



## MyStep4:

### Specify the String Parameter in the right way

The screenshot shows the AWS API Gateway console. On the left, the 'Resources' pane shows the hierarchy: `/` > `/getpetdetailsbyname` > `GET`. The main pane is titled 'Method Execution /getpetdetailsbyname - GET - Method Request'. It contains a 'Settings' section with the following configuration:

- Authorization: NONE
- Request Validator: NONE
- API Key Required: false
- URL Query String Parameters: A table with one parameter, `nameID`, which is required and has caching disabled.

Name	Required	Caching
<code>nameID</code>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Below the table, there is a link to 'Add query string'. Other expandable sections include 'HTTP Request Headers', 'Request Body', and 'SDK Settings'.

## MyStep5:

### Go to AWS Lambda and click the function part to create a lambda function.

The screenshot shows the AWS Lambda console. The left sidebar contains navigation links: Dashboard, Applications, Functions (highlighted), Additional resources, and Related AWS resources. The main area is titled 'Functions (2)' and shows a list of functions:

Function name	Description	Runtime	Code size	Last modified
<code>Test</code>		Python 3.8	232.0 byte	last week
<code>getPetDetailsByName</code>		Node.js 12.x	343.0 byte	last week

## MyStep6:

Write the function content and remember to click Deploy

The screenshot shows the AWS Lambda console for the function `getPetDetailsByName`. The function is configured with an API Gateway trigger. The function code is written in JavaScript and uses the AWS SDK to query a DynamoDB table named `PetDetails`. The code defines a handler function that takes an event, context, and callback as arguments. It constructs a query key based on the event's `NameID` and uses the `docClient.get` method to retrieve the pet details. The result is then passed back to the callback.

```
1 const AWS = require('aws-sdk');
2 var docClient = new AWS.DynamoDB.DocumentClient();
3
4 var tableName = "PetDetails";
5
6 exports.handler = (event, context, callback) => {
7
8   var params = {
9     TableName: tableName,
10    Key: { // capital
11      "NameID": event.NameID
12    }
13  }
14
15  docClient.get(params, function(err, data){
16    callback(err, data);
17  })
18
19  };
```

## MyStep7:

Test this function and get the result

The screenshot shows the AWS Lambda console for the function `getPetDetailsByName`. The function is configured with an API Gateway trigger. The function code is written in JavaScript and uses the AWS SDK to query a DynamoDB table named `PetDetails`. The code defines a handler function that takes an event, context, and callback as arguments. It constructs a query key based on the event's `NameID` and uses the `docClient.get` method to retrieve the pet details. The result is then passed back to the callback.

**Execution result: succeeded (logs)**

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

```
{
  "Item": {
    "Height": "45",
    "FirstName": "iPad",
    "LastName": "Wang",
    "NameID": "iPad Wang",
    "Weight": "6kg"
  }
}
```

**Summary**

Code SHA-256	Request ID
LhCK5KqU5Ju6MZy+qDZ+PGIRixSxRBcmEqcNqcPmaj0=	4c97cea2-ba7d-4d41-8b5e-234ec2c3519f
Duration	Billed duration
857.56 ms	900 ms

## MyStep8:

Also, use the postman to test whether the request link is correct

The screenshot shows the Postman interface for an "Untitled Request". The request method is "GET" and the URL is `https://d5bpzayoc2.execute-api.us-east-1.amazonaws.com/Dev/getpetdetailsbyname?NameID=iPad Wang`. The "Params" tab is selected, showing a table of query parameters.

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	NameID	iPad Wang			
	Key	Value	Description		

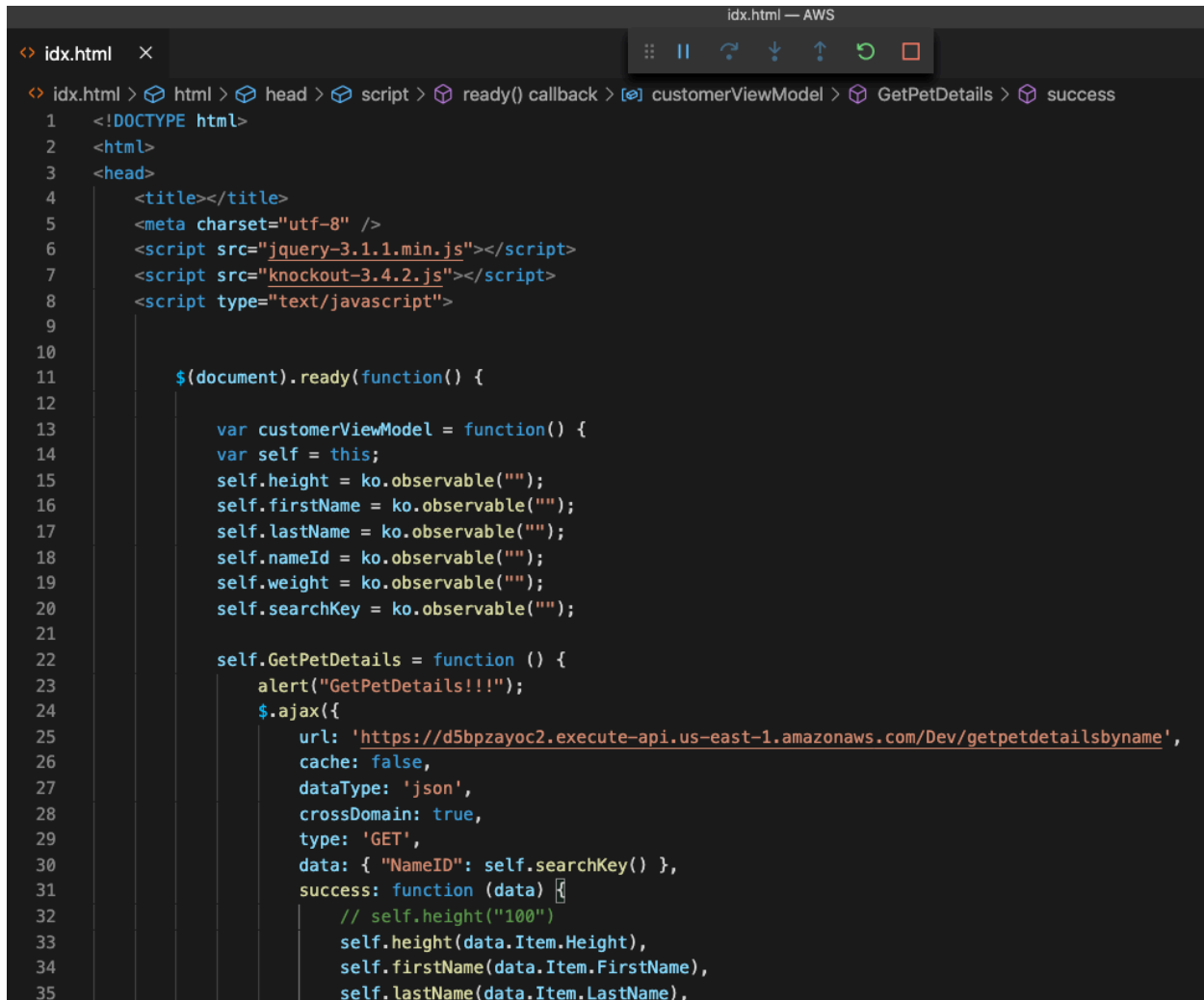
Below the query parameters, the "Body" tab is selected, showing a JSON response in "Pretty" format:

```
1 {
2   "Item": {
3     "Height": "45",
4     "FirstName": "iPad",
5     "LastName": "Wang",
6     "NameID": "iPad Wang",
7     "Weight": "6kg"
8   }
9 }
```

At the top of the interface, there is a "Launchpad" button, a "GET" method selector, and a "No Environment" dropdown. On the right, there are "Send" and "Save" buttons, and a "Cookies" tab. The status bar at the bottom indicates a "200 OK" response with a time of "1597 ms" and a size of "420 B".

## MyStep 9:

Then, let's write our html file with javascript



```
idx.html — AWS
idx.html x
idx.html > html > head > script > ready() callback > customerViewModel > GetPetDetails > success
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <title></title>
5 <meta charset="utf-8" />
6 <script src="jquery-3.1.1.min.js"></script>
7 <script src="knockout-3.4.2.js"></script>
8 <script type="text/javascript">
9
10
11 $(document).ready(function() {
12
13     var customerViewModel = function() {
14     var self = this;
15     self.height = ko.observable("");
16     self.firstName = ko.observable("");
17     self.lastName = ko.observable("");
18     self.nameId = ko.observable("");
19     self.weight = ko.observable("");
20     self.searchKey = ko.observable("");
21
22     self.GetPetDetails = function () {
23     alert("GetPetDetails!!!");
24     $.ajax({
25     url: 'https://d5bpzayoc2.execute-api.us-east-1.amazonaws.com/Dev/getpetdetailsbyname',
26     cache: false,
27     dataType: 'json',
28     crossDomain: true,
29     type: 'GET',
30     data: { "NameID": self.searchKey() },
31     success: function (data) {
32     // self.height("100")
33     self.height(data.Item.Height),
34     self.firstName(data.Item.FirstName),
35     self.lastName(data.Item.LastName),
```

## MyStep 10:

Copy this html file and the other two files to our VM(Amazon Linux) on EC2.

## MyStep 11:

Use apache to host my website on EC2

Type:

```
sudo yum update -y
sudo yum install -y httpd24 php56 php56-mysqlnd
```

to install necessary packages.

### MyStep 12:

Type:

```
sudo service httpd start
```

and we can see the apache website in the address. For example, <http://ec2-42-8-168-21.us-west-1.compute.amazonaws.com>.

### MyStep 13:

Copy(create) our html and javascript file into the /var/www/html

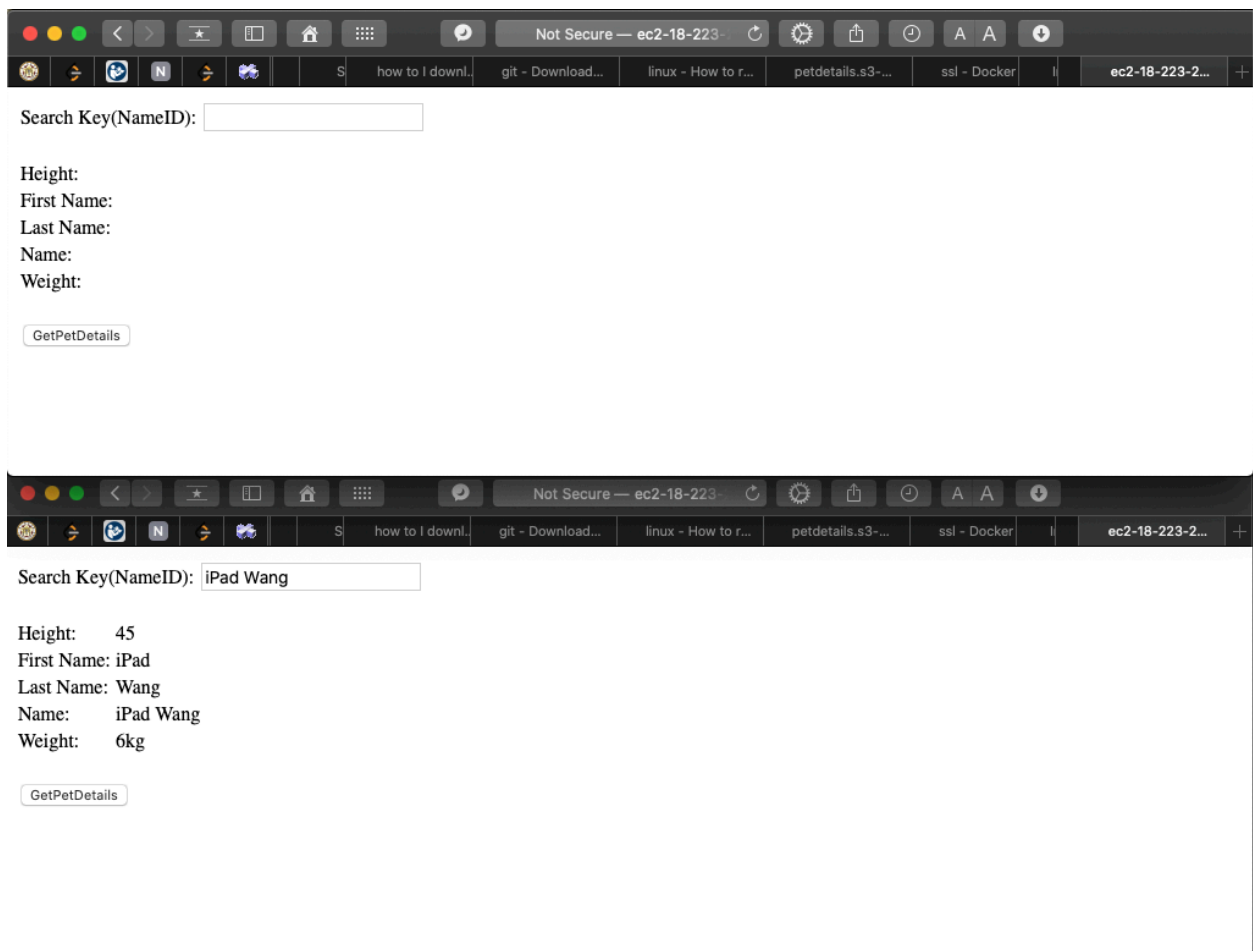
Type:

```
cd /var/www/html
```

and copy(create) the html and javascript files (which are in my github).

### MyStep 14:

Then use the browser and type [http://instance\\_public\\_address/index.html](http://instance_public_address/index.html)



**Discussion:**

In this homework, I have searched several websites and materials. I just found it is a little bit tricky in some part. For example, when I used the **ajax** to send the request, I always got error! After struggling, I finally discovered the solution. Therefore, it really help me improve my search skill.

**Github URL:**

[https://github.com/Cheng-Chi-Tang/Cloud\\_Computing\\_HW.git](https://github.com/Cheng-Chi-Tang/Cloud_Computing_HW.git)