## Spring Boot with AWS Dynamo DB Example

**IAM:**

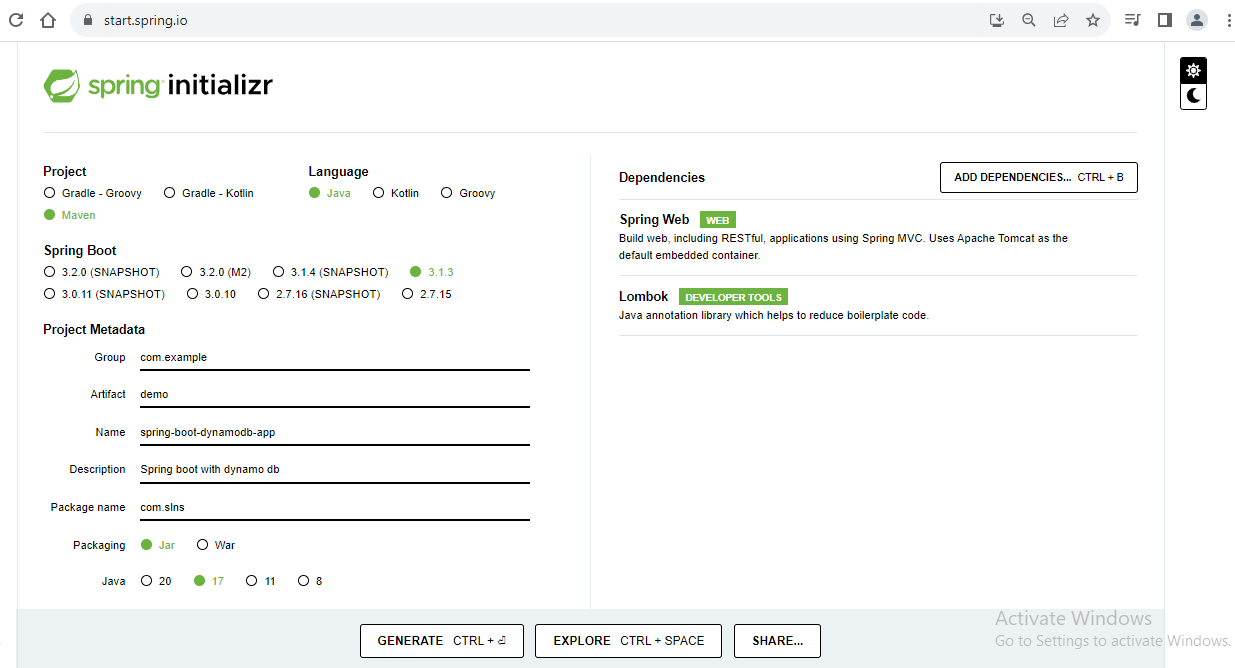
**Create Group**

* Login into AWS console
* Search for IAM service
* Click on groups
* Click on Create new group
* Group Name : **dynamodbdev**
* Click on Next step
* Attach policy to group : **AmazonDynamoDBFullAccess**
* Click on Next Step and Create Group

**Create User:**

* Click on User
* And Click on **Add user**
* User name: dynamodb-rami
* And click on **Programmatic Access** for getting secret key and access key.
* And click on Next
* Click on Add users to group
* Select the radio button for group **dynamodbdev**
* Click on next, next and click on **Create user**
* Once user is created you will the Secret key and access key in .csv format download it for future reference

**Create Spring boot Application:**

* [**https://start.spring.io/**](https://start.spring.io/)
* ****
* **Click on** Generate
* Go to download and extract it and open the intellij idea or any IDE and import the project into IDE

**Create Dynamo DB Table**

* Search for Dynamo db service in search bar
* Click Dynamo db service
* Click on Create Table
* Table name: employee
* Primary key : employee id , type String
* Click on Create button

**Create packages in spring-boot-dynamodb-app**

* **com.slns.controller**
* **com.slns.config**
* **com.slns.entity**
* **com.slns.repositroy**

**Dynamo DB Configuration file:**

package com.slns.config;  
  
import com.amazonaws.auth.AWSStaticCredentialsProvider;  
import com.amazonaws.auth.BasicAWSCredentials;  
import com.amazonaws.client.builder.AwsClientBuilder;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;  
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClientBuilder;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
  
@Configuration  
public class DynamoDBConfiguration {  
  
 @Bean  
 public DynamoDBMapper dynamoDBMapper(){  
 return new DynamoDBMapper(buildDynamoAmazonDBConfig());  
 }  
  
 private AmazonDynamoDB buildDynamoAmazonDBConfig() {  
 return AmazonDynamoDBClientBuilder  
 .*standard*()  
 .withEndpointConfiguration(  
 new AwsClientBuilder.EndpointConfiguration(  
 "dynamodb.us-east-1.amazonaws.com",  
 "us-east-1"  
 )  
 ).withCredentials(  
 new AWSStaticCredentialsProvider(  
 new BasicAWSCredentials(  
 "",  
 ""  
 )  
 )  
 ).build();  
 }  
}

**Entity classes:**

**Employee.java**

package com.slns.entity;  
  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBAttribute;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBTable;  
import lombok.AllArgsConstructor;  
import lombok.Data;  
import lombok.NoArgsConstructor;  
  
@Data  
@AllArgsConstructor  
@NoArgsConstructor  
@DynamoDBTable(tableName = "employee")  
public class Employee {  
 @DynamoDBAttribute(attributeName = "employeeId")  
 private String employeeId;  
 @DynamoDBAttribute  
 private String firstName;  
 @DynamoDBAttribute  
 private String lastName;  
 @DynamoDBAttribute  
 private String email;  
 @DynamoDBAttribute  
 private Department department;  
}

**Department.java**

@Data  
@AllArgsConstructor  
@NoArgsConstructor  
@DynamoDBDocument  
public class Department {  
 @DynamoDBAttribute  
 private String departmentName;  
 @DynamoDBAttribute  
 private String departmentCode;  
}

**Note:** Here we annotated with **@DynamoDBDocument** annotation this means this is not an dynamo db table it is also attribute.

**EmployeeRepository.java**

package com.slns.repository;  
  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBMapper;  
import com.amazonaws.services.dynamodbv2.datamodeling.DynamoDBSaveExpression;  
import com.amazonaws.services.dynamodbv2.model.AttributeValue;  
import com.amazonaws.services.dynamodbv2.model.ExpectedAttributeValue;  
import com.slns.entity.Employee;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public class EmployeeRepository {  
  
 @Autowired  
 private DynamoDBMapper dynamoDBMapper;  
  
 public Employee save(Employee employee){  
 this.dynamoDBMapper.save(employee);  
 return employee;  
 }  
  
 public Employee getEmployee(String employeeId){  
 return this.dynamoDBMapper.load(Employee.class, employeeId);  
 }  
  
 public String updateEmployee(String employeeId){  
 Employee employee1 = dynamoDBMapper.load(Employee.class, employeeId);  
 employee1.setEmail("Ricky@gmail.com");  
 dynamoDBMapper.save(employee1);  
 return "Employee updated";  
 }  
  
 //Another example of update employee in dynamo db  
 //Update the employee object based on matching condition  
 //It is quiet similar to update emp set sal = 10000 where sal < 20000;  
 public String updateEmployeeBasedONCondition(Employee employee){  
 dynamoDBMapper.save(employee, new DynamoDBSaveExpression()  
 .withExpectedEntry("employeeId",  
 new ExpectedAttributeValue(  
 new AttributeValue()  
 .withS(employee.getEmployeeId())  
 )));  
 return "Employee updated";  
 }  
  
 public String deleteEmployee(String employeeId){  
 Employee employee = dynamoDBMapper.load(Employee.class, employeeId);  
 dynamoDBMapper.delete(employee);  
 return "Employee Deleted";  
 }  
  
}

**EmployeeController.java**

package com.slns.controller;  
  
import com.slns.entity.Employee;  
import com.slns.repository.EmployeeRepository;  
import lombok.extern.slf4j.Slf4j;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
  
@Slf4j  
@RestController("/v1")  
public class EmployeeController {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @GetMapping("/employee/{employeeId}")  
 public ResponseEntity<Employee> getEmployeeDetails(@PathVariable("employeeId") String employeeId){  
 *log*.info("Started fetching employee details for employeeId {}", employeeId);  
 Employee employee = employeeRepository.getEmployee(employeeId);  
 *log*.info("Fetched employee details for employeeId {}", employeeId);  
 return ResponseEntity.*status*(HttpStatus.*OK*).body(employee);  
 }  
  
 @PostMapping("/employee")  
 public ResponseEntity<Employee> saveEmployeeDetails(@RequestBody Employee employee){  
 *log*.info("Saving Employee Details");  
 employeeRepository.save(employee);  
 *log*.info("Saved employee Details in Dynamo DD");  
 return ResponseEntity.*status*(HttpStatus.*CREATED*).body(employee);  
 }  
  
 @PutMapping("/employee")  
 public ResponseEntity<String> getEmployeeDetails(@RequestBody Employee employee){  
 *log*.info("Before updating employee details for employeeId {} ", employeeId);  
 String status = employeeRepository.updateEmployeeBasedONCondition(employee);  
 *log*.info("After updating employee details for employeeId {}", employeeId);  
 return ResponseEntity.*status*(HttpStatus.*OK*).body(status);  
 }  
  
 @DeleteMapping("/employee")  
 public ResponseEntity<String> deleteEmployeeDetails(@RequestHeader("employeeId") String employeeId){  
 *log*.info("Before Deleting employee details for employeeId {} ", employeeId);  
 String status = employeeRepository.deleteEmployee(employeeId);  
 *log*.info("After deleting employee details for employeeId {}", employeeId);  
 return ResponseEntity.*status*(HttpStatus.*OK*).body(status);  
 }  
  
  
}