# Spring Security: Basic Authentication and Authorization using spring boot

In this we will see how to secure rest API using Basic Authentication and Authorization. So, as we know providing security to API is a key component of a application. So, lets develop a microservice and then prove security to rest api using spring boot and spring security.

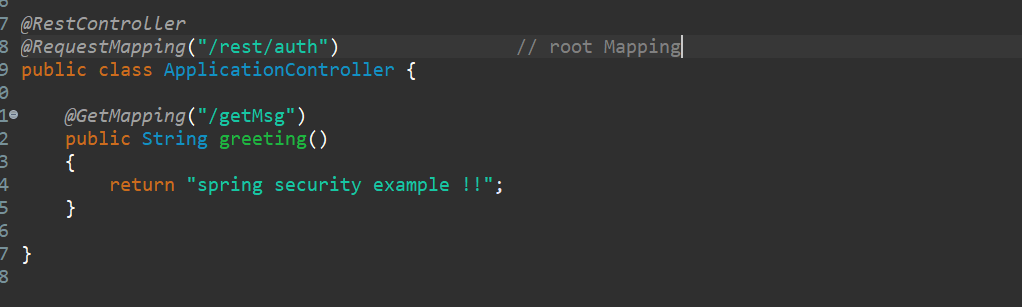
Application- spring-basic-security

Dependencies- Devtools, Web and Spring Security

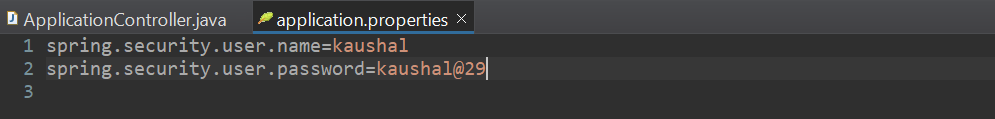
So, lets first use Spring boot provided default security login then we will customize based on role and URL.

So, to enable security just annotate class with @EnableWebSecurity and then write a rest resource…create a package controller and class ApplicationController.



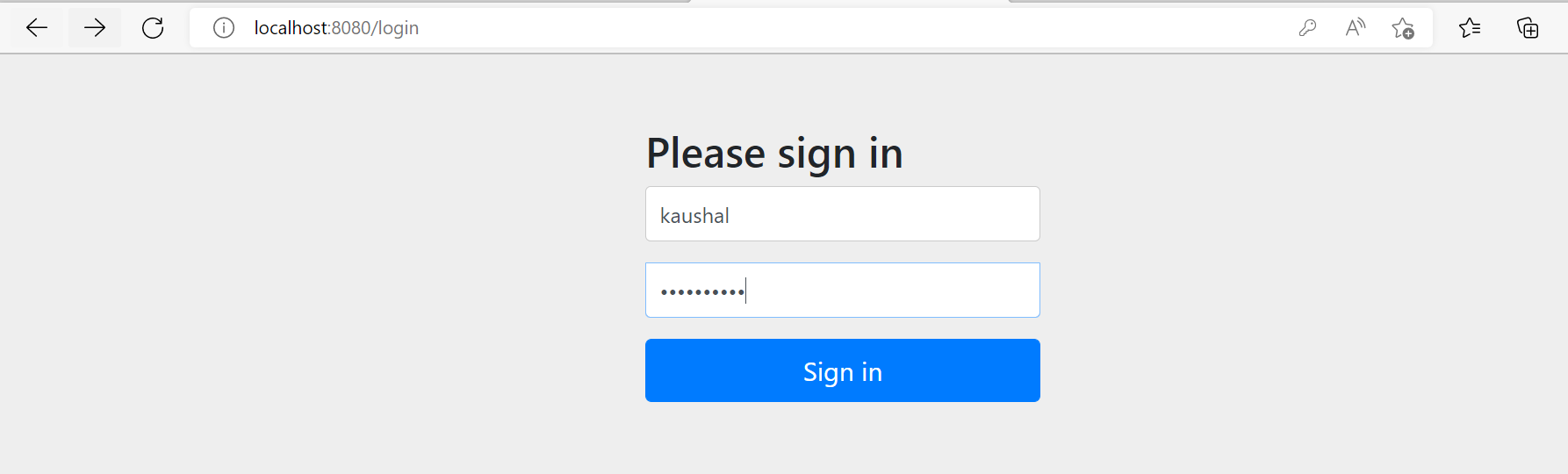


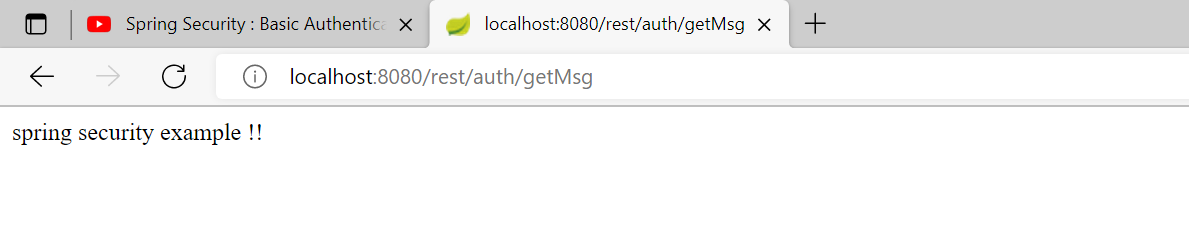
So, to use spring default security we need to mentioned username and password in application. properties.



Then run your application.

<http://localhost:8080/rest/auth/getMsg>





But there is a problem in this approach. Suppose if you are using it a local testing purpose then we can specify username and password in properties file, and we can check.

Suppose you want to implement in your application based on URL, suppose you want specific URL you want to apply security or some specific method you want to apply security, so it can’t be possible by default mechanism. When u need something like rile based security like few of the api can access by admin or access by the user. So, role-based security also you can’t mentioned in this properties file.

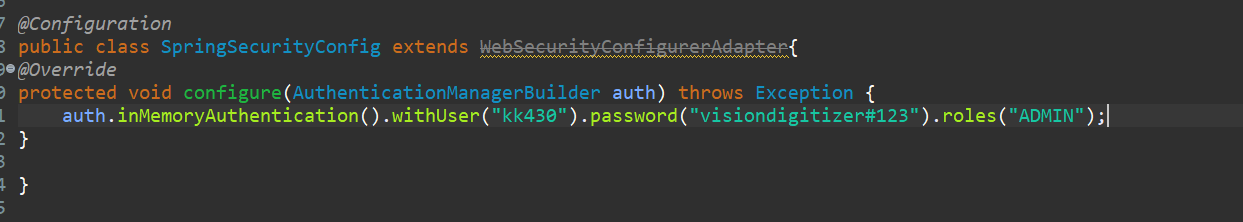
So, how we can do the role-based security and URL based security let’s check that.

So, lets write our Custom Configuration for Spring Security.

Now we will learn how to make security for-

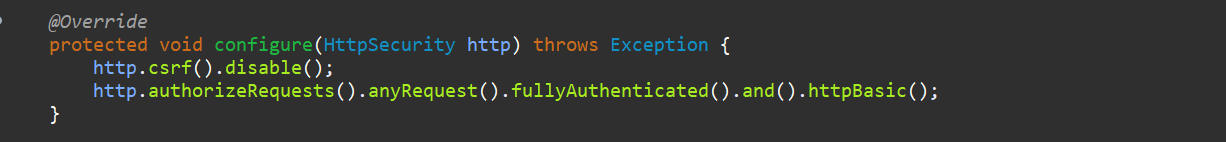
1. All the rest endpoint.
2. Based on specific URL.
3. Based on the specific Role.

Let’s do the code for that…



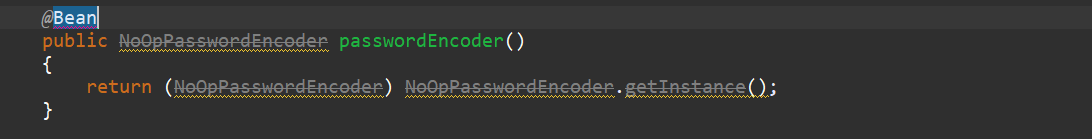
Now write a method where we will mention the cross origin in the sense in which are the method, we want to implement the security. So, lets override one more configure method…. Let me first disable the csrf token then now specify the method where you want to implement the security.

So now here for every request I want to implement the security.



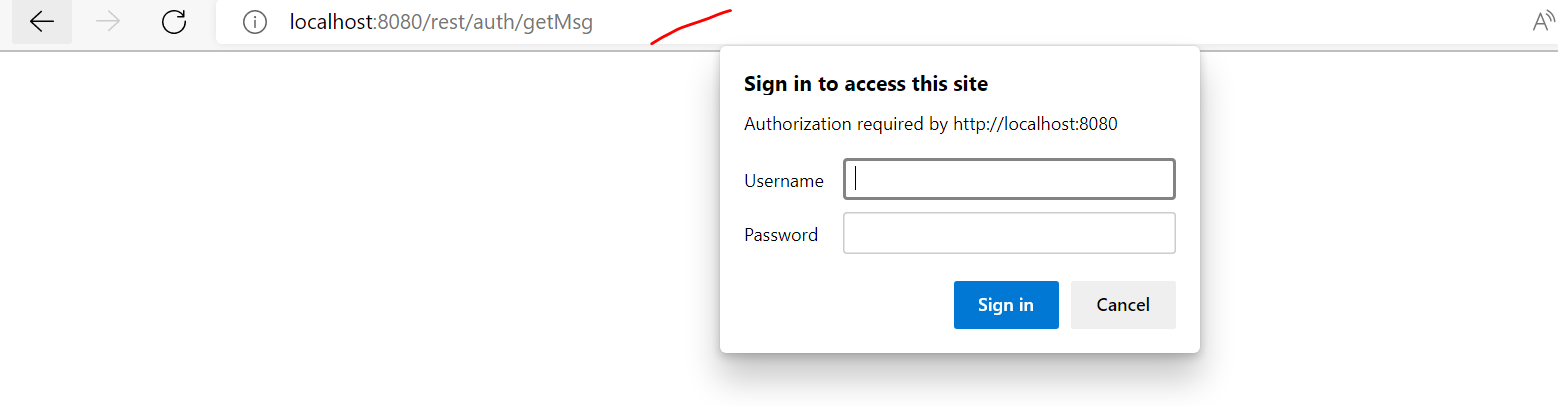
Now as we are using spring 2.0 so password expecting in encrypted format…

So lets add some no password output some class is there…so initialize the Bean.

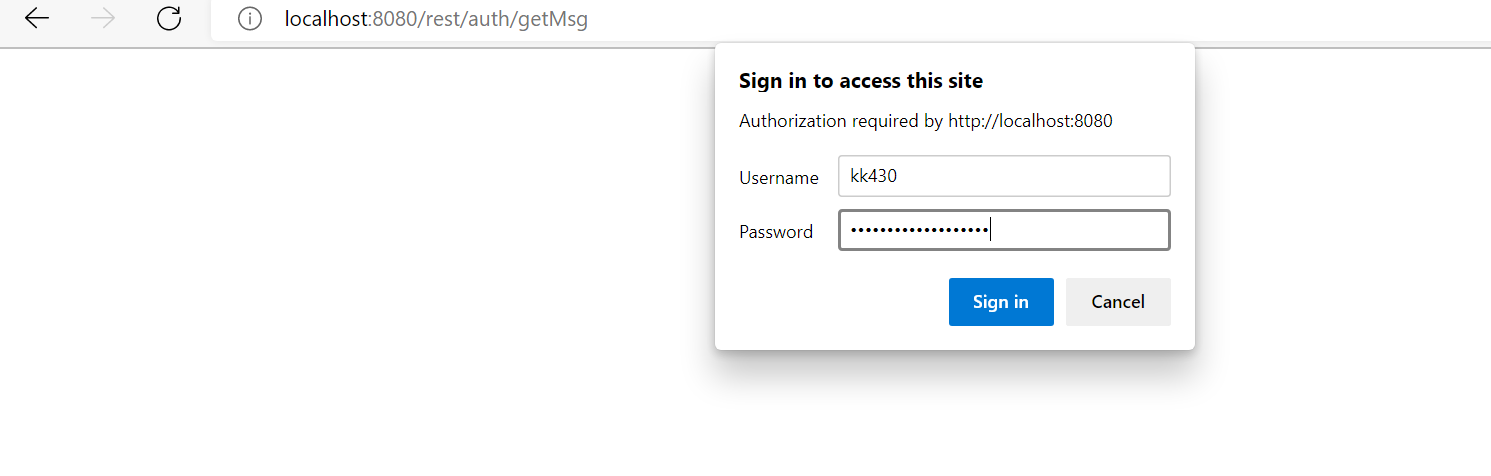


Now let’s run our application…as of now we are not using any role-based authentication.





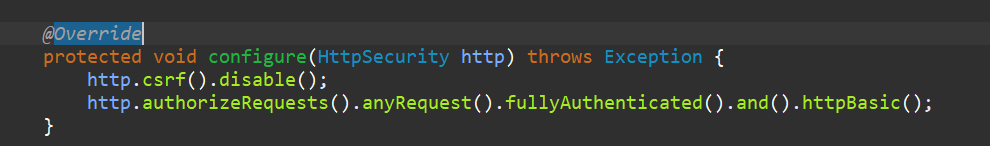
Once we start our application, we are getting a popup here…



Then we are getting the response…

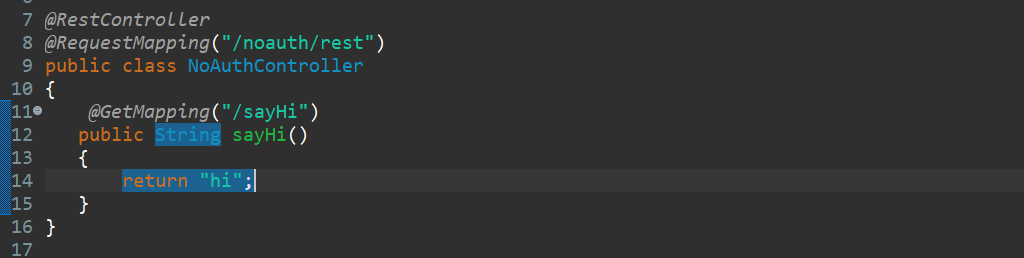


As here we mentioned authorized any request which is coming to our application …which is basic authentication. For each request I want security that’s what I write.



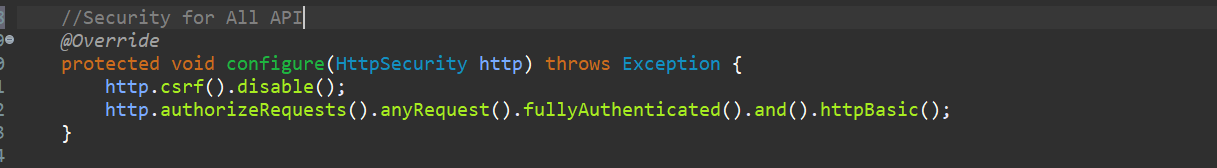
As we do have one more controller for that we don’t need any security…so for that case what we can do

For that we can do URL based security pattern.



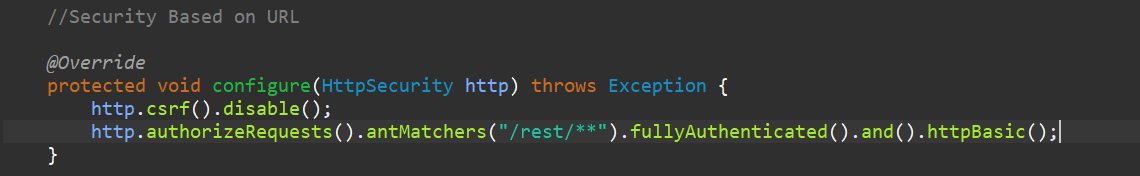
Like this if we have a bunch of controllers but we want security in specific controller method. Not for all the controller not for all the methods. So, in that case what we can do…**we can do using our URL Based Security.** Let’s comment this first…

**Below one is the Security for All API**

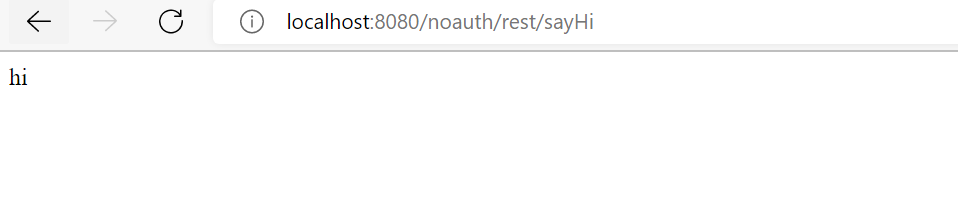


**Now we are going to write Security Based on URL**

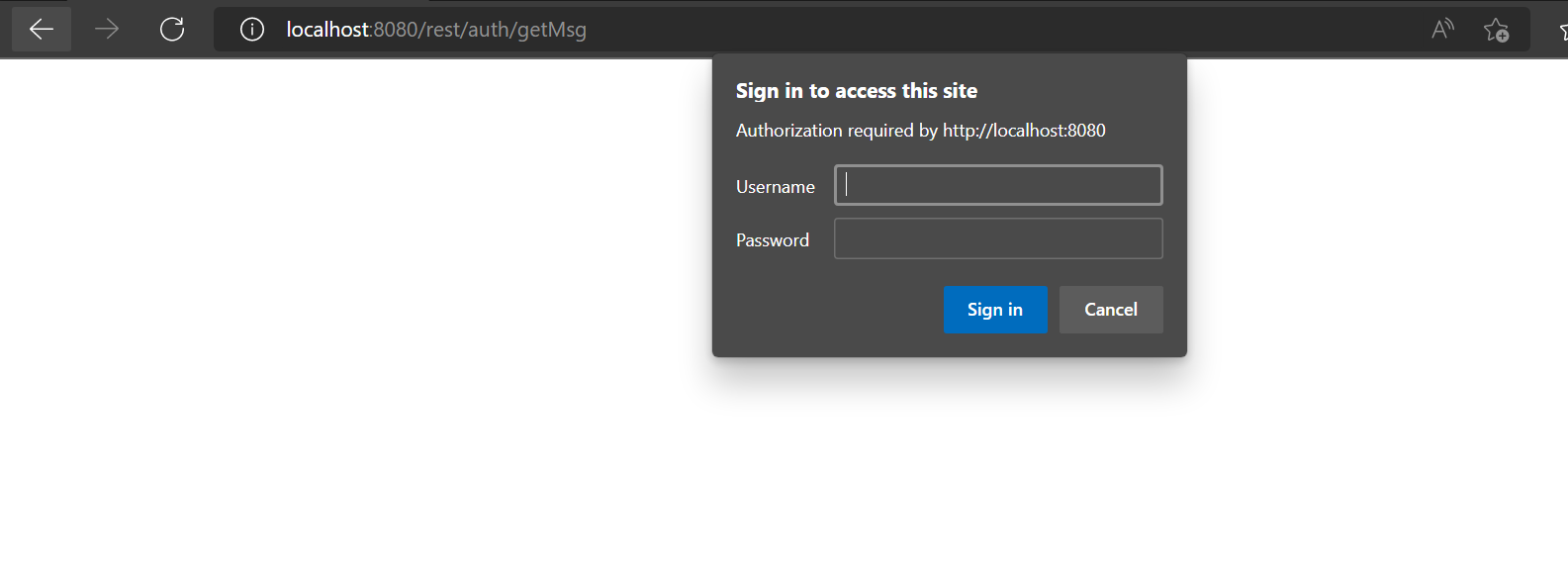
Now for all request coming with /rest URL for all that methods we are going to write a security…

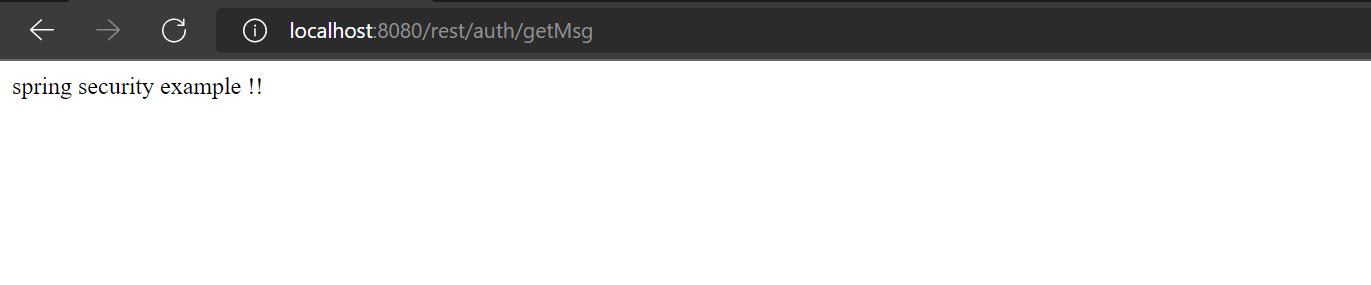


<http://localhost:8080/noauth/rest/sayHi>



http://[localhost:8080/rest/auth/getMsg](http://localhost:8080/rest/auth/getMsg)





For noAuth it’s not asking any security but for /rest kind of URL it asking security….so this is the way we can implement the URL based security…

Now we want the getMsg () method should only access by the User who is ADMIN not by the User. So that is the Role Based Security…

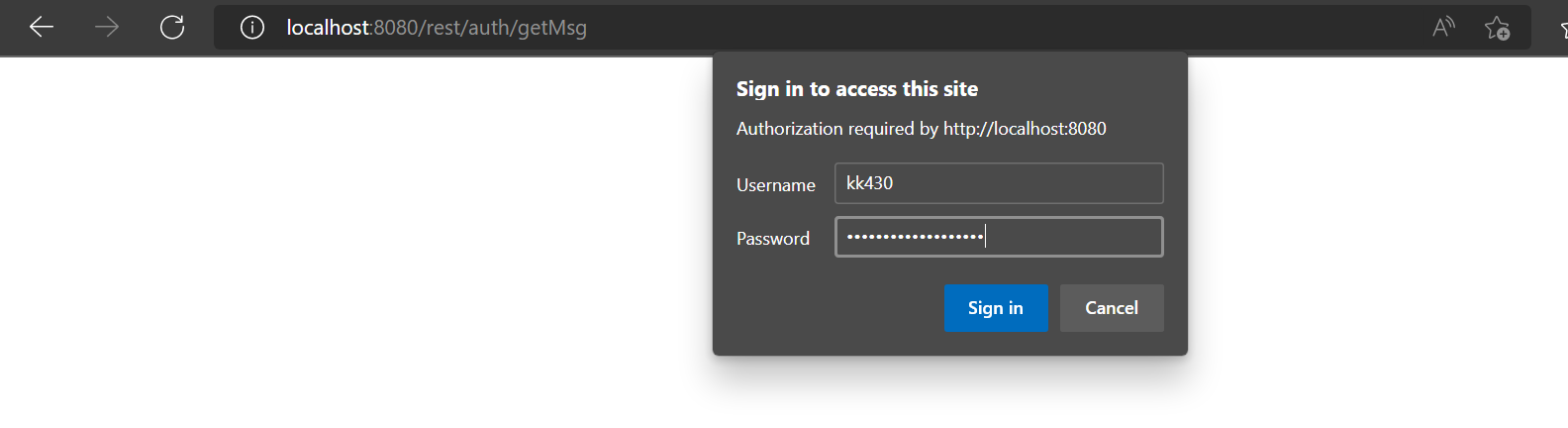
**So, lets implement the Role Based Security…**

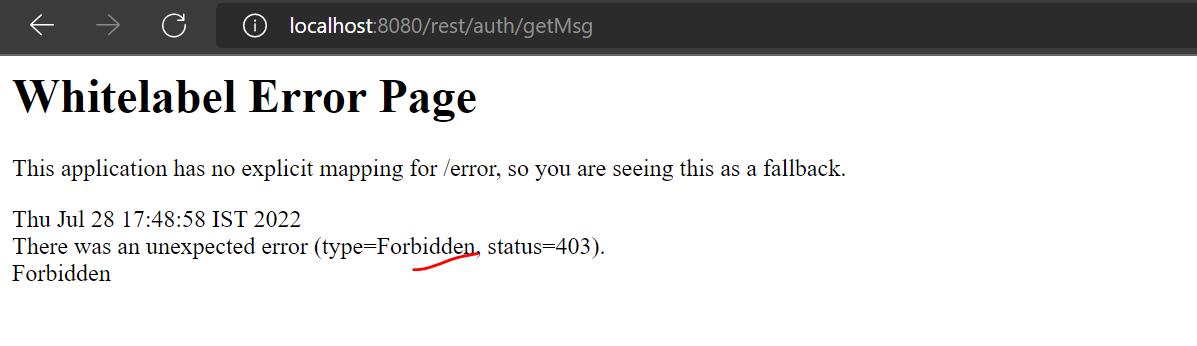
So, lets configure for that…

**We have created only one user whose role is admin lets create another user whose role is User.**

So, when any request coming with /rest URL will only authorized by Admin User only not by any other User.

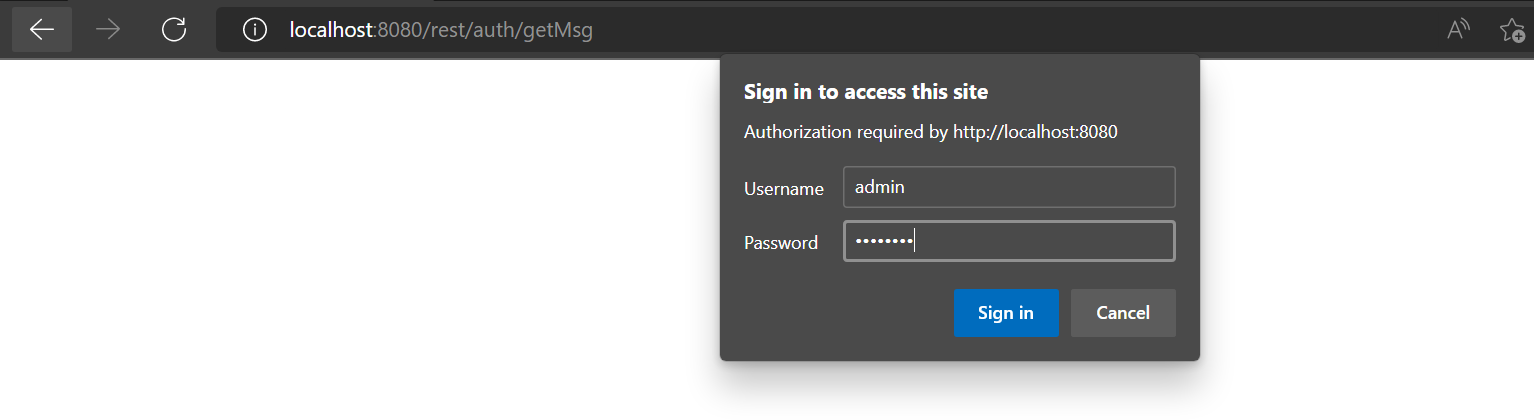
So, let me try with User access first then we will try with Admin Role….

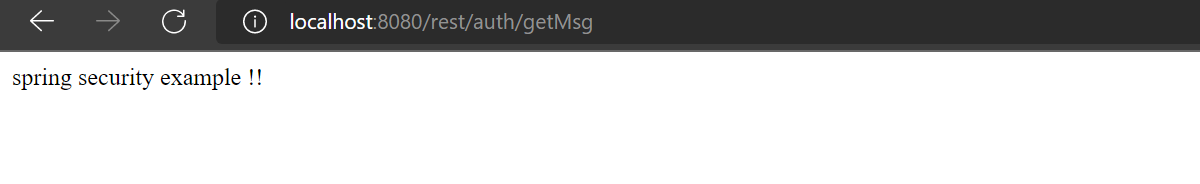




Forbidden means un-authorized access.

So, let me try with Admin Role….





**\*\*\*\*\*\*\*\*\*\*Some time you need to run rest endpoint in private Cognito windows\*\*\*\*\*\***