs

SHOPRUNNER

Quick Intro to DynamoDB

- AWS NoSQL DB
  - More letters: Non-relational schema-less cloud database
- Dynamo table stores objects (rows) with attributes (columns)
  - Not all rows have the same attributes
  - Only requirement is that the table's Key attribute(s) must be provided
- Table keys specify how to organize the data to make searching fast
- For more, look up the AWS DynamoDB Developer Guide
- Run locally with Docker, and populate with AWS Command Line

What does Dynamo Data Look Like? 📊

- Json (kinda)
- Typed (“M”, “S”, “L”, “NULL”)
  - But schema-less!
- For DML (Data Manipulation Language)
  - Table Name
  - Request Type
  - Data to handle

Benefits of Kotlin with Spring/Dynamo

- Significant reduction in Spring boilerplate
- Read maps (even nested maps!) directly into data classes
  - Even if type conversion is needed!
- Null-safety means clearly signaling nullable fields
  - BUT be careful: some defaults are needed
  - if it is null in Dynamo it will throw an IllegalArgumentException

Describe a Table ┻━━┻︵ヽ(° -°ヽ)

- The properties on the Spring class determine what's read from the table
- If the table does not exist, writing a class with @DynamoDBTable annotation can create it
  - Specify keys (mandatory), and attributes (optional)
  - Auto-generation can be disabled in application.properties
- If your table does exist, write your table based on what you will query with and read from the table

JPA Provider: Spring-Data-DynamoDB 🔮

- Java Persistence API brings in functions for database querying
- Spring-Data-DynamoDB library
  - https://github.com/derjust/spring-data-dynamodb
- Implements Create-Read-Update-Delete methods, Spring annotation integration, and dynamic query generation
  - Dynamic query generation makes a method from a function name

DynamoDB Repository

- Required:
  - @EnableScan
  - Functions specifying query terms
- Not required
  - Function bodies
  - Functions querying by id

New Route 🗺️

- Include Repository in controller arguments
  - Dependency Injection handles the rest
- Repository handles query

```
@GetMapping( ...value: "/user/{firstName}")
fun user(@PathVariable firstName: String): List<User> =
    userRepository.findByFirstName(firstName)
```

Composite Key 🔑🗝️

- Dynamo DB has two key types: Hash/Partition keys, and Range/Sort keys
- Though two columns are used, querying or deleting by id expects only one id, not two
- Use a separate class to combine both Keys into one Composite Key
- Getters and Setters still needed on Table class 😞😖😭

Next, the Title 📜

- When adding the title, we want to bring in all three pieces
  - Map within a map

Option 1: Change type

```
@DynamoDBAttribute(attributeName = "title")
var userTitle: Map<String, Map<String, String>>? = null
```

Option 2 (cooler option 🕶️): Create data class

- Multiple types allowed
- No getters
- No setters
- No methods
- No problems 😎

```
@DynamoDBDocument
data class UserTitle(
    @DynamoDBAttribute
    var intro: Map<String, String>? = null,
    @DynamoDBAttribute
    var prefix: String? = null,
    @DynamoDBAttribute
    var suffix: String? = null
)
```

Plugs right into table attribute 

- Table-level attribute specifies Object attribute
  - title
- Data class arguments specify Map keys
  - intro, prefix, suffix

Type Conversion 📏

- Convert types when saving/loading
- Includes custom conversion
  - E.g. Json parsing, string splitting, date converting
- Marshaller is deprecated, converter is what the cool kids use
- Json strings? Number Strings? U got it!!! 👏👏👏

🎉 Thanks! 🎉