HU Extension Assignment 06 E-90 Cloud Computing

Handed out: 10/10/2015 Due by 11:59 PM EST on Friday, 10/16/2015

Place all of your narratives and illustrations in a single Word or PDF document named E90_LastNameFirstNameHW06.docx [.pdf]. Use this assignment as the initial template. Add your steps and your code below problem statements used for that problem. Upload your homework file and your working code (e.g., filename.java) into your Assignment 6 folder. Do not include executables.

Problem 1.

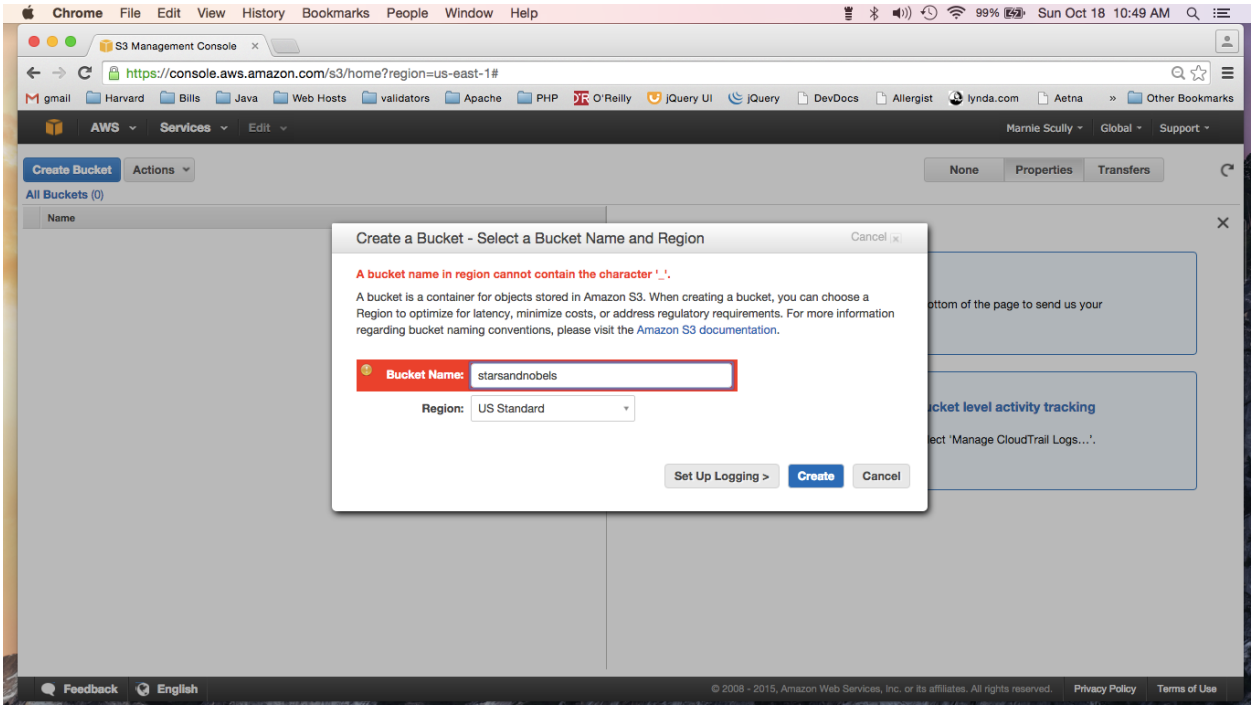
I. Manually populate an S3 bucket with Images and Resumes of movie stars and Nobel laureates:
   a. Create folder structures with two folders: **stars** and **nobels**. Inside either folder you should have subfolders **images** and **resumes**.
   b. From Google Images fetch images (pictures) of three movie stars and three Nobel winners. Produce 3+3 MS Word documents with bogus resumes for selected movie stars and Nobel winners on your own. The resumes should be very short; three sentences each.
   c. Upload those images and resume into your S3 bucket manually.
   d. Manually grant general public access to those images and resumes.

II. Create one SimpleDB domain and programmatically insert one row for each movie star and each Nobel laureates - write your program either in Java or .Net or your favorite language supported by AWS:
   a. Use their full names with hyphens between the first and the last name as the item keys.
   b. In rows for movie stars record full name, most popular movie, S3 URL of the picture of the star and S3 URL of his or her resume.
   c. Use those same attributes for the Nobel laureates and add the year they won the prize and the field of science in which they got the prize as two additional attributes.
   d. Demonstrate that you can change (correct) the year of one Nobel prize award, programmatically.
   e. Demonstrate that you can delete one movie star from SimpleDB domain programmatically.
   f. Capture the content of your database as displayed in the Database Development perspective.

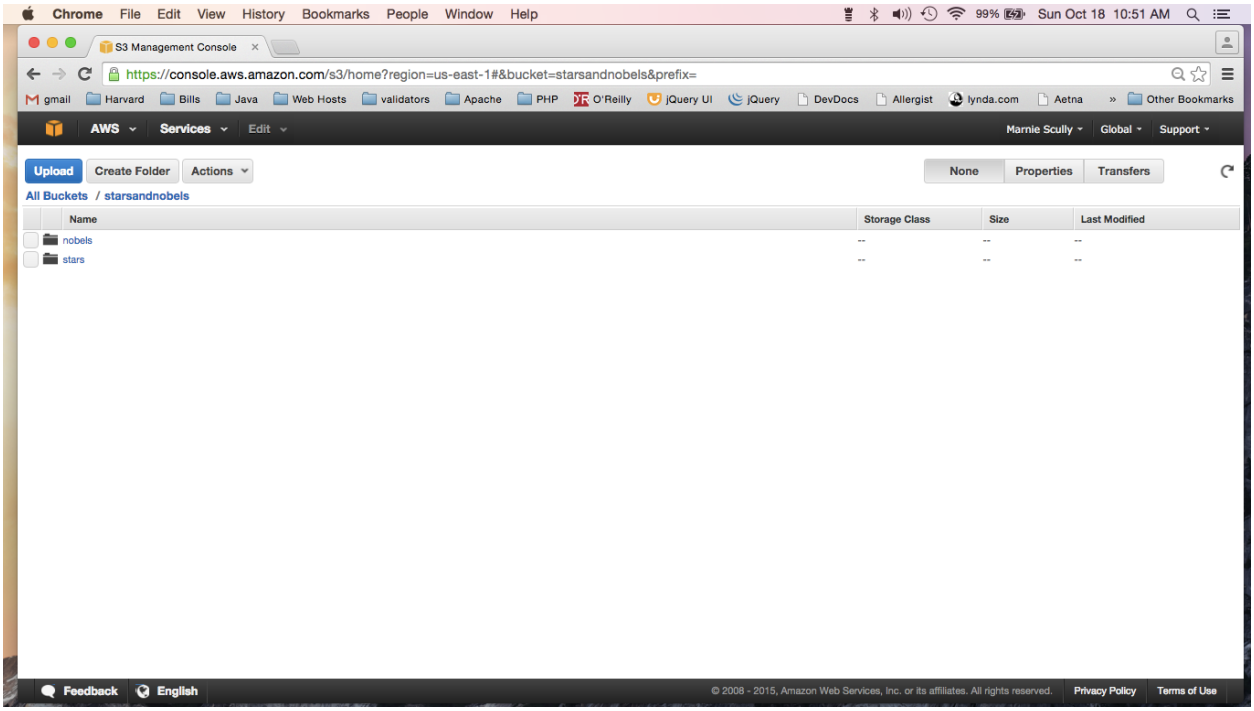
Provide working code and capture all stages of testing. Write your program either in Java or .Net or your favorite language supported by AWS. **[30 Points]**

HW6_ScullyMarnie

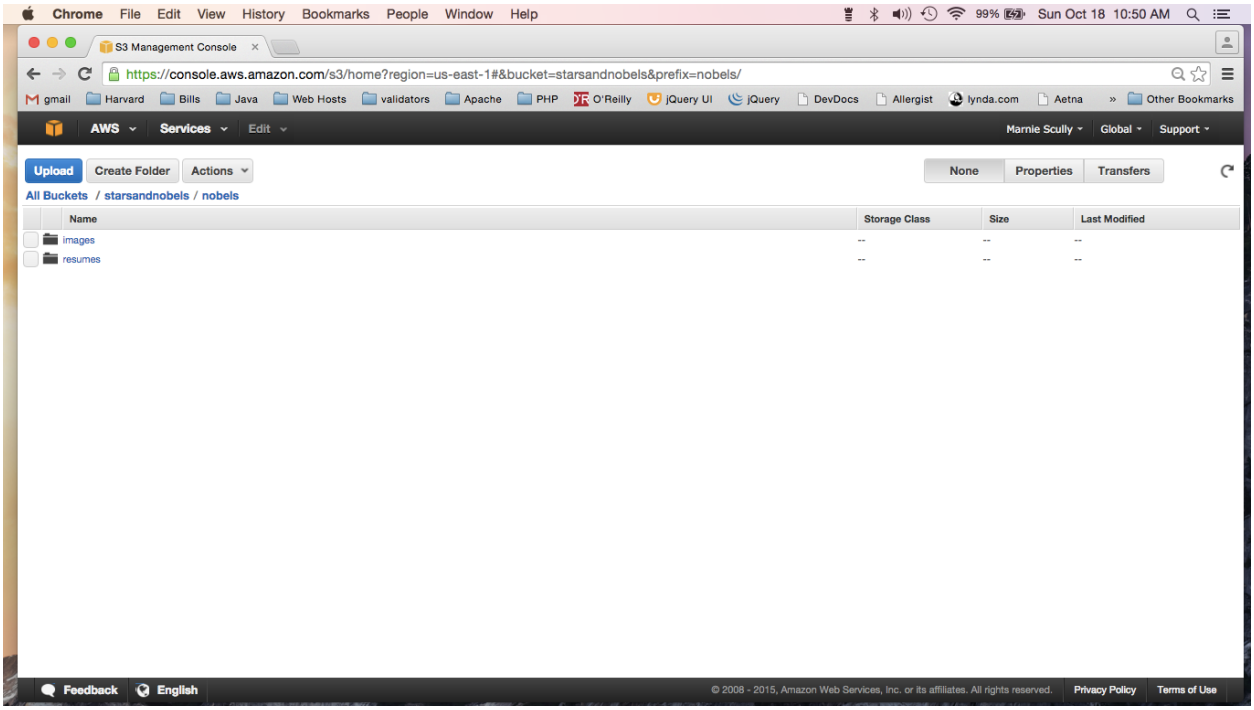
Create a S3 Bucket named starsandnobels



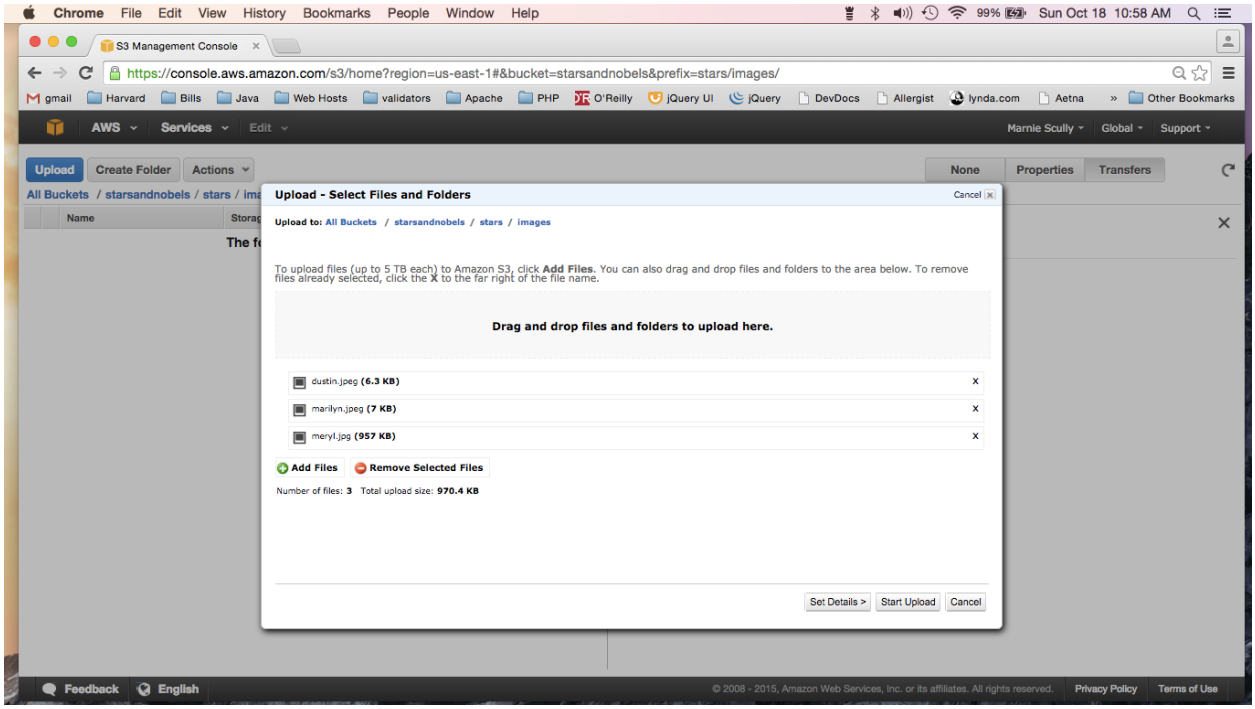
Create folders within bucket for stars, nobels, with subfolders images and resumes in each.



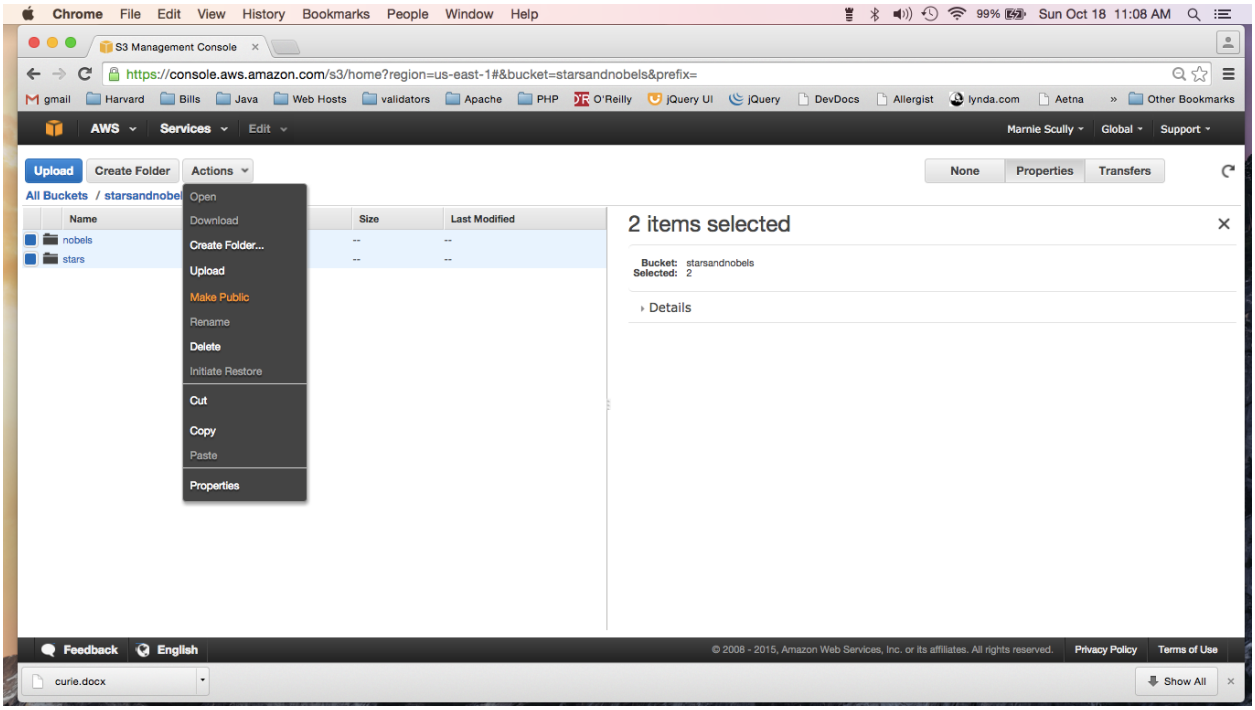
HW6_ScullyMarnie



Upload all images and resumes to proper folders



Grant Public access to the images and resumes within stars and novels



Create a SimpleDB domain

```
String myDomain = "starsandnobels";
        System.out.println("Creating domain called " + myDomain + ".\n");
        sdb.createDomain(new CreateDomainRequest(myDomain));
```

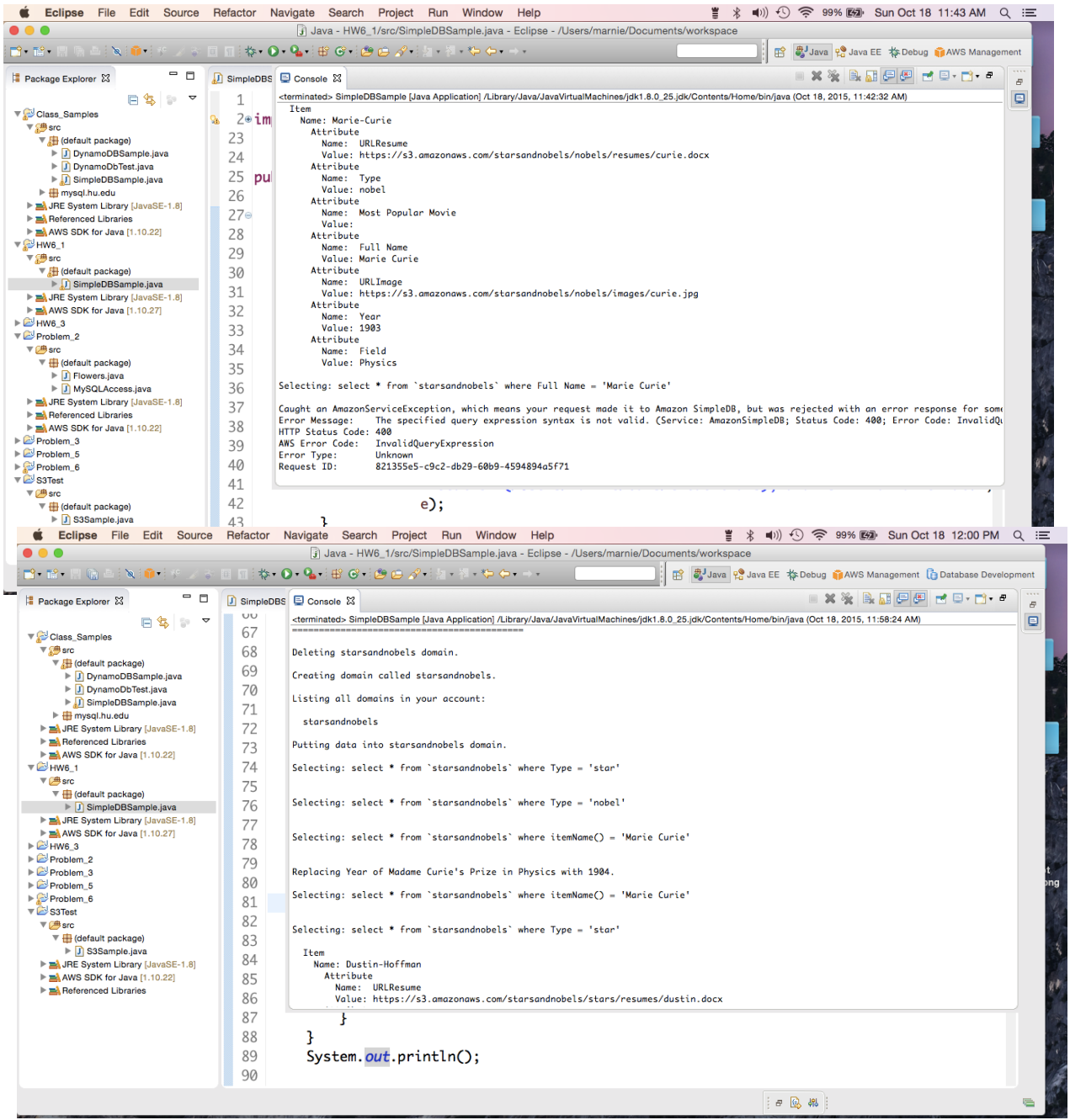
Insert a row for each star and laureate

```
/* Creates an array of SimpleDB ReplaceableItems populated with sample data. */
    private static List<ReplaceableItem> createSampleData() {
        List<ReplaceableItem> sampleData = new ArrayList<ReplaceableItem>();

        sampleData.add(new ReplaceableItem("Dustin-Hoffman").withAttributes(
                new ReplaceableAttribute("Full Name", "Dustin Hoffman", true),
                new ReplaceableAttribute("Type", "star", true),
                new ReplaceableAttribute("Most Popular Movie", "Tootsie", true),
                new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/
stars/images/dustin.jpeg", true),
                new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/
stars/resumes/dustin.docx", true)));
```

```
        sampleData.add(new ReplaceableItem("Marilyn-Monroe").withAttributes(
                new ReplaceableAttribute("Full Name", "Marilyn Monroe", true),
                new ReplaceableAttribute("Type", "star", true),
                new ReplaceableAttribute("Most Popular Movie", "Some Like it Hot", true),
                new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/
stars/images/marilyn.jpeg", true),
                new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/
stars/resumes/marilyn.docx", true)));

        sampleData.add(new ReplaceableItem("Meryl-Streep").withAttributes(
                new ReplaceableAttribute("Full Name", "Meryl Streep", true),
                new ReplaceableAttribute("Type", "star", true),
                new ReplaceableAttribute("Most Popular Movie", "Sophie's Choice", true),
                new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/
stars/images/meryl.jpeg", true),
                new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/
stars/resumes/meryl.docx", true)));

        sampleData.add(new ReplaceableItem("Marie-Curie").withAttributes(
                new ReplaceableAttribute("Full Name", "Marie Curie", true),
                new ReplaceableAttribute("Type", "nobel", true),
                new ReplaceableAttribute("Most Popular Movie", "", true), // the instructions said to
include this for the nobels too
                new ReplaceableAttribute("Year", "1903", true),
                new ReplaceableAttribute("Field", "Physics", true),
                new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/
nobels/images/curie.jpg", true),
                new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/
nobels/resumes/curie.docx", true)));

        sampleData.add(new ReplaceableItem("Eugene-Oneil").withAttributes(
                new ReplaceableAttribute("Full Name", "Eugene O'Neil", true),
                new ReplaceableAttribute("Type", "nobel", true),
                new ReplaceableAttribute("Most Popular Movie", "", true),
                new ReplaceableAttribute("Year", "1936", true),
                new ReplaceableAttribute("Field", "Literature", true),
                new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/
nobels/images/oneil.jpg", true),
                new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/
nobels/resumes/oneil.docx", true)));

        sampleData.add(new ReplaceableItem("George-Bernard-Shaw").withAttributes(
                new ReplaceableAttribute("Full Name", "George Bernard Shaw", true),
```

```
            new ReplaceableAttribute("Type", "nobel", true),
            new ReplaceableAttribute("Most Popular Movie", "", true),
            new ReplaceableAttribute("Year", "1925", true),
            new ReplaceableAttribute("Field", "Literature", true),
            new ReplaceableAttribute("URLImage", "https://s3.amazonaws.com/starsandnobels/
nobels/images/shaw.jpg", true),
            new ReplaceableAttribute("URLResume", "https://s3.amazonaws.com/starsandnobels/
nobels/resumes/shaw.docx", true)));

        return sampleData;
    }
```

Some testing as I developed



Demonstrate editing a value in an item and Demonstrate deleting an Item

(Output in Eclipse)

```
===========================================
Getting Started with Amazon SimpleDB
===========================================

Deleting starsandnobels domain.

Creating domain called starsandnobels.

Listing all domains in your account:

  starsandnobels

Putting data into starsandnobels domain.

Selecting: select * from `starsandnobels`

  Item
    Name: Dustin-Hoffman
      Attribute
        Name:  URLResume
        Value: https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx
      Attribute
        Name:  Type
        Value: star
      Attribute
        Name:  Most Popular Movie
        Value: Tootsie
      Attribute
        Name:  Full Name
        Value: Dustin Hoffman
      Attribute
        Name:  URLImage
        Value: https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg
  Item
    Name: Eugene-Oneil
      Attribute
        Name:  URLResume
```

Value: https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx
Attribute
Name: Type
Value: nobel
Attribute
Name: Most Popular Movie
Value:
Attribute
Name: Full Name
Value: Eugene O'Neil
Attribute
Name: URLImage
Value: https://s3.amazonaws.com/starsandnobels/nobels/images/oneil.jpg
Attribute
Name: Year
Value: 1936
Attribute
Name: Field
Value: Literature
Item
Name: George-Bernard-Shaw
Attribute
Name: URLResume
Value: https://s3.amazonaws.com/starsandnobels/nobels/resumes/shaw.docx
Attribute
Name: Type
Value: nobel
Attribute
Name: Most Popular Movie
Value:
Attribute
Name: Full Name
Value: George Bernard Shaw
Attribute
Name: URLImage
Value: https://s3.amazonaws.com/starsandnobels/nobels/images/shaw.jpg
Attribute
Name: Year
Value: 1925
Attribute
Name: Field
Value: Literature
Item

Name: Marie-Curie
Attribute
Name: URLResume
Value: https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx
Attribute
Name: Type
Value: nobel
Attribute
Name: Most Popular Movie
Value:
Attribute
Name: Full Name
Value: Marie Curie
Attribute
Name: URLImage
Value: https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpg
Attribute
Name: Year
Value: 1903
Attribute
Name: Field
Value: Physics
Item
Name: Marilyn-Monroe
Attribute
Name: URLResume
Value: https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx
Attribute
Name: Type
Value: star
Attribute
Name: Most Popular Movie
Value: Some Like it Hot
Attribute
Name: Full Name
Value: Marilyn Monroe
Attribute
Name: URLImage
Value: https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg
Item
Name: Meryl-Streep
Attribute
Name: URLResume

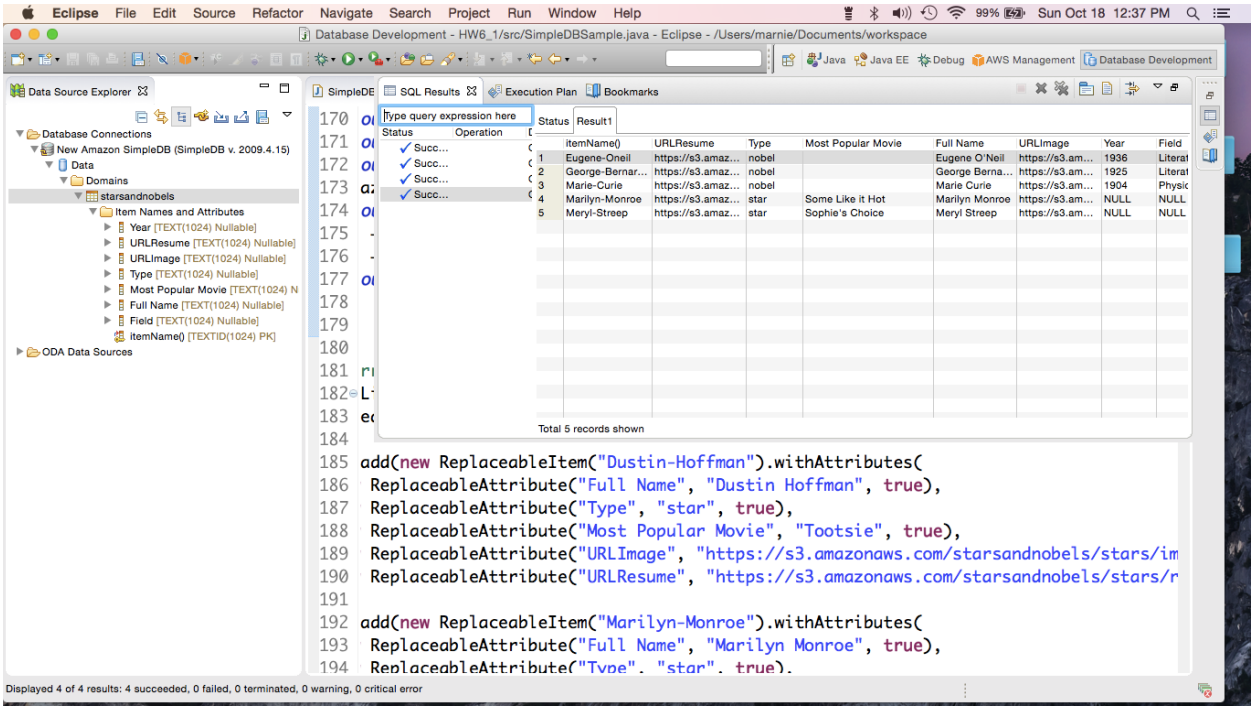
```
      Value: https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx
    Attribute
      Name:  Type
      Value: star
    Attribute
      Name:  Most Popular Movie
      Value: Sophie's Choice
    Attribute
      Name:  Full Name
      Value: Meryl Streep
    Attribute
      Name:  URLImage
      Value: https://s3.amazonaws.com/starsandnobels/stars/images/meryl.jpeg

Selecting: select * from `starsandnobels` where itemName() = 'Marie-Curie'

  Item
    Name: Marie-Curie
    Attribute
      Name:  URLResume
      Value: https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx
    Attribute
      Name:  Type
      Value: nobel
    Attribute
      Name:  Most Popular Movie
      Value:
    Attribute
      Name:  Full Name
      Value: Marie Curie
    Attribute
      Name:  URLImage
      Value: https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpg
    Attribute
      Name:  Year
      Value: 1903
    Attribute
      Name:  Field
      Value: Physics

Replacing Year of Madame Curie's Prize in Physics with 1904.

Selecting: select * from `starsandnobels` where itemName() = 'Marie-Curie'
```

```
Item
  Name: Marie-Curie
    Attribute
      Name:  URLResume
      Value: https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx
    Attribute
      Name:  Type
      Value: nobel
    Attribute
      Name:  Most Popular Movie
      Value:
    Attribute
      Name:  Full Name
      Value: Marie Curie
    Attribute
      Name:  URLImage
      Value: https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpg
    Attribute
      Name:  Year
      Value: 1904
    Attribute
      Name:  Field
      Value: Physics

Selecting: select * from `starsandnobels` where Type = 'star'

  Item
    Name: Dustin-Hoffman
    Attribute
      Name:  URLResume
      Value: https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx
    Attribute
      Name:  Type
      Value: star
    Attribute
      Name:  Most Popular Movie
      Value: Tootsie
    Attribute
      Name:  Full Name
      Value: Dustin Hoffman
    Attribute
      Name:  URLImage
```

```
      Value: https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg
  Item
    Name: Marilyn-Monroe
    Attribute
      Name:  URLResume
      Value: https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx
    Attribute
      Name:  Type
      Value: star
    Attribute
      Name:  Most Popular Movie
      Value: Some Like it Hot
    Attribute
      Name:  Full Name
      Value: Marilyn Monroe
    Attribute
      Name:  URLImage
      Value: https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg
  Item
    Name: Meryl-Streep
    Attribute
      Name:  URLResume
      Value: https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx
    Attribute
      Name:  Type
      Value: star
    Attribute
      Name:  Most Popular Movie
      Value: Sophie's Choice
    Attribute
      Name:  Full Name
      Value: Meryl Streep
    Attribute
      Name:  URLImage
      Value: https://s3.amazonaws.com/starsandnobels/stars/images/meryl.jpeg

Deleting Dustin Hoffman.

Selecting: select * from `starsandnobels` where Type = 'star'

  Item
    Name: Marilyn-Monroe
    Attribute
```

Name: URLResume
Value: https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx
Attribute
Name: Type
Value: star
Attribute
Name: Most Popular Movie
Value: Some Like it Hot
Attribute
Name: Full Name
Value: Marilyn Monroe
Attribute
Name: URLImage
Value: https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg
Item
Name: Meryl-Streep
Attribute
Name: URLResume
Value: https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx
Attribute
Name: Type
Value: star
Attribute
Name: Most Popular Movie
Value: Sophie's Choice
Attribute
Name: Full Name
Value: Meryl Streep
Attribute
Name: URLImage
Value: https://s3.amazonaws.com/starsandnobels/stars/images/meryl.jpeg

Database after changes in Database Development Perspective



Problem 2.

DynamoDB version. Use the same S3 bucket, images and resumes created in Problem 1. Use AWS DynamoDB Console for all the work

a) Create a new table, "CELEBRITIES_CONSOLE", where primary key is of a Hash type, and is constructed as "firstName" + "-" + "lastName". (for example, "Julia-Roberts")

b) Insert three rows for movie stars, with attributes: full name (= firstName + " " + lastName), most popular movie, S3 URL of the picture of the star and S3 URL of his or her resume

c) Insert three rows for Noble laureates, with attributes: full name (same format as above), S3 URL of the picture of the lauret, S3 URL of his or her resume, award year, science field in which they got the prize

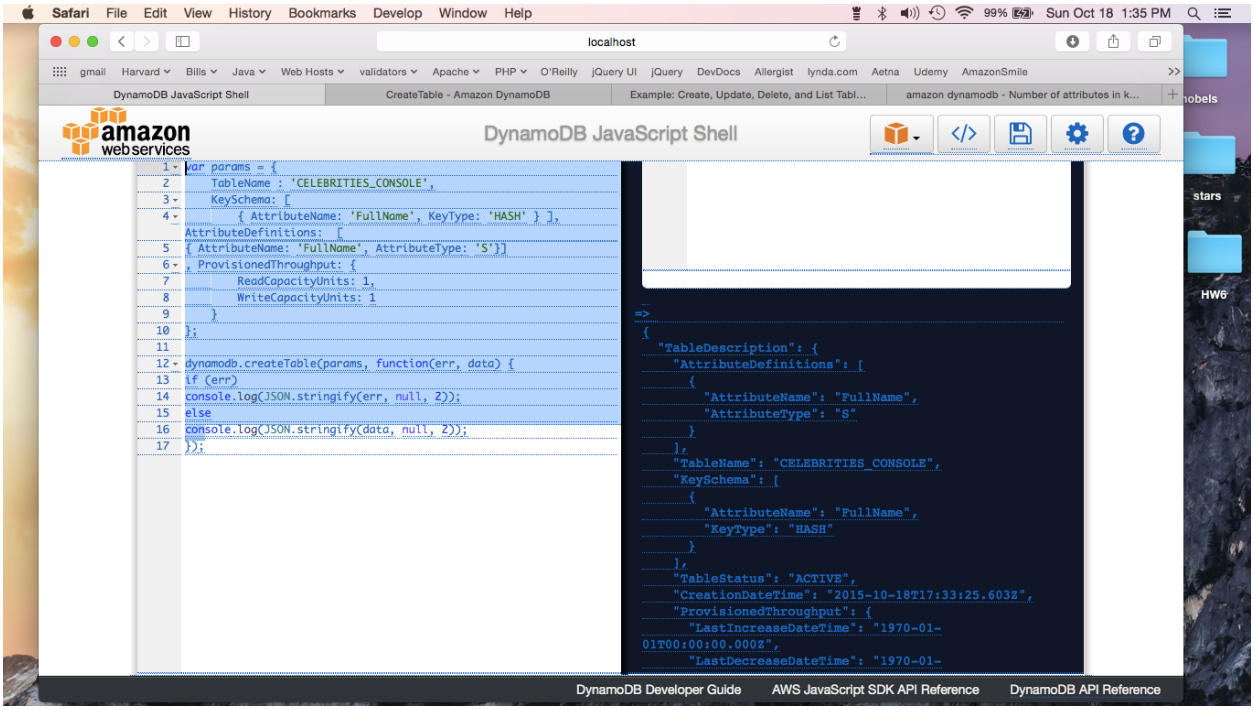
d) Explore the table in the Console and search by providing filters on ID - both scans and get by ID types of queries

e) Update an award year of one of the Nobles rows - demo that the data is updated. Capture all stages of testing. **[15 Points]**

HW6_ScullyMarnie

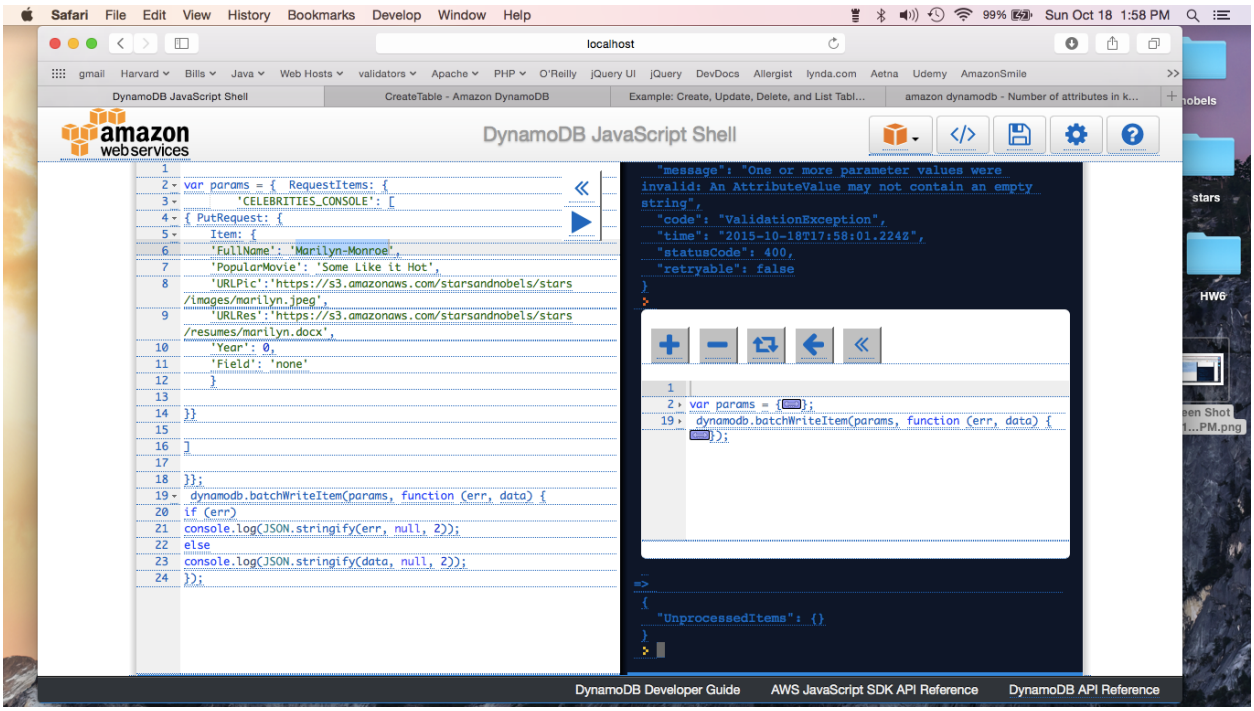
In DynamoDBLocal place this code and run. See results on right pane.

```
var params = {
   TableName : 'CELEBRITIES_CONSOLE',
   KeySchema: [
       { AttributeName: 'FullName', KeyType: 'HASH' } ], AttributeDefinitions:  [
{ AttributeName: 'FullName', AttributeType: 'S'}]
, ProvisionedThroughput: {
       ReadCapacityUnits: 1,
       WriteCapacityUnits: 1
   }
};

dynamodb.createTable(params, function(err, data) {
if (err)
console.log(JSON.stringify(err, null, 2));
else
console.log(JSON.stringify(data, null, 2));
});
```

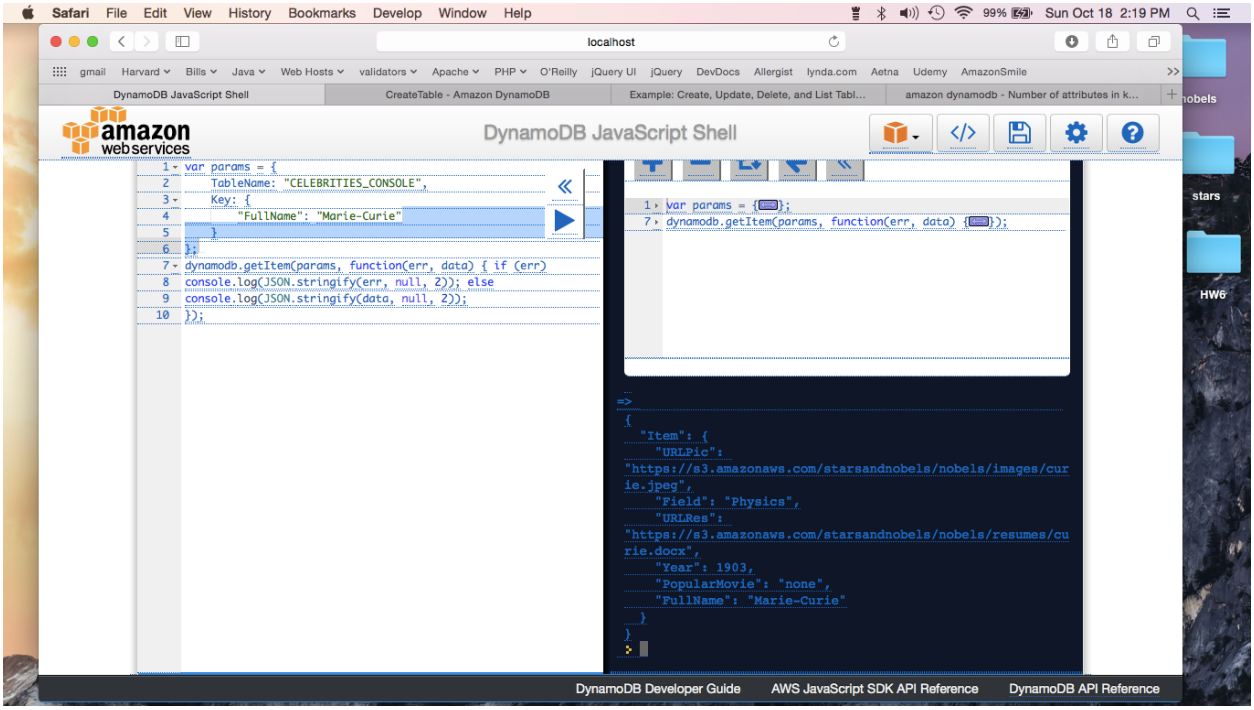


HW6_ScullyMarnie

Insert All items in DB

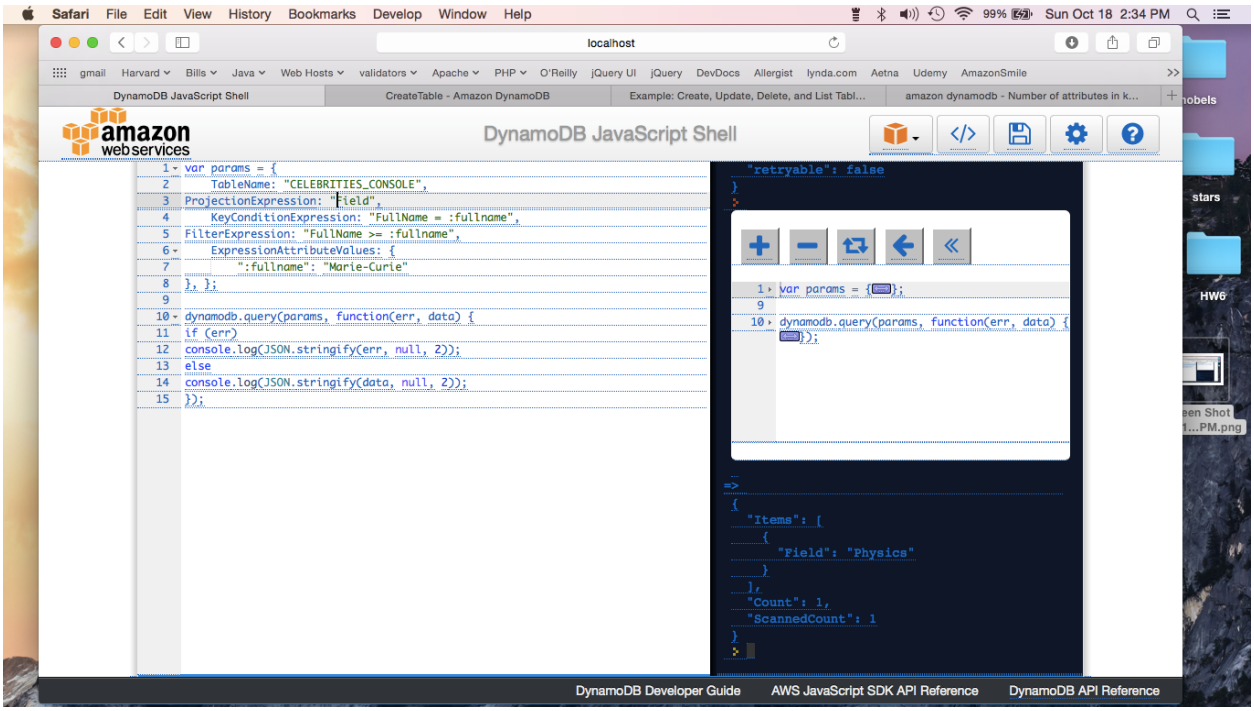


Query Table by PrimaryKey

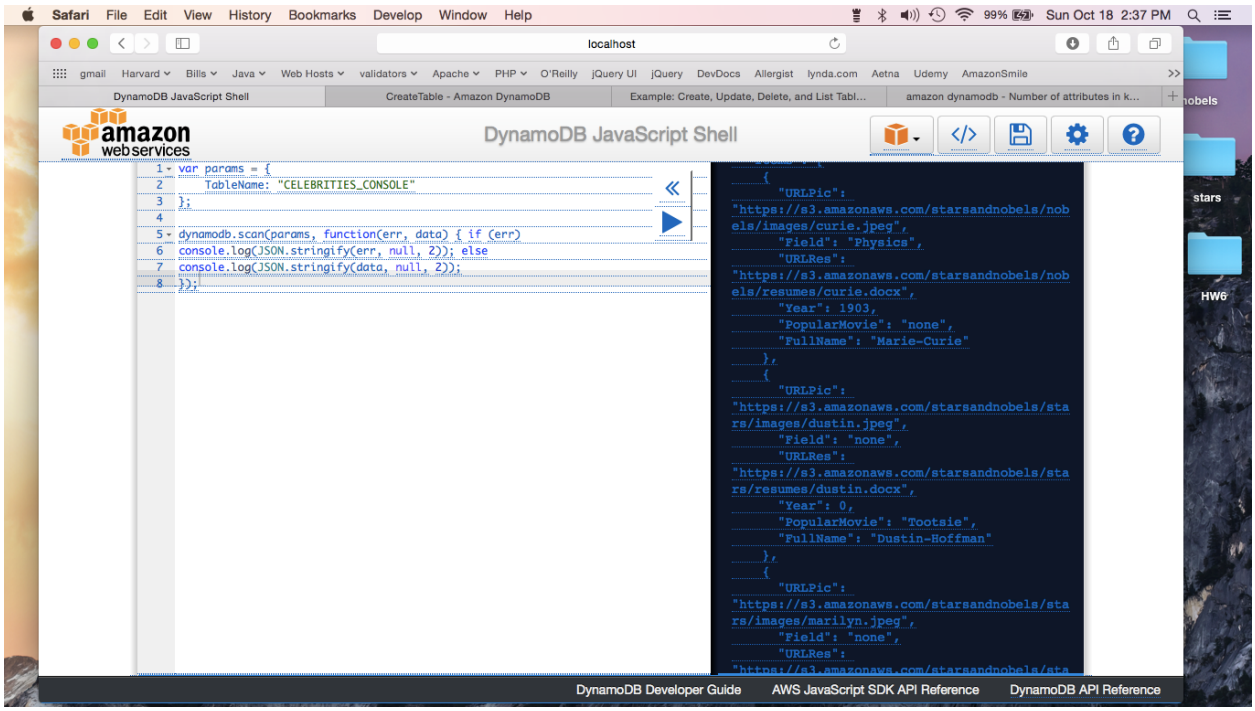


HW6_ScullyMarnie

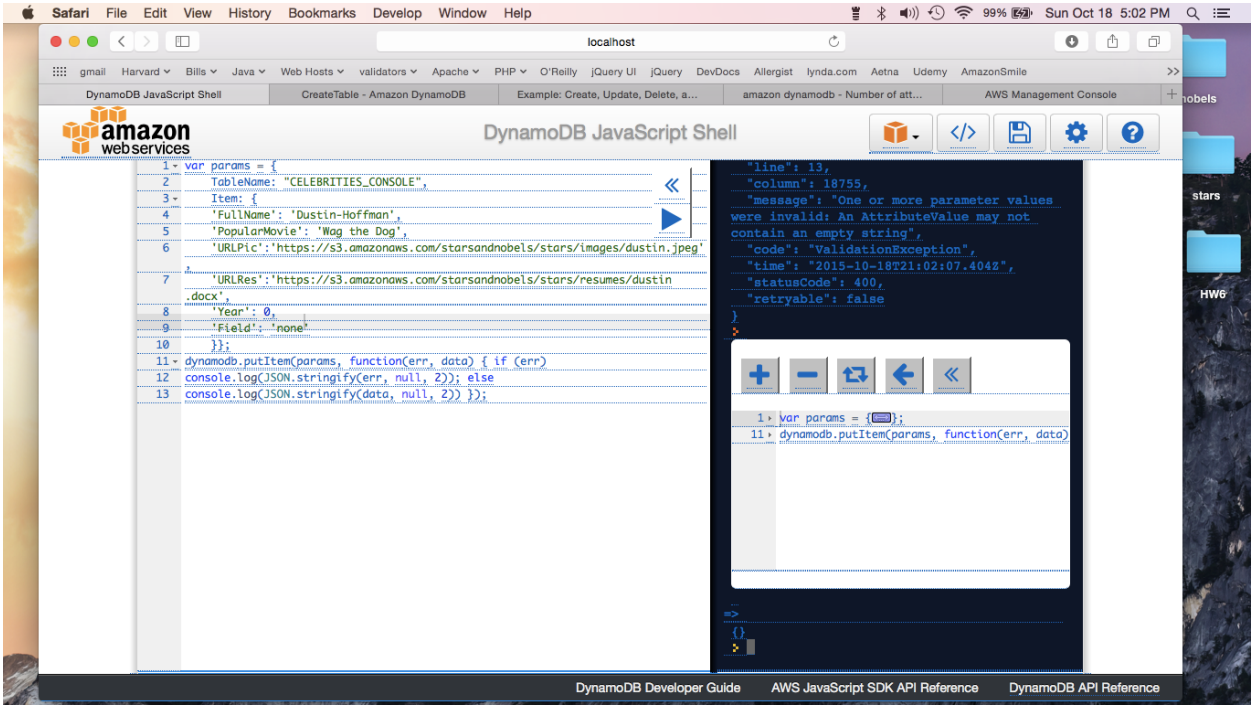
Query with filter



Query using Scan

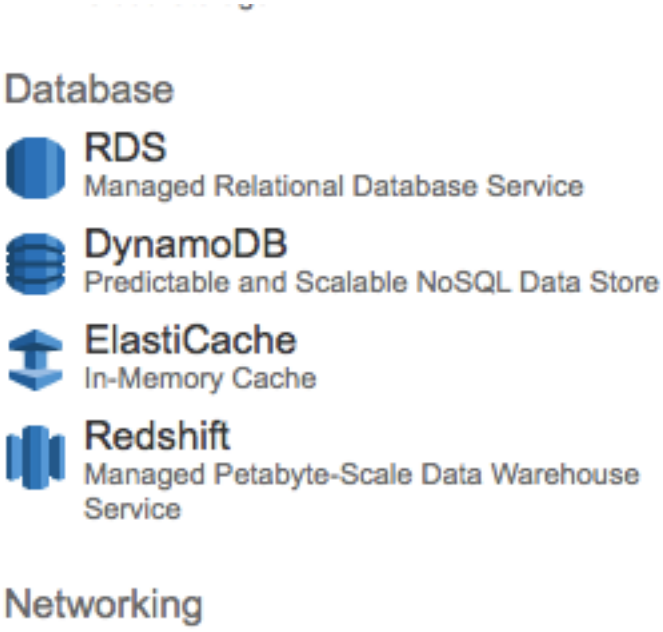


Demonstrate the ability to update a row

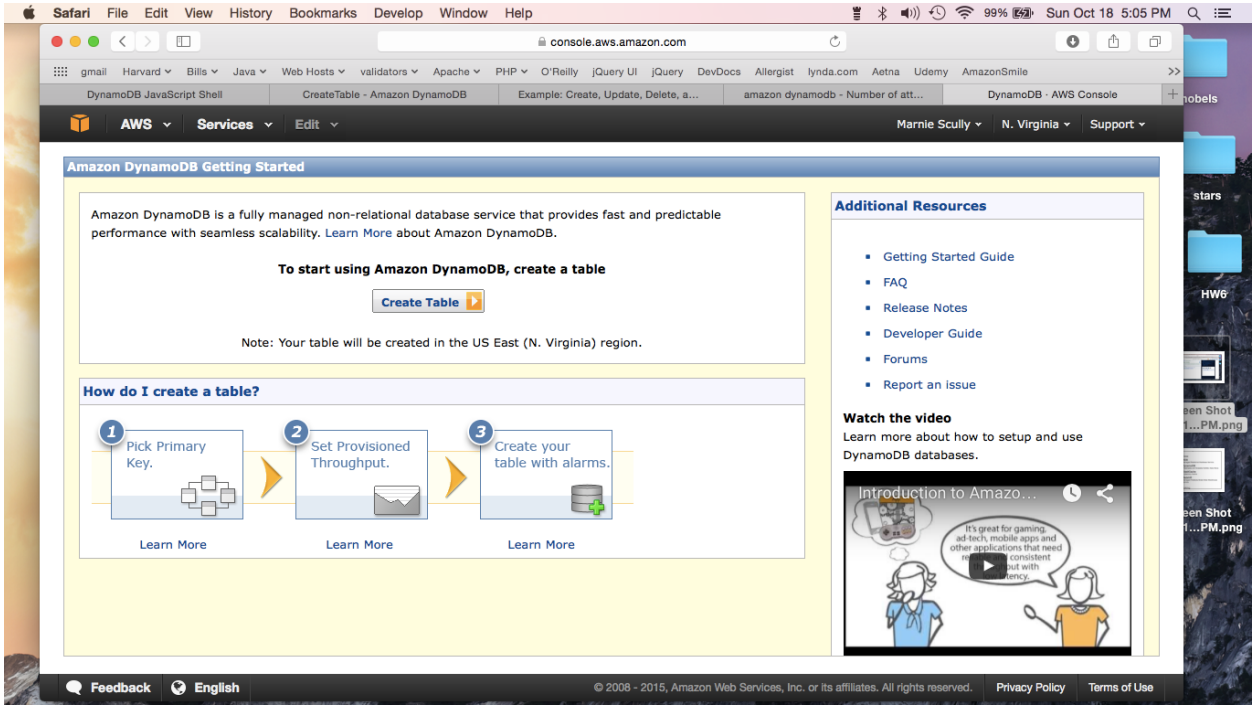


And Now the same problem with the AWS DynamoDB console (because I'm not sure which one you want honestly)

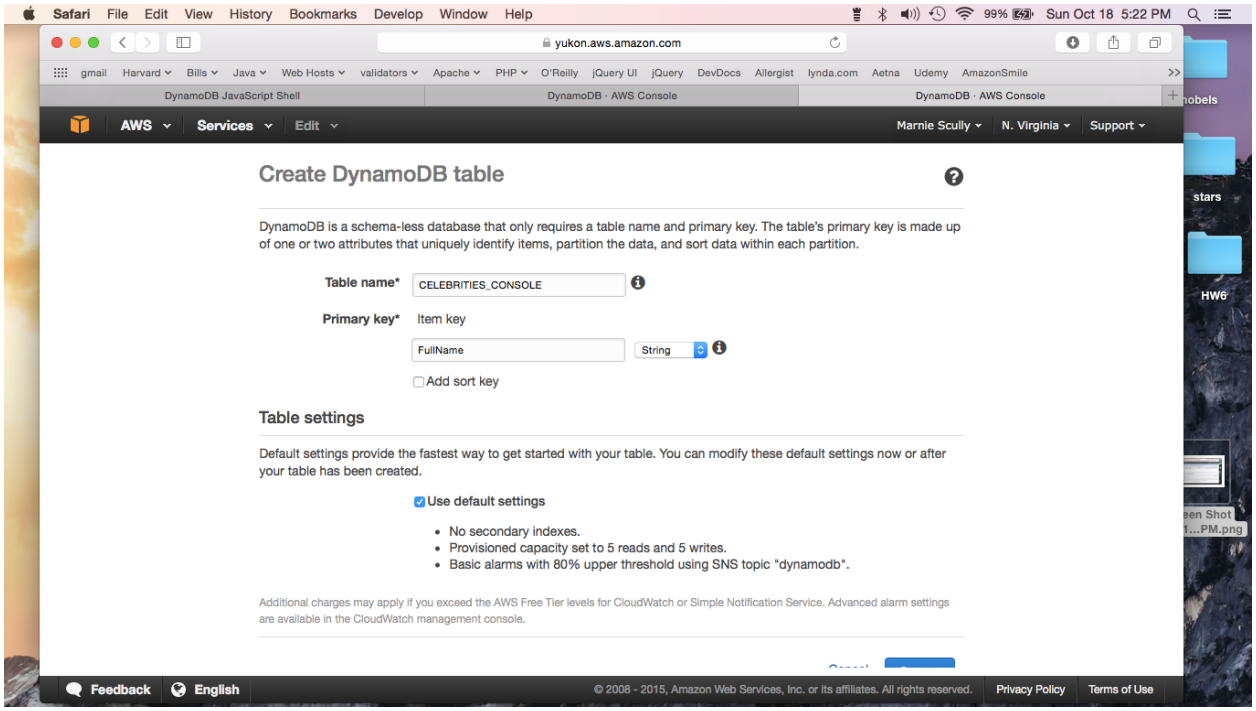
Choose DynamoDB



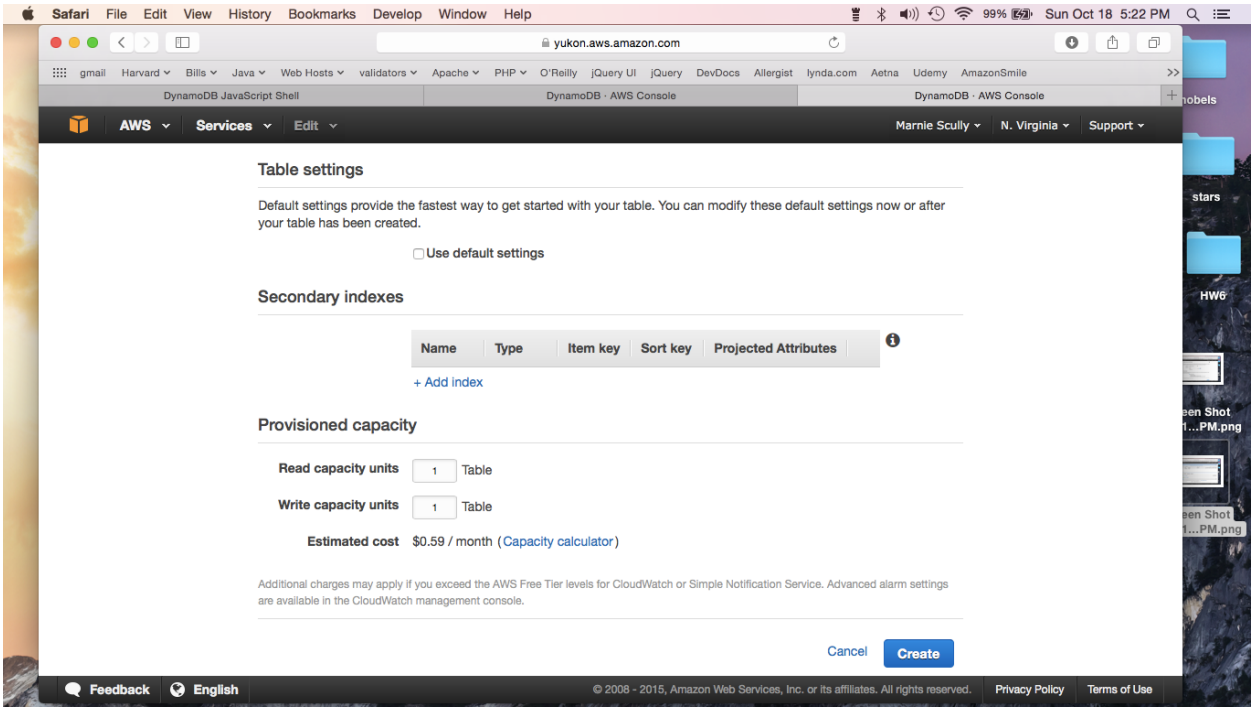
Click “Create Table”



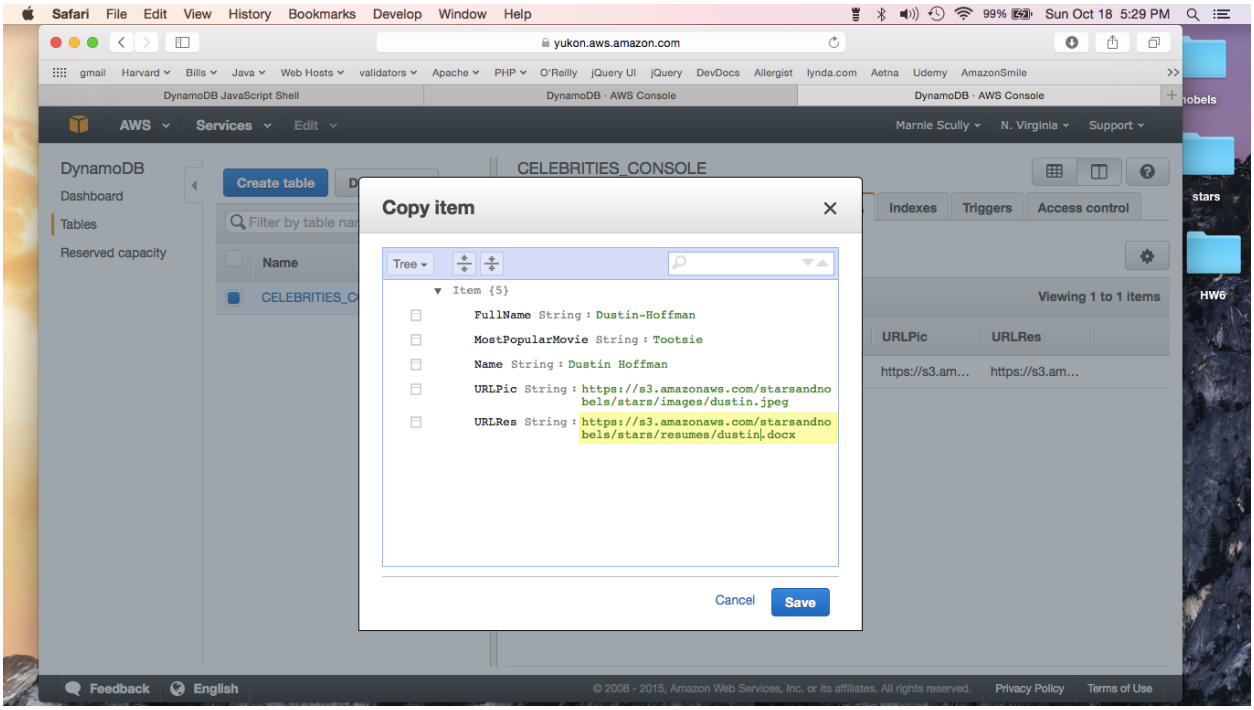
Name the table CELEBRITIES_CONSOLE with a hash type primary key for the full name of the celebrity



HW6_ScullyMarnie

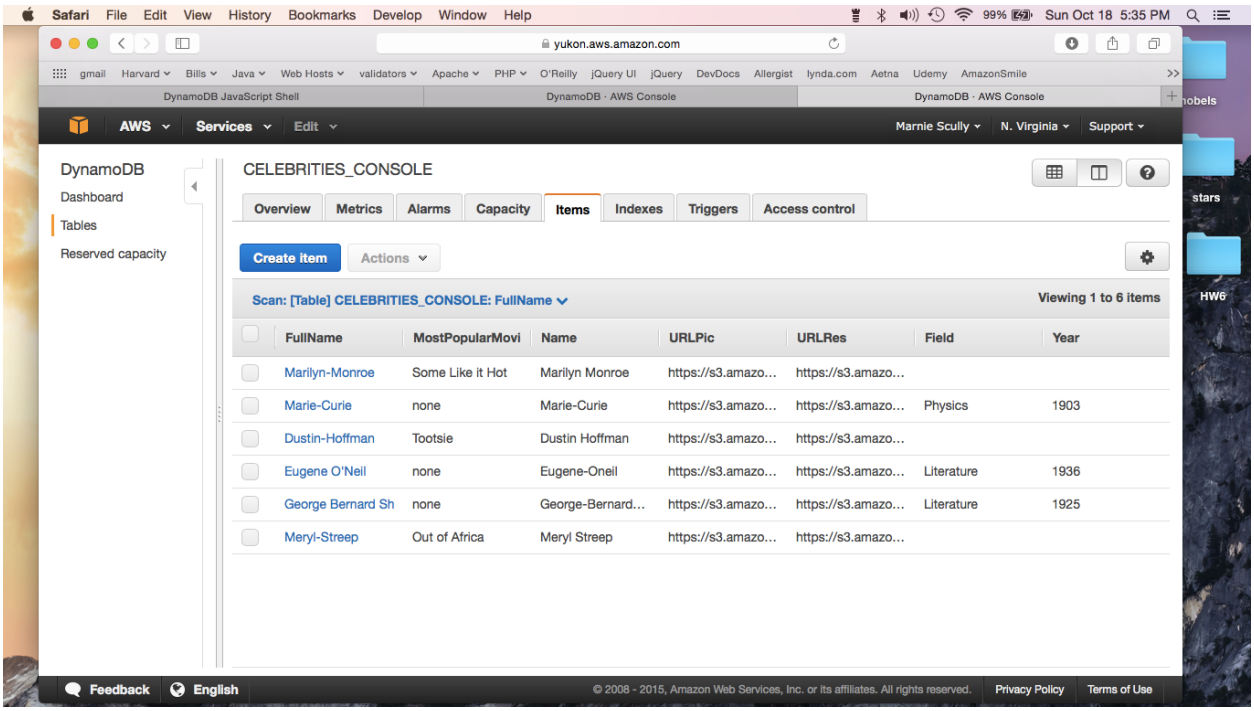


Create the items in the DB



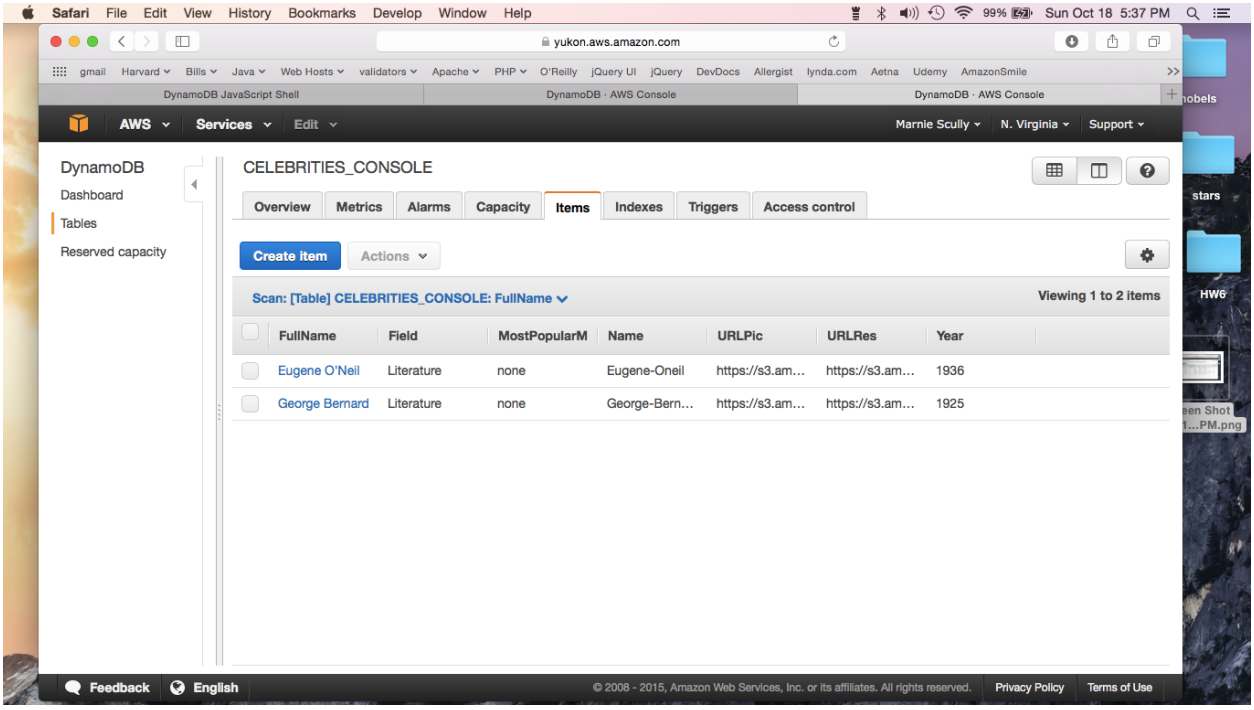
HW6_ScullyMarnie

After inserting all records explore the table in the console



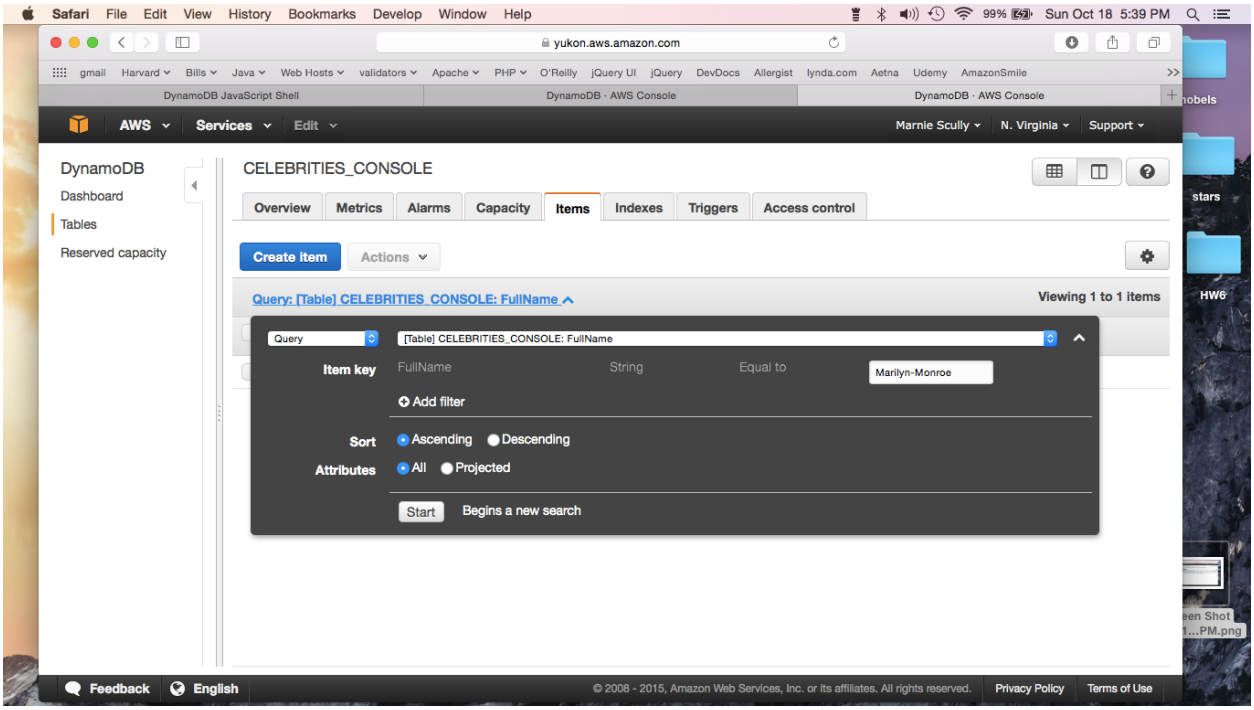
Search by filters on ID

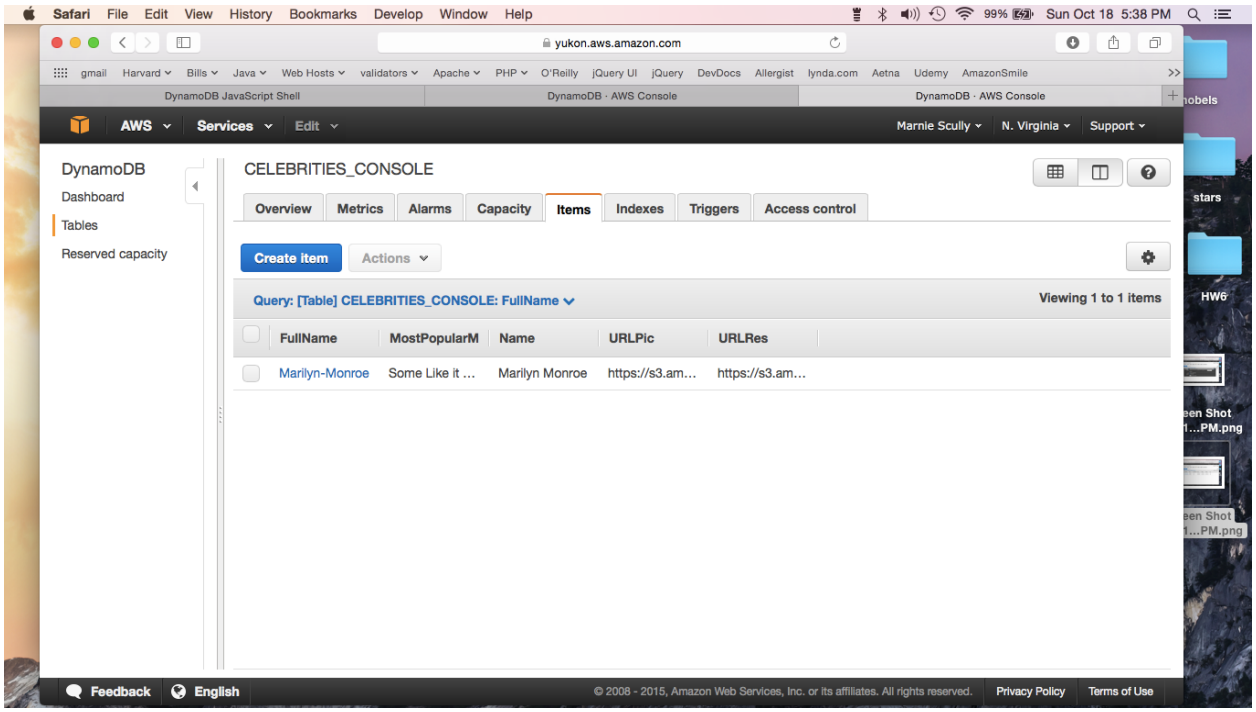
Scan filter by Field = “Literature”



HW6_ScullyMarnie

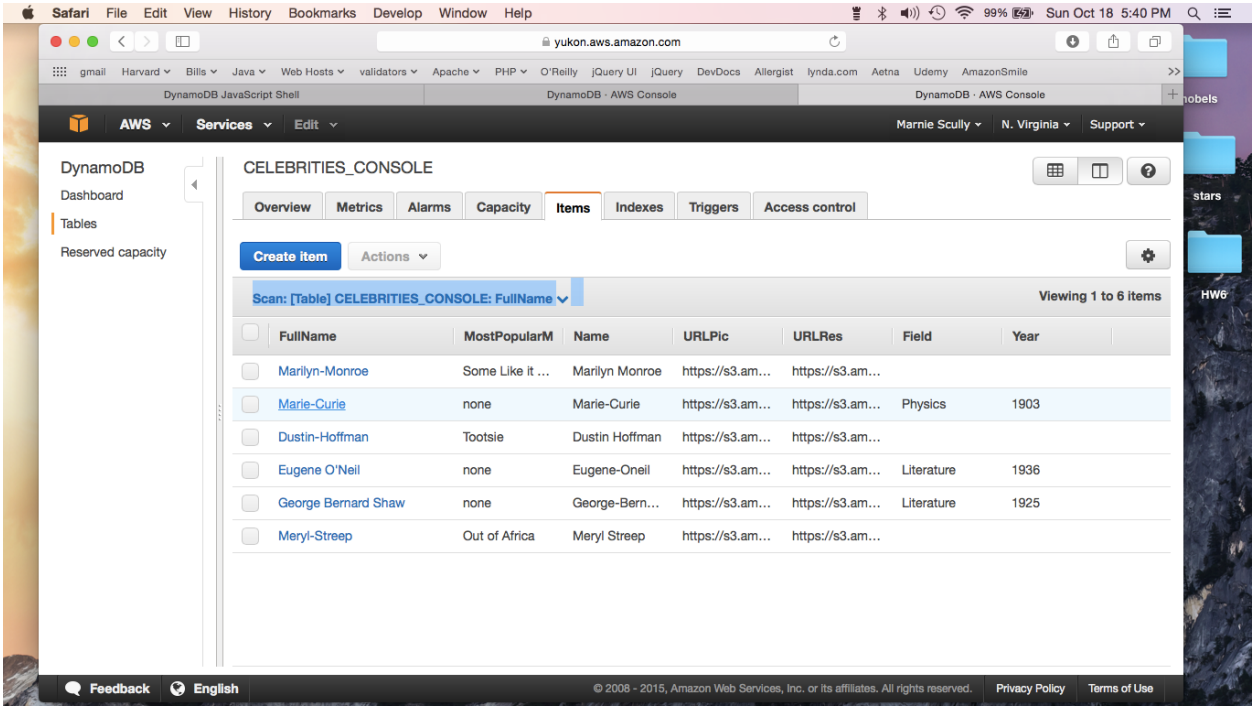
Query get by ID= “Marilyn-Monroe”



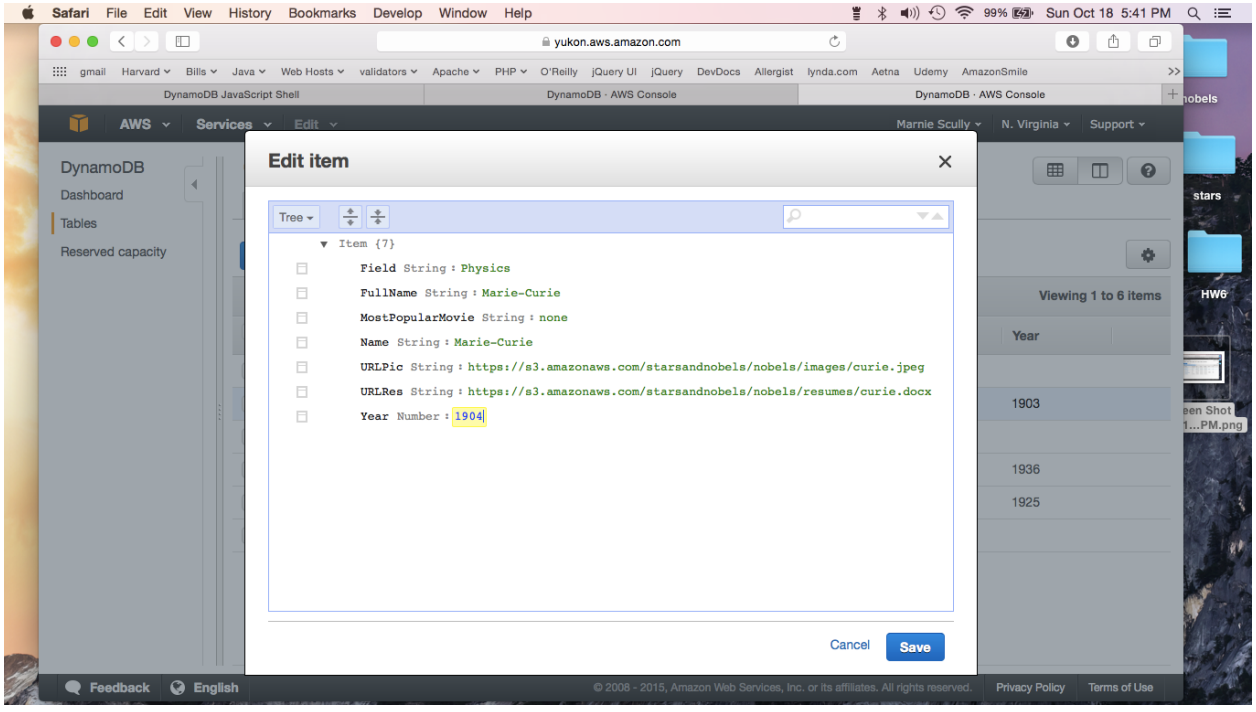


HW6_ScullyMarnie

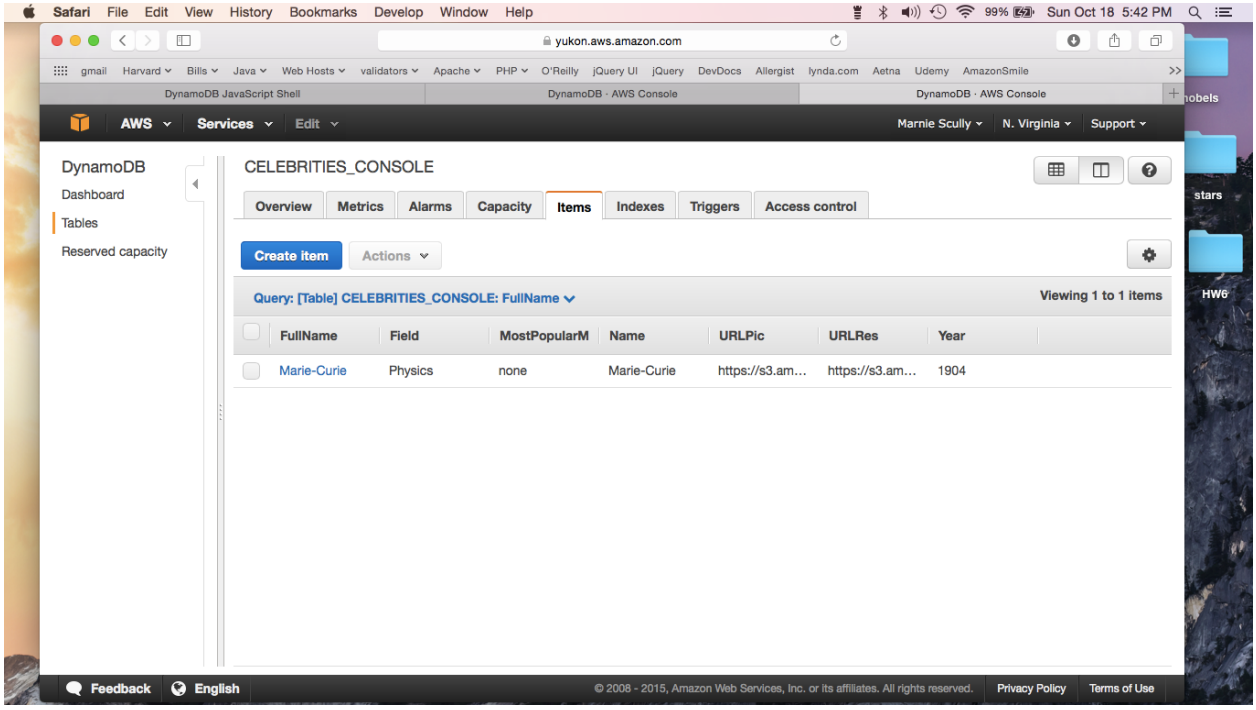
Marie Curie before Update



Editing



After Update



Problem 3.
Use AWS SDK (Java or any other) to do the same work as in Problem 2, with minor changes:
a) Same - except name the table "CELEBRITIES_SDK"
b) Same
c) Same
d) Query both CELEBRITIES_CONSOLE and CELEBRITIES_SDK tables via SDK APIs - demonstrate that you see the inserted data
e) Same
Provide working code and capture all stages of testing. **[25 Points]**

DynamoDBSample.java

```
import java.util.HashMap;
import java.util.Map;

import com.amazonaws.AmazonClientException;
import com.amazonaws.AmazonServiceException;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Region;
```

```
import com.amazonaws.regions.Regions;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.AttributeValue;
import com.amazonaws.services.dynamodbv2.model.ComparisonOperator;
import com.amazonaws.services.dynamodbv2.model.Condition;
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;
import com.amazonaws.services.dynamodbv2.model.DescribeTableRequest;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.services.dynamodbv2.model.PutItemRequest;
import com.amazonaws.services.dynamodbv2.model.PutItemResult;
import com.amazonaws.services.dynamodbv2.model.ScalarAttributeType;
import com.amazonaws.services.dynamodbv2.model.ScanRequest;
import com.amazonaws.services.dynamodbv2.model.ScanResult;
import com.amazonaws.services.dynamodbv2.model.TableDescription;
import com.amazonaws.services.dynamodbv2.util.Tables;

/**
 * Marnie Scully HW6 Problem 3
 */
public class DynamoDBSample {

    /*
     * Before running the code:
     *      Fill in your AWS access credentials in the provided credentials
     *      file template, and be sure to move the file to the default location
     *     (/Users/marnie/.aws/credentials) where the sample code will load the
     *      credentials from.
     *      https://console.aws.amazon.com/iam/home?#security_credential
     *
     * WARNING:
     *      To avoid accidental leakage of your credentials, DO NOT keep
     *      the credentials file in your source directory.
     */

    static AmazonDynamoDBClient dynamoDB;

    /**
     * The only information needed to create a client are security credentials
     * consisting of the AWS Access Key ID and Secret Access Key. All other
     * configuration, such as the service endpoints, are performed
```

```
     * automatically. Client parameters, such as proxies, can be specified in an
     * optional ClientConfiguration object when constructing a client.
     *
     * @see com.amazonaws.auth.BasicAWSCredentials
     * @see com.amazonaws.auth.ProfilesConfigFile
     * @see com.amazonaws.ClientConfiguration
     */
    private static void init() throws Exception {
        /*
         * The ProfileCredentialsProvider will return your [marniescully]
         * credential profile by reading from the credentials file located at
         * (/Users/marnie/.aws/credentials)..
         */
        AWSCredentials credentials = null;
        try {
            credentials = new ProfileCredentialsProvider("marniescully").getCredentials();
        } catch (Exception e) {
            throw new AmazonClientException(
                    "Cannot load the credentials from the credential profiles file. " +
                    "Please make sure that your credentials file is at the correct " +
                    "location (/Users/marnie/.aws/credentials)., and is in valid format.",
                    e);
        }
        dynamoDB = new AmazonDynamoDBClient(credentials);
        Region usEast1 = Region.getRegion(Regions.US_EAST_1);
        dynamoDB.setRegion(usEast1);
    }

    public static void main(String[] args) throws Exception {
        init();

        try {
            String tableName = "CELEBRITIES_SDK";

            // Create table if it does not exist yet
            if (Tables.doesTableExist(dynamoDB, tableName)) {
                System.out.println("Table " + tableName + " is already ACTIVE");
            } else {
                // Create a table with a primary hash key named 'name', which holds a string
                CreateTableRequest createTableRequest = new
CreateTableRequest().withTableName(tableName)
                    .withKeySchema(new
KeySchemaElement().withAttributeName("name").withKeyType(KeyType.HASH))
```

```
            .withAttributeDefinitions(new
AttributeDefinition().withAttributeName("name").withAttributeType(ScalarAttributeType.S))
            .withProvisionedThroughput(new
ProvisionedThroughput().withReadCapacityUnits(1L).withWriteCapacityUnits(1L));
            TableDescription createdTableDescription =
dynamoDB.createTable(createTableRequest).getTableDescription();
            System.out.println("Created Table: " + createdTableDescription);

            // Wait for it to become active
            System.out.println("Waiting for " + tableName + " to become ACTIVE...");
            Tables.awaitTableToBecomeActive(dynamoDB, tableName);
        }

        // Describe our new table
        DescribeTableRequest describeTableRequest = new
DescribeTableRequest().withTableName(tableName);
        TableDescription tableDescription =
dynamoDB.describeTable(describeTableRequest).getTable();
        System.out.println("Table Description: " + tableDescription);

        // Add an item
        Map<String, AttributeValue> item = newItem("Marilyn-Monroe", 0, "none", "https://
s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg",
                    "https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx",
"Some Like it Hot");
        PutItemRequest putItemRequest = new PutItemRequest(tableName, item);
        PutItemResult putItemResult = dynamoDB.putItem(putItemRequest);
        System.out.println("Result: " + putItemResult);

        // Add another item
        item = newItem("Dustin-Hoffman", 0, "none", "https://s3.amazonaws.com/
starsandnobels/stars/images/dustin.jpeg",
                    "https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx",
"Tootsie");
        putItemRequest = new PutItemRequest(tableName, item);
        putItemResult = dynamoDB.putItem(putItemRequest);
        System.out.println("Result: " + putItemResult);

        // Add another item
        item = newItem("Meryl-Streep", 0, "none", "https://s3.amazonaws.com/starsandnobels/
stars/images/meryl.jpeg",
                    "https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx",
"Out of Africa");
```

```
        putItemRequest = new PutItemRequest(tableName, item);
        putItemResult = dynamoDB.putItem(putItemRequest);
        System.out.println("Result: " + putItemResult);

        // Add another item
        item = newItem("Marie-Curie", 1903, "Physics", "https://s3.amazonaws.com/
starsandnobels/nobels/images/curie.jpeg",
                        "https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx",
"none");
        putItemRequest = new PutItemRequest(tableName, item);
        putItemResult = dynamoDB.putItem(putItemRequest);
        System.out.println("Result: " + putItemResult);

        // Add another item
        item = newItem("George-Bernard-Shaw", 1925, "Literature", "https://
s3.amazonaws.com/starsandnobels/nobels/images/shaw.jpeg",
                        "https://s3.amazonaws.com/starsandnobels/nobels/resumes/shaw.docx",
"none");
        putItemRequest = new PutItemRequest(tableName, item);
        putItemResult = dynamoDB.putItem(putItemRequest);
        System.out.println("Result: " + putItemResult);

        // Add another item
        item = newItem("Eugene-Oneil", 1936, "Literature", "https://s3.amazonaws.com/
starsandnobels/nobels/images/oneil.jpeg",
                        "https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx",
"none");
        putItemRequest = new PutItemRequest(tableName, item);
        putItemResult = dynamoDB.putItem(putItemRequest);
        System.out.println("Result: " + putItemResult);

        // Scan CELEBRITIES_SDK
        // Scan items for Laureates with a prize year attribute greater than -1 (which would be all)
        HashMap<String, Condition> scanFilter = new HashMap<String, Condition>();
        Condition condition = new Condition()
            .withComparisonOperator(ComparisonOperator.GT.toString())
            .withAttributeValueList(new AttributeValue().withN("-1"));
        scanFilter.put("year", condition);
        ScanRequest scanRequest = new ScanRequest(tableName).withScanFilter(scanFilter);
        ScanResult scanResult = dynamoDB.scan(scanRequest);
        System.out.println("CELEBRITIES_SDK Result: " + scanResult);

        // Scan CELEBRITIES_CONSOLE Will only return Laureates
```

```
        // Scan items for Laureates with a prize year attribute greater than -1 (which would be all)
        tableName = "CELEBRITIES_CONSOLE";
        scanFilter = new HashMap<String, Condition>();
        condition = new Condition()
            .withComparisonOperator(ComparisonOperator.GT.toString())
            .withAttributeValueList(new AttributeValue().withN("1900"));
        scanFilter.put("Year", condition);
        scanRequest = new ScanRequest(tableName).withScanFilter(scanFilter);
        scanResult = dynamoDB.scan(scanRequest);
        System.out.println("CELEBRITIES_CONSOLE Result: " + scanResult);

        // Changing the year of nobel prize for Eugene O'Neil
        tableName = "CELEBRITIES_SDK";
        item = newItem("Eugene-Oneil", 1946, "Literature", "https://s3.amazonaws.com/
starsandnobels/nobels/images/oneil.jpeg",
                        "https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx",
"none");
        putItemRequest = new PutItemRequest(tableName, item);
        putItemResult = dynamoDB.putItem(putItemRequest);
        System.out.println("Changing O'Neil Year: ");

        // View CELEBRITIES_SDK to see last update
        scanRequest = new ScanRequest(tableName);
        scanResult = dynamoDB.scan(scanRequest);
        System.out.println("CELEBRITIES_SDK Result: " + scanResult);


    } catch (AmazonServiceException ase) {
        System.out.println("Caught an AmazonServiceException, which means your request
made it "
                + "to AWS, but was rejected with an error response for some reason.");
        System.out.println("Error Message:    " + ase.getMessage());
        System.out.println("HTTP Status Code: " + ase.getStatusCode());
        System.out.println("AWS Error Code:   " + ase.getErrorCode());
        System.out.println("Error Type:       " + ase.getErrorType());
        System.out.println("Request ID:       " + ase.getRequestId());
    } catch (AmazonClientException ace) {
        System.out.println("Caught an AmazonClientException, which means the client
encountered "
                + "a serious internal problem while trying to communicate with AWS, "
                + "such as not being able to access the network.");
        System.out.println("Error Message: " + ace.getMessage());
    }
```

```
    }

    // I edited this function to have the fields of the CELEBRITIES_SDK table
    private static Map<String, AttributeValue> newItem(String name, int year, String field, String
URLPic, String URLRes, String popularMovie) {
        Map<String, AttributeValue> item = new HashMap<String, AttributeValue>();
        item.put("name", new AttributeValue(name));
        item.put("year", new AttributeValue().withN(Integer.toString(year)));
        item.put("field", new AttributeValue(field));
        item.put("URLPic", new AttributeValue(URLPic));
        item.put("URLRes", new AttributeValue(URLRes));
        item.put("popularMovie", new AttributeValue(popularMovie));
        return item;
    }

}
```

Output in Eclipse Console

Table CELEBRITIES_SDK is already ACTIVE
Table Description: {AttributeDefinitions: [{AttributeName: name,AttributeType: S}],TableName: CELEBRITIES_SDK,KeySchema: [{AttributeName: name,KeyType: HASH}],TableStatus: ACTIVE,CreationDateTime: Sun Oct 18 18:23:18 EDT 2015,ProvisionedThroughput: {NumberOfDecreasesToday: 0,ReadCapacityUnits: 1,WriteCapacityUnits: 1},TableSizeBytes: 0,ItemCount: 0,TableArn: arn:aws:dynamodb:us-east-1:413513583861:table/CELEBRITIES_SDK,}
Result: {}
Result: {}
Result: {}
Result: {}
Result: {}
Result: {}
CELEBRITIES_SDK Result: {Items: [{URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx,}, field={S: Literature,}, year={N: 1936,}, name={S: Eugene-Oneil,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/oneil.jpeg,}, popularMovie={S: none,}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx,}, field={S: none,}, year={N: 0,}, name={S: Marilyn-Monroe,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg,}, popularMovie={S: Some Like it Hot,}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx,}, field={S: Physics,}, year={N: 1903,}, name={S: Marie-Curie,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpeg,}, popularMovie={S: none,}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx,}, field={S: none,},

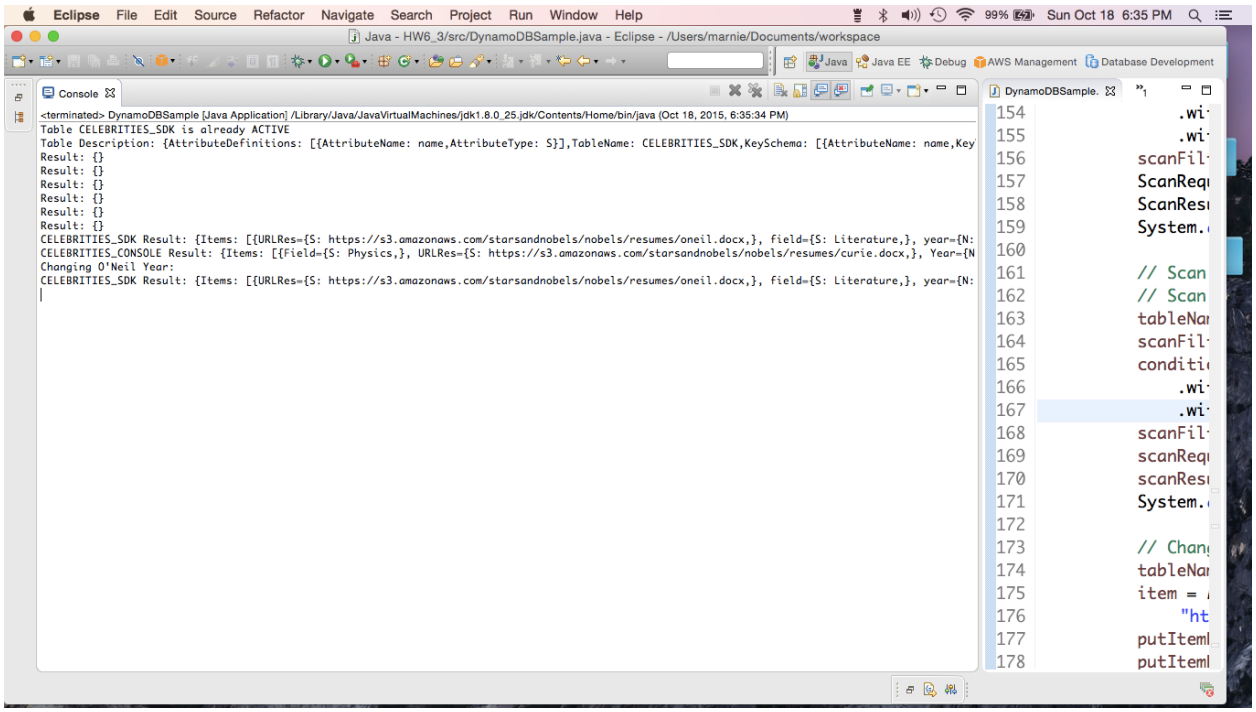
year={N: 0,}, name={S: Dustin-Hoffman,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg,}, popularMovie={S: Tootsie,}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/shaw.docx,}, field={S: Literature,}, year={N: 1925,}, name={S: George-Bernard-Shaw,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/shaw.jpeg,}, popularMovie={S: none,}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx,}, field={S: none,}, year={N: 0,}, name={S: Meryl-Streep,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/stars/images/meryl.jpeg,}, popularMovie={S: Out of Africa,}}],Count: 6,ScannedCount: 6,}

CELEBRITIES_CONSOLE Result: {Items: [{Field={S: Physics,}, URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx,}, Year={N: 1904,}, MostPopularMovie={S: none,}, FullName={S: Marie-Curie,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpeg,}, Name={S: Marie-Curie,}}, {Field={S: Literature,}, URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx,}, Year={N: 1936,}, MostPopularMovie={S: none,}, FullName={S: Eugene O'Neil,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/oneil.jpeg,}, Name={S: Eugene-Oneil,}}, {Field={S: Literature,}, URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/shaw.docx,}, Year={N: 1925,}, MostPopularMovie={S: none,}, FullName={S: George Bernard Shaw,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/shaw.jpeg,}, Name={S: George-Bernard-Shaw,}}],Count: 3,ScannedCount: 6,}

Changing O'Neil Year:
CELEBRITIES_SDK Result: {Items: [{URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/oneil.docx,}, field={S: Literature,}, year={N: 1946,}, name={S: Eugene-Oneil,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/oneil.jpeg,}, popularMovie={S: none,}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/marilyn.docx,}, field={S: none,}, year={N: 0,}, name={S: Marilyn-Monroe,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/stars/images/marilyn.jpeg,}, popularMovie={S: Some Like it Hot,}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/curie.docx,}, field={S: Physics,}, year={N: 1903,}, name={S: Marie-Curie,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/curie.jpeg,}, popularMovie={S: none,}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/dustin.docx,}, field={S: none,}, year={N: 0,}, name={S: Dustin-Hoffman,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/stars/images/dustin.jpeg,}, popularMovie={S: Tootsie,}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/nobels/resumes/shaw.docx,}, field={S: Literature,}, year={N: 1925,}, name={S: George-Bernard-Shaw,}, URLPic={S: https://s3.amazonaws.com/starsandnobels/nobels/images/shaw.jpeg,}, popularMovie={S: none,}}, {URLRes={S: https://s3.amazonaws.com/starsandnobels/stars/resumes/meryl.docx,}, field={S: none,}, year={N: 0,}, name={S: Meryl-Streep,}, URLPic={S: https://

s3.amazonaws.com/starsandnobels/stars/images/meryl.jpeg,}, popularMovie={S: Out of Africa,}}],Count: 6,ScannedCount: 6,}

ScreenShot of Eclipse Console

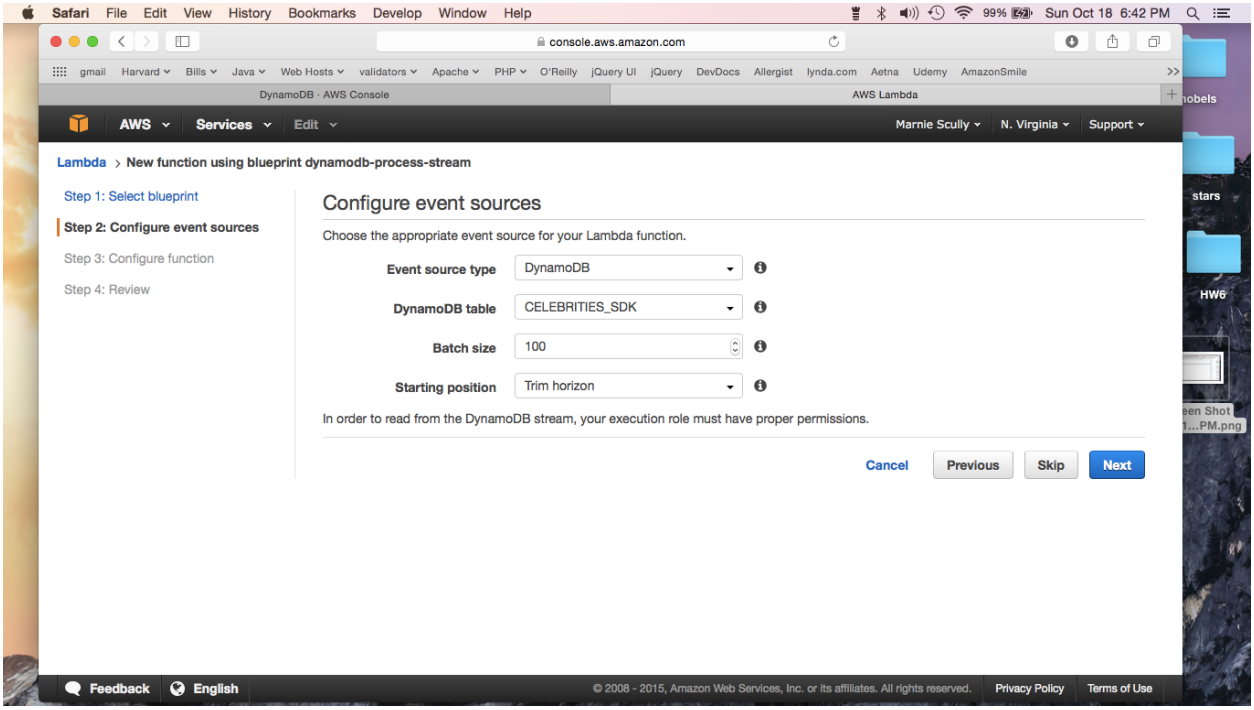


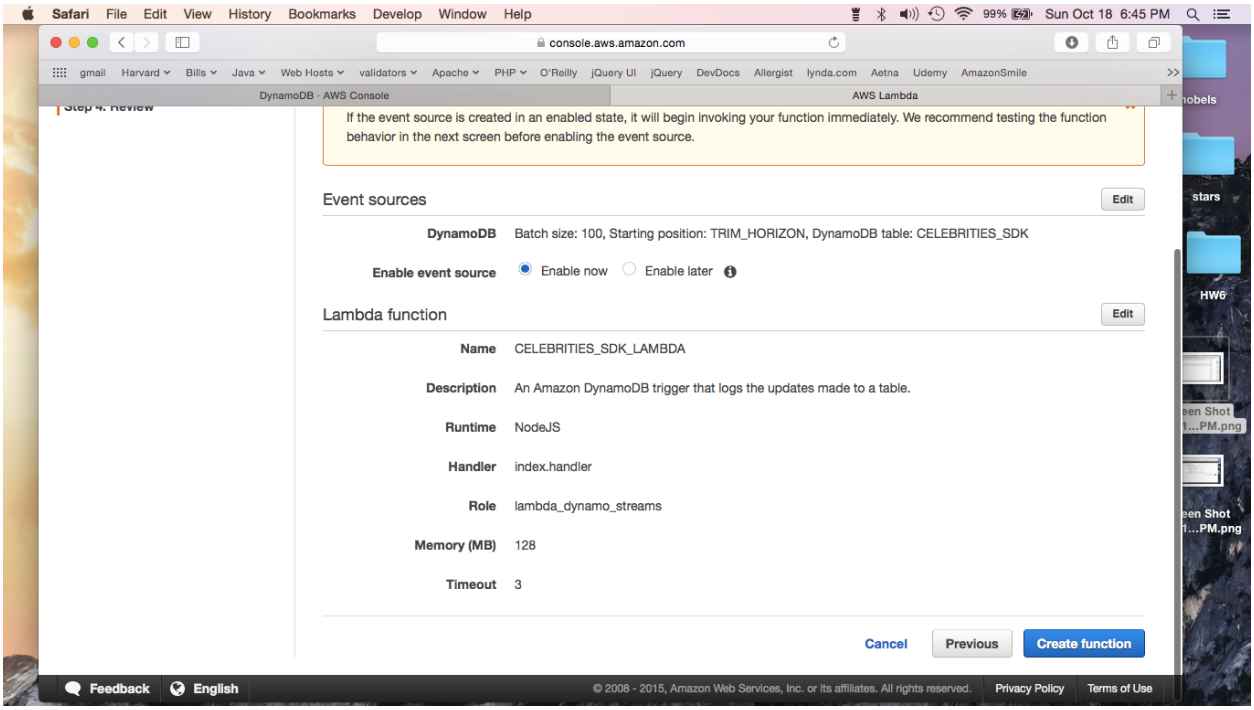
Problem 4.

Create a Lambda function that will monitor DynamoDB stream of table CELEBRITIES_SDK and write all operations performed on that table into CloudWatch logs. Use Java class developed in Problem 3 to generate load on table CELEBRITIES_SDK. Comment out the section of the class where you create the table

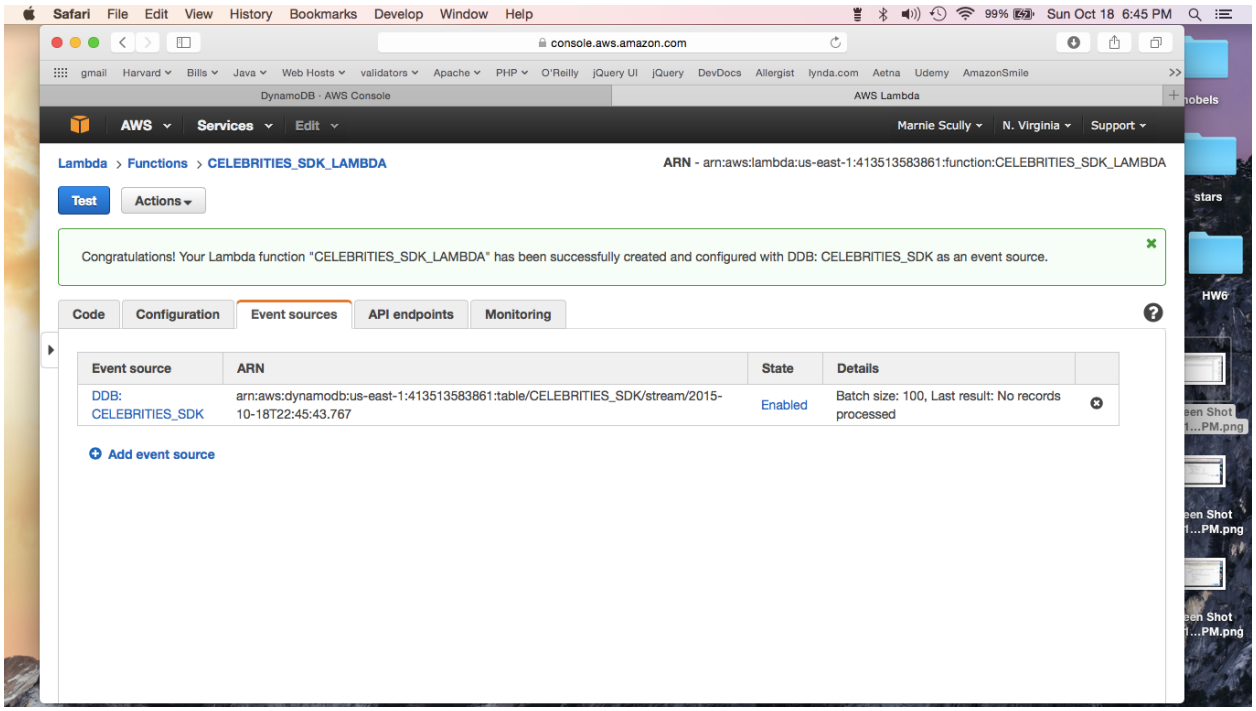
itself. Capture all steps of your development and testing. [**20 Points]**

Create Lambda Function

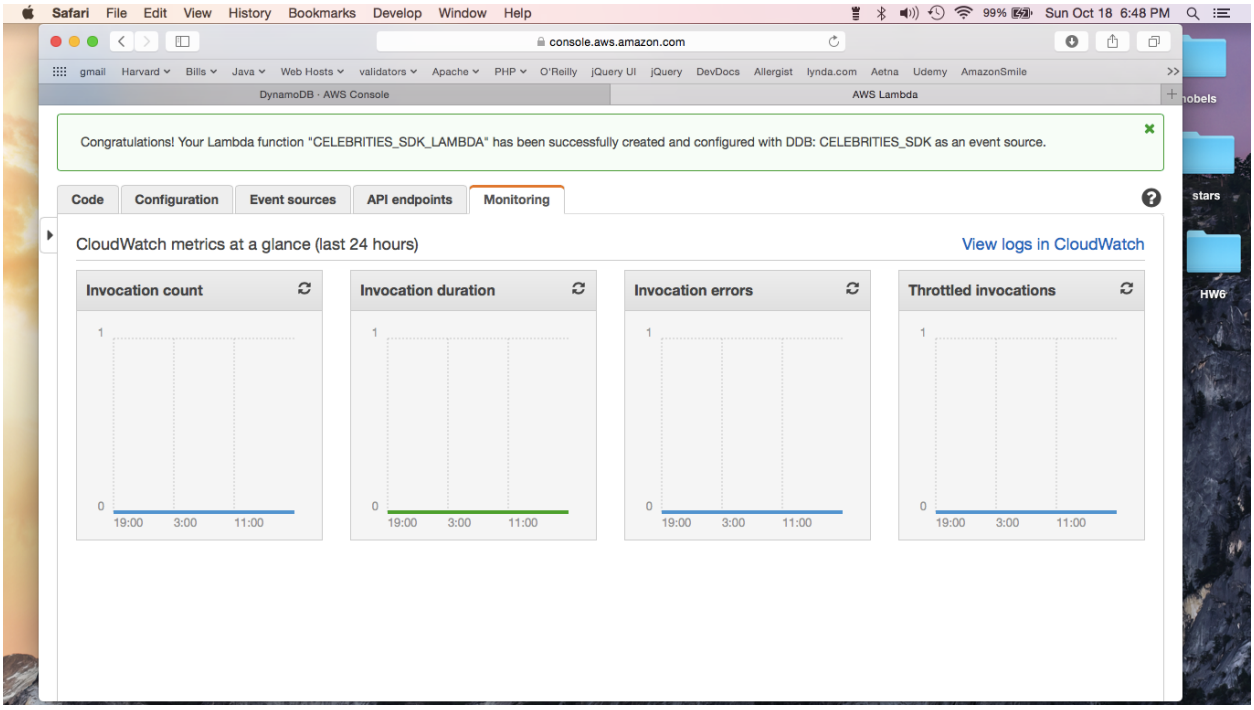




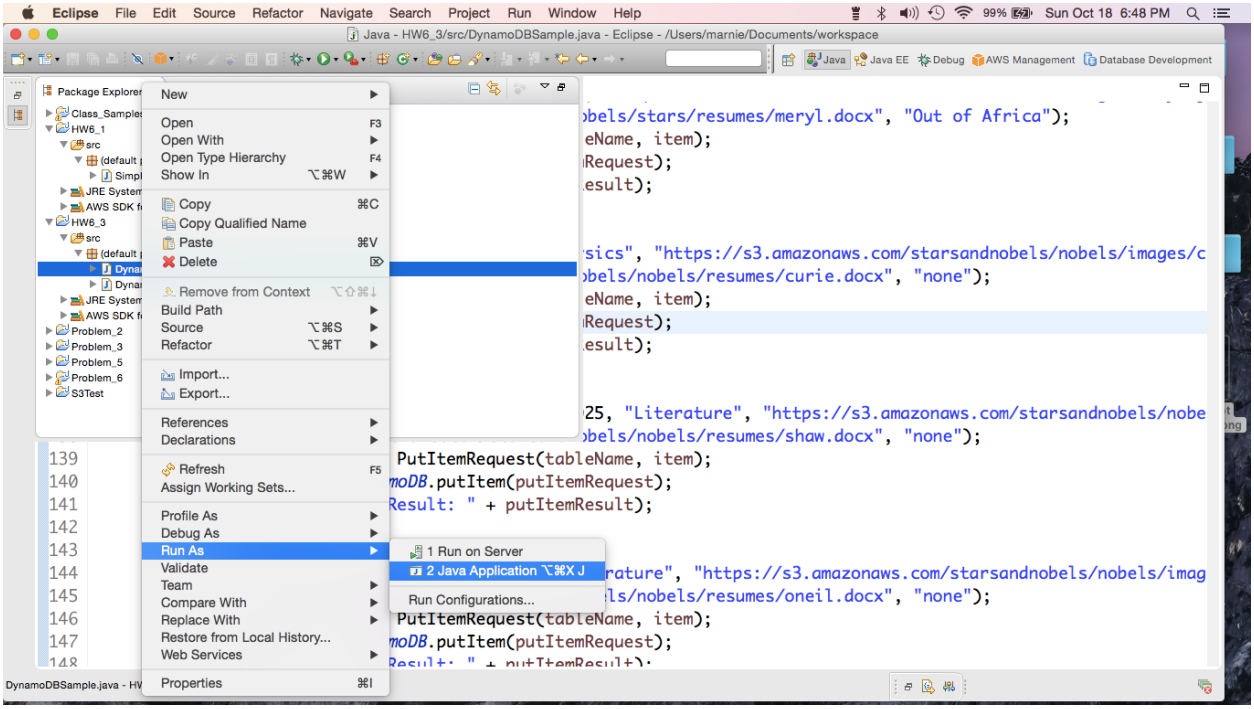
HW6_ScullyMarnie



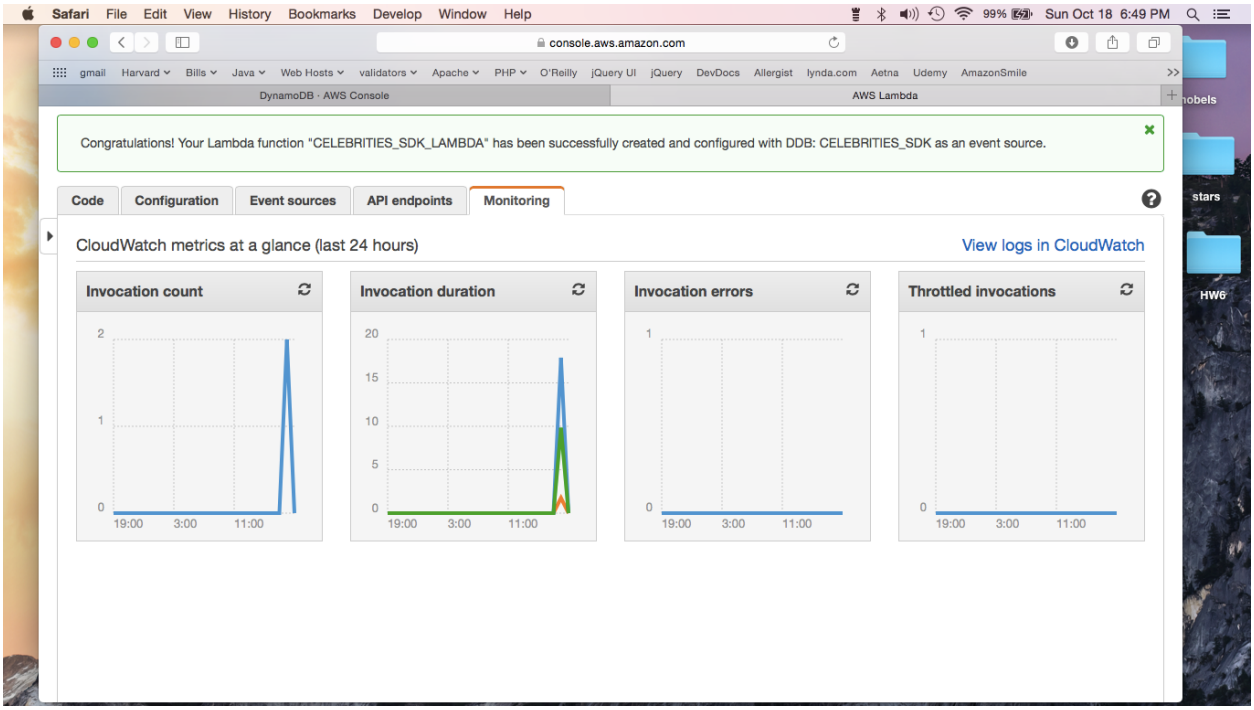
Before Load



Run Code From Problem 3 again



CloudWatch after Problem 3 ran



Problem 5.

When done, delete all DynamoDB tables and Lambda functions, programmatically or using AWS CLI. Provide working code and capture all stages of testing. **[10 Points]**

Delete Lambda Function from AWS CLI

Delete Tables programmatically with DynamoDBTest.java (modified)

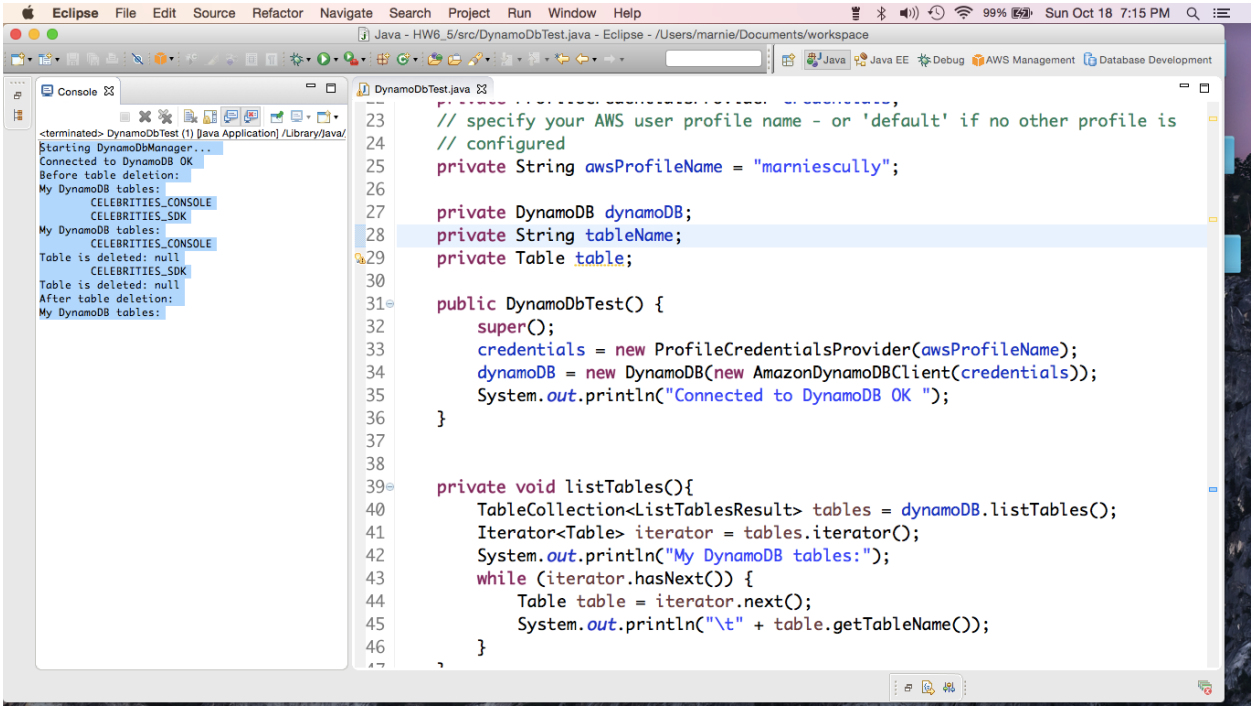
Delete Tables Programmatically

Output in Eclipse

Starting DynamoDbManager...
Connected to DynamoDB OK
Before table deletion:
My DynamoDB tables:
CELEBRITIES_CONSOLE
CELEBRITIES_SDK
My DynamoDB tables:

CELEBRITIES_CONSOLE
Table is deleted: null
CELEBRITIES_SDK
Table is deleted: null
After table deletion:
My DynamoDB tables:

Screen Shot of Eclipse



DynamoDBTest.java

```
import java.util.ArrayList;
import java.util.Iterator;

import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.dynamodbv2.AmazonDynamoDBClient;
import com.amazonaws.services.dynamodbv2.document.DynamoDB;
import com.amazonaws.services.dynamodbv2.document.Item;
import com.amazonaws.services.dynamodbv2.document.Table;
import com.amazonaws.services.dynamodbv2.document.TableCollection;
import com.amazonaws.services.dynamodbv2.model.AttributeDefinition;
import com.amazonaws.services.dynamodbv2.model.CreateTableRequest;
import com.amazonaws.services.dynamodbv2.model.KeySchemaElement;
import com.amazonaws.services.dynamodbv2.model.KeyType;
```

```
import com.amazonaws.services.dynamodbv2.model.ListTablesResult;
import com.amazonaws.services.dynamodbv2.model.ProvisionedThroughput;
import com.amazonaws.services.dynamodbv2.model.TableDescription;

public class DynamoDbTest {

	private ProfileCredentialsProvider credentials;
	// specify your AWS user profile name - or 'default' if no other profile is
	// configured
	private String awsProfileName = "marniescully";

	private DynamoDB dynamoDB;
	private String tableName;
	private Table table;

	public DynamoDbTest() {
		super();
		credentials = new ProfileCredentialsProvider(awsProfileName);
		dynamoDB = new DynamoDB(new AmazonDynamoDBClient(credentials));
		System.out.println("Connected to DynamoDB OK ");
	}


	private void listTables(){
		TableCollection<ListTablesResult> tables = dynamoDB.listTables();
		Iterator<Table> iterator = tables.iterator();
		System.out.println("My DynamoDB tables:");
		while (iterator.hasNext()) {
			Table table = iterator.next();
			System.out.println("\t" + table.getTableName());
		}
	}

	private void deleteTables() {

		TableCollection<ListTablesResult> tables = dynamoDB.listTables();
		Iterator<Table> iterator = tables.iterator();
		System.out.println("My DynamoDB tables:");
		while (iterator.hasNext()) {
			Table table = iterator.next();
			System.out.println("\t" + table.getTableName());
			table.delete();
				try {
					table.waitForDelete();
				} catch (InterruptedException e) {
					// TODO Auto-generated catch block
					System.out.println("Failed to delete table: " + tableName);
					e.printStackTrace();
				}
			System.out.println("Table is deleted: " + tableName);
		}

	}
```

```
	public static void main(String[] args) {
		System.out.println("Starting DynamoDbManager... ");
		DynamoDbTest ddb = new DynamoDbTest();
		System.out.println("Before table deletion: ");
		ddb.listTables();
		ddb.deleteTables();
		System.out.println("After table deletion: ");
		ddb.listTables();
		/*  */
	}

}
```