20\_07\_학습내용

**목차**

1. **Get 메소드**

**1-1. 데이터 검색 : scan**

**1-2. 데이터 삽입 : put**

**1-3. 데이터 변경 : update**

**1-4. 데이터 삭제 : delete**

1. **Post 메소드**
2. **응답에서 특정 데이터만 추출**

어제에 이어서

1. Get 메소드

테이블 항목 이름이 Value->Greet로 변경되었다.

그 이유는, 어제 에러 발생 원인이, Value가 예약어여서 안된 것이었다.

**1-1. 대이터 검색**

1. Scan

index.js는 다음과 같은 코드를 가진다.

const AWS = require('aws-sdk');

const dynamo = new AWS.DynamoDB.DocumentClient();

exports.handler = async (param1, context) => {

let body;

body = await dynamo.scan(param1).promise();

return body;

};

ProjectionExpression 및 FilterExpression을 쓸 수 있는데

FilterExpression은 모든 Column(Attribute)들에 대해서 가능하지만

ProjectionExpression은 파티션 키에 대해서만 가능한 것으로 파악된다.

FilterExpression은

|  |  |
| --- | --- |
| {  "TableName": "TestTable",  "FilterExpression":"#id = :y",  "ExpressionAttributeNames":{  "#id": "ID" },  "ExpressionAttributeValues": {  ":y": "1" }  }  처럼 사용하면 | {  "TableName": "TestTable",  "ProjectionExpression":"ID",  "FilterExpression":"#hello = :y",  "ExpressionAttributeNames":{  "#hello": "Greet" },  "ExpressionAttributeValues": {  ":y": "안녕" }  } |

굳이 json으로 하지 않더라도

index.js를 다음과 같이 바꾸면 똑같은 결과를 얻을 수 있다.

|  |  |
| --- | --- |
| const AWS = require('aws-sdk');  const dynamo = new AWS.DynamoDB.DocumentClient();  exports.handler = async (param1, context) => {  var param = {  TableName: "TestTable",  ProjectionExpression: "ID",  FilterExpression: "#hi = :y",  ExpressionAttributeNames: {  "#hi": "Greet",  },  ExpressionAttributeValues: {  ":y": "안녕"  }  };  let body;  body = await dynamo.scan(param).promise();  return body;  }; |  |

1. 현재 Query로 진행시 Timeout으로 인해 진행되지 않는다. 이에 대해서는 더 찾아봐야 할 듯...

코드는

const AWS = require('aws-sdk');

const dynamo = new AWS.DynamoDB.DocumentClient();

var params = {

TableName : "TestTable",

KeyConditionExpression: "Greet = :y",

ExpressionAttributeValues: {

":y": "안녕"

}

};

var body = dynamo.query(params, function(err, data) {});

return body;

로 진행하였다.

======================================================================

**1-2. 데이터 추가**

|  |  |
| --- | --- |
| const AWS = require('aws-sdk');  const dynamo = new AWS.DynamoDB.DocumentClient();  exports.handler = async (param1, context) => {  var param = {  TableName: "TestTable",  Item :  {  ID : "3",  Greet : "Ni hao ma",  Filter : "false"  }  };  let body;  body = await dynamo.put(param).promise();  return body;  }; | {  "TableName": "TestTable",  "Item" : {  "ID" : "4",  "Greet" : "Hello",  "Filter" : "false"  }  }  -----------------------------------------------------  const AWS = require('aws-sdk');  const dynamo = new AWS.DynamoDB.DocumentClient();  exports.handler = async (param1, context) => {  let body;  body = await dynamo.put(param1).promise();  return body;  }; |
|  |  |

**1-3. 데이터 변경**

|  |  |
| --- | --- |
| {  "TableName": "TestTable",  "Key" : {  "ID" : "3"  },  "UpdateExpression" : "set Greet= :y",  "ExpressionAttributeValues" : {  ":y" : "Hi"  }  }  -----------------------------------------------------const AWS = require('aws-sdk');  const dynamo = new AWS.DynamoDB.DocumentClient();  exports.handler = async (param1, context) => {  let body;  body = await dynamo.update(param1).promise();  return body;  }; | const AWS = require('aws-sdk');  const dynamo = new AWS.DynamoDB.DocumentClient();  exports.handler = async (param1, context) => {  var param =  {  TableName: "TestTable",  Key : {  ID : "3"  },  UpdateExpression : "set Greet= :y",  ExpressionAttributeValues : {  ":y" : "안녕히계세요."  }  }  let body;  body = await dynamo.update(param).promise();  return body;  }; |

**1-4. 데이터 삭제**

처음에는

 에러 뜨기에 IAM으로 가서 정책을 변경해 주었다.



밑줄 친 내용을 추가해 주었다.

그 후 실행해본 결과는 다음 페이지와 같다.

|  |  |
| --- | --- |
| const AWS = require('aws-sdk');  const dynamo = new AWS.DynamoDB.DocumentClient();  exports.handler = async (param1, context) => {  var param =  {  TableName: "TestTable",  Key : {  ID : "4"  }  }  let body;  body = await dynamo.delete(param).promise();  return body;  }; | {  "TableName" : "TestTable",  "Key" : {  "ID" : "3"  }  }  -----------------------------------------------------  const AWS = require('aws-sdk');  const dynamo = new AWS.DynamoDB.DocumentClient();  exports.handler = async (param1, context) => {  let body;  body = await dynamo.delete(param1).promise();  return body;  }; |

람다에서의 테스트 뿐만 아니라

쭉 전체적인 테스트를 해보아도

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  | |
|  | |  | |

이 내용들을 적용하여

Rest 방식으로 접근해 보자! (URL으로)

(배포 주소를 통해 get 메소드 및 파라미터로 dynamo DB의 항목 리턴받기!)

매핑 템플릿을

{

"TableName": "TestTable",

"ProjectionExpression":"ID",

"FilterExpression":"Greet = :y",

"ExpressionAttributeValues": {

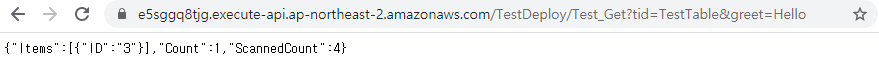
":y": "$input.params().querystring.greet" }

}

과 같이 작성한 후

아래의 url로 접속하게 되면

<https://e5sggq8tjg.execute-api.ap-northeast-2.amazonaws.com/TestDeploy/Test_Get?tid=TestTable&greet=Hello>



다음과 같은 결과를 얻을 수 있다.

================================================================

**2. Post 방식**

* 데이터 검색

1) 요청 패킷 바디에다가 json 형식으로 쿼리 집어넣고

{

"TableName": "TestTable",

"FilterExpression":"#id = :y",

"ExpressionAttributeNames":{

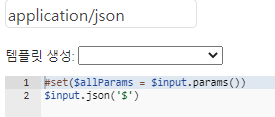
"#id": "ID" },

"ExpressionAttributeValues": {

":y": "1" }

}

2) 매핑 템플릿을 다음과 같이 작성



#set부분을 지워도 가능함(allParams객체에 파라미터들 다 집어넣겠다!라 현재는 안쓰기에 지워도 된다.)

3) 람다는 Get과 동일

const AWS = require('aws-sdk');

const dynamo = new AWS.DynamoDB.DocumentClient();

exports.handler = async (param, context) => {

let body;

body = await dynamo.scan(param).promise();

return body;

};

4) 결과



* 데이터 삽입도 똑같다.

{

"TableName": "TestTable",

"Item" : {

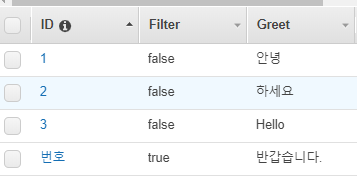
"ID" : "3",

"Greet" : "Hello",

"Filter" : "false"

}

}



const AWS = require('aws-sdk');

const dynamo = new AWS.DynamoDB.DocumentClient();

exports.handler = async (param, context) => {

let body;

body = await dynamo.scan(param).promise();

return body;

};

데이터 삭제와 변경은 생략

================================================================**3. 응답에서 원하는 데이터만 추출**

위의 Post의 결과의 ‘Hello’이라는 값을 나중에 변수로 쓰고 싶을 수 있다.

ProjectionExpression을 쓰면

Response:

{

"Items": [

{

"ID": "3"

}

],

"Count": 1,

"ScannedCount": 4

}

로 전달 받기에 후처리가 필요하다.

따라서 바로 “Hello”만 클라이언트에게 리턴해 주기 위해서는

응답에서 Items를 뽑아낸 다음 Greet를 뽑아내면 될 것이다.

* Response의 Items

위에 것을 시도하기 전에 우선 Items에 대해 확인해야 할 것이다.

|  |  |
| --- | --- |
| {  "TableName": "TestTable"  } | const AWS = require('aws-sdk');  const dynamo = new AWS.DynamoDB.DocumentClient();  exports.handler = async (param, context) => {  let body;  body = await dynamo.scan(param).promise();  return body;  }; |

를 실행하게 될 경우

Response:

{

"Items": [

{ -----------------------------------

"ID": "2", 얘가 Items[0]

"Filter": false,

"Greet": "하세요"

}, ----------------------------------

{ ----------------------------------

"ID": "1", 얘가 Items[1]

"Filter": false,

"Greet": "안녕"

}, ----------------------------------

{ ----------------------------------

"ID": "3", 얘가 Items[2]

"Greet": "Hello",

"Filter": "false"

}, ----------------------------------

{ ----------------------------------

"ID": "번호", 얘가 Items[3]에 해당한다.

"Filter": true,

"Greet": "반갑습니다."

} ----------------------------------

],

"Count": 4,

"ScannedCount": 4

}

순서는 왜 자기 마음대로인지 모르겠다…

이렇게 순서가 엉망이 될 수도 있기에, 결과 값이 여러개 나오는 경우에는 이 추출 방식을 사용하지 말자.

Json을 다시 결과값 하나만 나오는 예제로 바꾸면

{

"TableName": "TestTable",

"FilterExpression": "#id >= :y",

"ExpressionAttributeNames": {

"#id": "ID"

},

"ExpressionAttributeValues": {

":y": "3"

}

}

얘의 결과 값들은 Items[0]에 들어가게 되어있다.

따라서 index.js코드 리턴부분을 살짝 바꿔주게 되면

원하는 “Hello”만 리턴 받을 수 있다.

