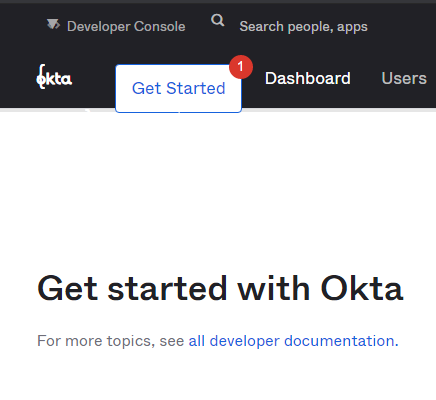
**Building Single Sign-on in Java using Okta**

**Objective:**

We will be using Okta and Spring Boot to implement single sign-on with two client applications and a single resource server.

**Prerequisites:**

Register and create a free Okta developer account [here](https://developer.okta.com/signup/)

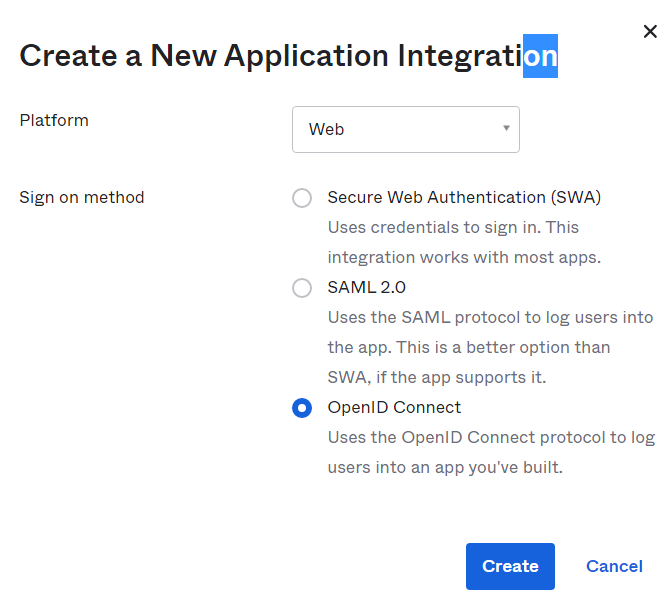


**Steps:**

Click on the developer console to access the admin panel

**Create Two OpenID Connect Applications**

In the Okta developer console, navigate to Applications and click Add Application. Then click Create new application . Choose Web and click create.



Populate the fields with these values:

Application name OIDC App 1

Login redirect URIs http://localhost:8080/login/oauth2/code/okta

Click Done. Scroll down and make a note of the Client ID and Client Secret. You’ll use those values shortly.Repeat these steps for your second application with these values:

Name OIDC App 2

Login redirect URIs http://localhost:8081/login/oauth2/code/okta

Click Done.

You’ll also need the Client ID and Client Secret from this OIDC application as well.

**Create two Users**

We will create two users in such a way that, the first user can only log into the first application (OIDC App 1) and the second user can log into both applications.

In the developer console, click on Directory > People and then click on Add Person. Fill out the form with the information for the first user using the information given below. Repeat this for the second user, also using the details below.

User2:

First Name Tanya

Last Name Tester

Username tanyaTester@mail.com

Primary Email [tanyaTester@mail.com](mailto:tanyaTester@mail.com)

Password Set by Admin

Password Value Test1234

User must change unchecked

password

User1:

First Name Amanda

Last Name Tester

Username amandaTester@mail.com

Primary Email amandaTester@mail.com

Password Set by Admin

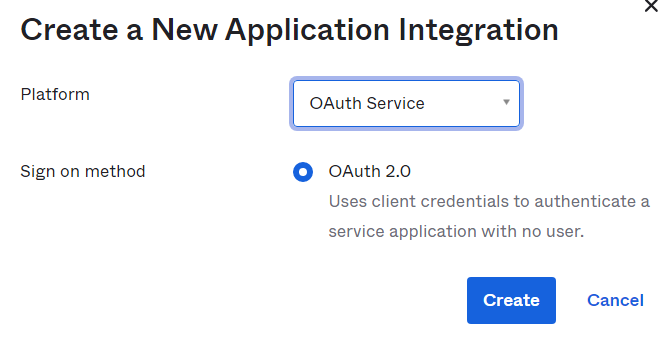
Password Value Test1234

User must change unchecked

password

**Create a Service Application for Your Resource Server**

In the Okta developer console, navigate to Applications and click Add Application> Create New App. Choose OAuth Service and click Create.



Populate the fields with these values:

Name OIDC Resource Server

Click Done.

Scroll down and copy the Client ID and Client Secret

**Create an Authorization Server**

Navigate to API > Authorization Servers. Click Add Authorization Server and fill in the values as follows:

Name OIDC Auth Server

Audience api://oidcauthserver

Description OIDC Auth Server

Click Done and then click the Claims tab. In Claims, click Add Claim, fill in the fields with the values for Claim 1 below, and click Create. You can leave any value not mentioned below as default. When done, repeat and create a second claim with the values under Claim 2 below.

Claim 2

Name userEmail

Include in token type Access Token Always

Value user.email

Include in -> email

The following scopes

Claim 1

Name fullName

Include in token type Access Token Always

Value user.fullName

Include in -> profile

The following scopes

Next, you will add an Access Policy for the first application. This application will allow both users access to it. Click the Access Policies tab, Add New Access Policy, fill in the fields with these values, and click Create Policy.

This binds the policy to your OIDC app.

Name OIDC App 1

Description OIDC App 1

Assign to the following clients

Assign to clients OIDC App 1

Next click Add Rule. Set OIDC App 1 for the Rule Name field. Deselect all the grant types except for Authorization Code and click Create Rule.

Next, you will add an Access Policy for the second application. This application will allow only the second user, Tanya Tester to access it. From the Access Policies tab, Add Policy, fill in the fields with these values and click Create Policy.

Name OIDC App 2

Description OIDC App 2

Assign to the following clients

Assign to clients OIDC App 2 Name OIDC App 2

Description OIDC App 2

Assign to the following clients

Assign to clients OIDC App 2

Next click Add Rule. Set OIDC App 2 for the Rule Name field. Deselect all the grant types except for Authorization Code. Find the User Is section and select the second radio button labeled “Assigned the app and a member of one of the following:” In the Users box that appears, start typing Tanya and select Tanya Tester from the list. This tells it that only this user can log in to the OIDC App 2 application.

Click Create Rule.

Click the Settings tab and copy the Issuer URL.

**Create the OAuth 2.0 Resource App**

Clone [this](https://github.com/oktadeveloper/okta-java-spring-sso-example) repo

git clone https://github.com/oktadeveloper/okta-java-spring-sso-example.git

cd okta-java-spring-sso-example/oauth2-resource-server

Configure the resource application with the values from the “OIDC Resource Server” application you created in Okta. Open the src/main/resources/application.properties file.

okta.oauth2.issuer={issuerUri}

okta.oauth2.clientId={clientId}

okta.oauth2.clientSecret={clientSecret}

okta.oauth2.audience=api://oidcauthserver

server.port=8082

Replace the {clientId} and {clientSecret} with the ones you wrote down for the resource server above. The {issuerUri} is the Issuer URI for the authorization server you created above.

Open a shell and launch an instance of the resource server using Maven.

Command*: ./mvnw spring-boot:run*

It will now be listening on port 8082.

**Create the OAuth 2.0 Client App**

When you run the client application, you will run it first for OIDC App 1, which has the profile scope set. You will also run it for OIDC App 2, but for this application, you will run it with both the profile and email scopes set. For both instance of the client applications, start with the code in the oauth2-client directory of the example project. The complete code for this application is in the *SingleSignOnApplication* class.

*src/main/java/com/okta/examples/sso/SingleSignOnApplication.java*

**Configure the Client Apps Using Spring Run Profiles**

Now you need to configure the two different instances of the client application in the oauth2-client project folder.

If you look under *oauth2-client/src/main/resources,* you’ll see three .properties files.

*application.properties* is common to all three profiles application-client1.properties has the config values for client 1 application-client2.properties has the config values for client 2

Open oauth2-client/src/main/resources/application.properties and fill in the Issuer URI for the resource server you created above. To find the Issuer URI (if you didn’t write it down) go to API and Authorization Servers. Look in the table next to OIDC Auth Server under Issuer URI.

okta.oauth2.clientId={yourClient1Id}

okta.oauth2.clientSecret={yourClient1Secret}

okta.oauth2.scopes=openid,profile

server.port=8080

okta.oauth2.issuer={yourIssuerUri}

resourceServer.url=http://localhost:8082

Open oauth2-client/src/main/resources/application-client1.properties and fill in the Client ID and Client Secret for the first OIDC client app.

okta.oauth2.clientId={yourClient2Id}

okta.oauth2.clientSecret={yourClient2Secret}

okta.oauth2.scopes=openid,profile,email

server.port=8081 okta.oauth2.clientId={yourClient2Id}

okta.oauth2.clientSecret={yourClient2Secret}

okta.oauth2.scopes=openid,profile,email

server.port=8081 okta.oauth2.clientId={yourClient2Id}

okta.oauth2.clientSecret={yourClient2Secret}

okta.oauth2.scopes=openid,profile,email

server.port=8081

Open oauth2-client/src/main/resources/application-client2.properties and fill in the Client ID and Client Secret for the second OIDC client app.

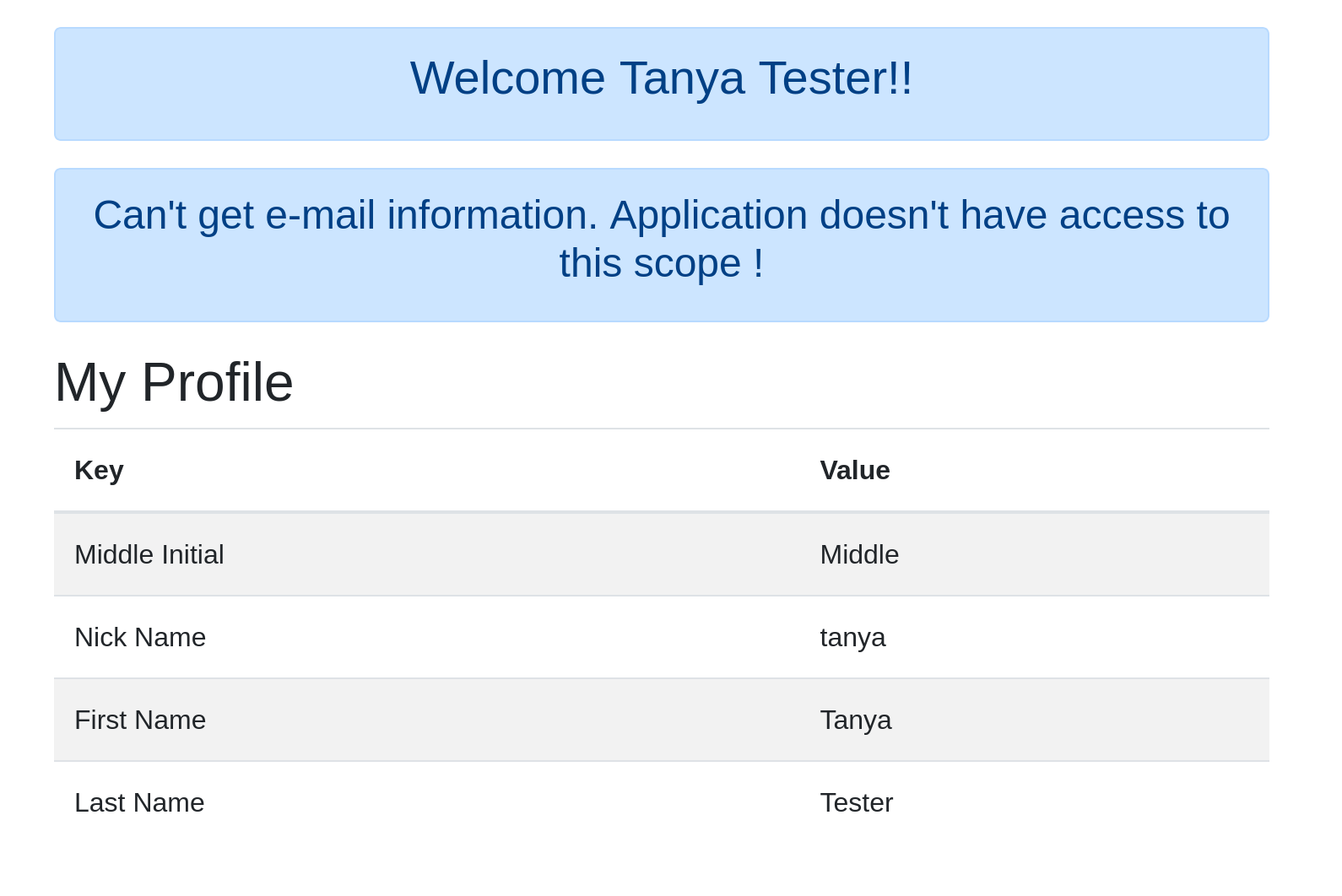
The last step is to run two instances of this client application. Run the following commands in two separate shell windows. This loads a client with each run profile.

Run client app 1 at http://localhost:8080:

*./mvnw spring-boot:run -Dspring-boot.run.profiles=client1*

Run client app 2 at http://localhost:8081:

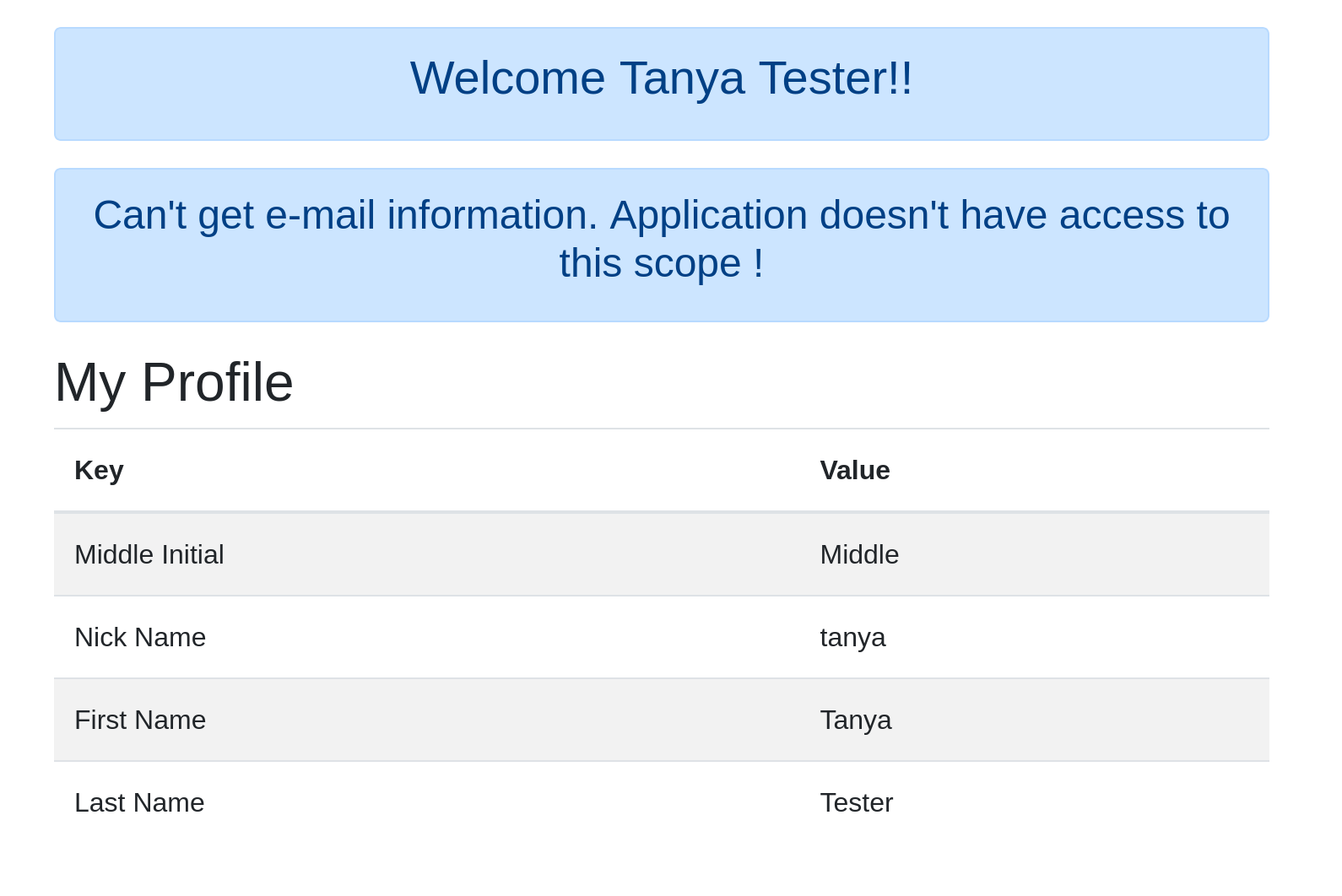
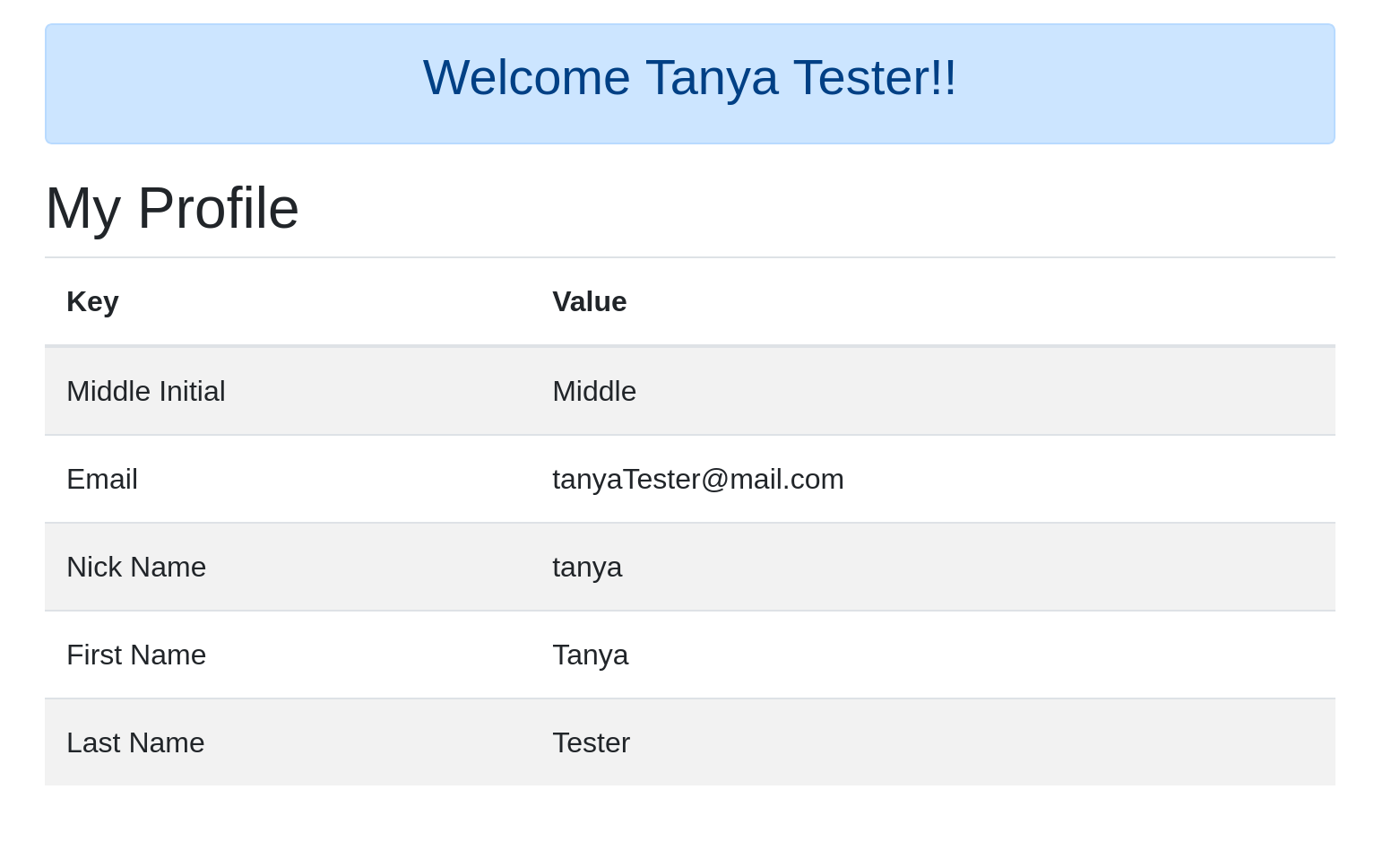
*./mvnw spring-boot:run -Dspring-boot.run.profiles=client2*



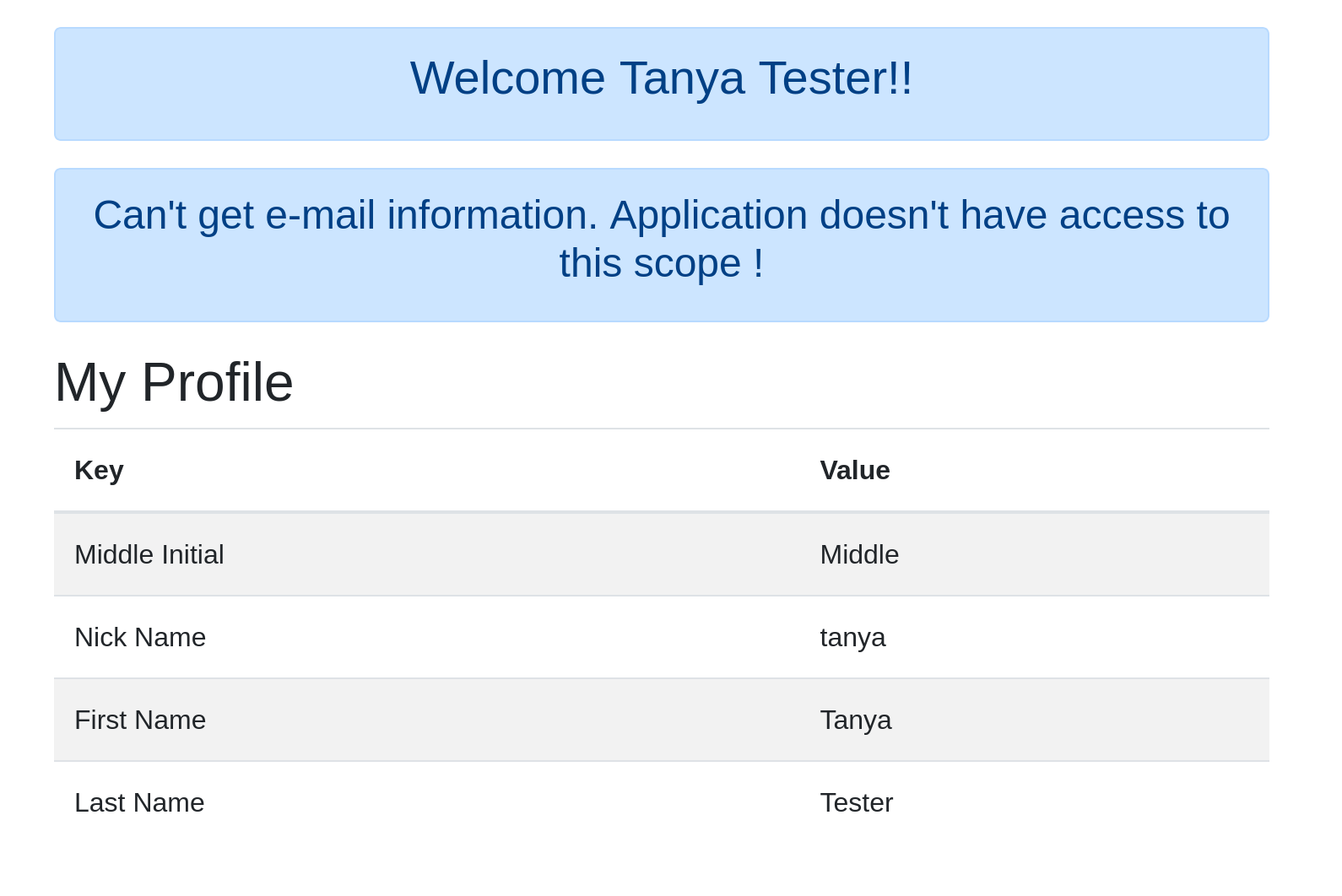
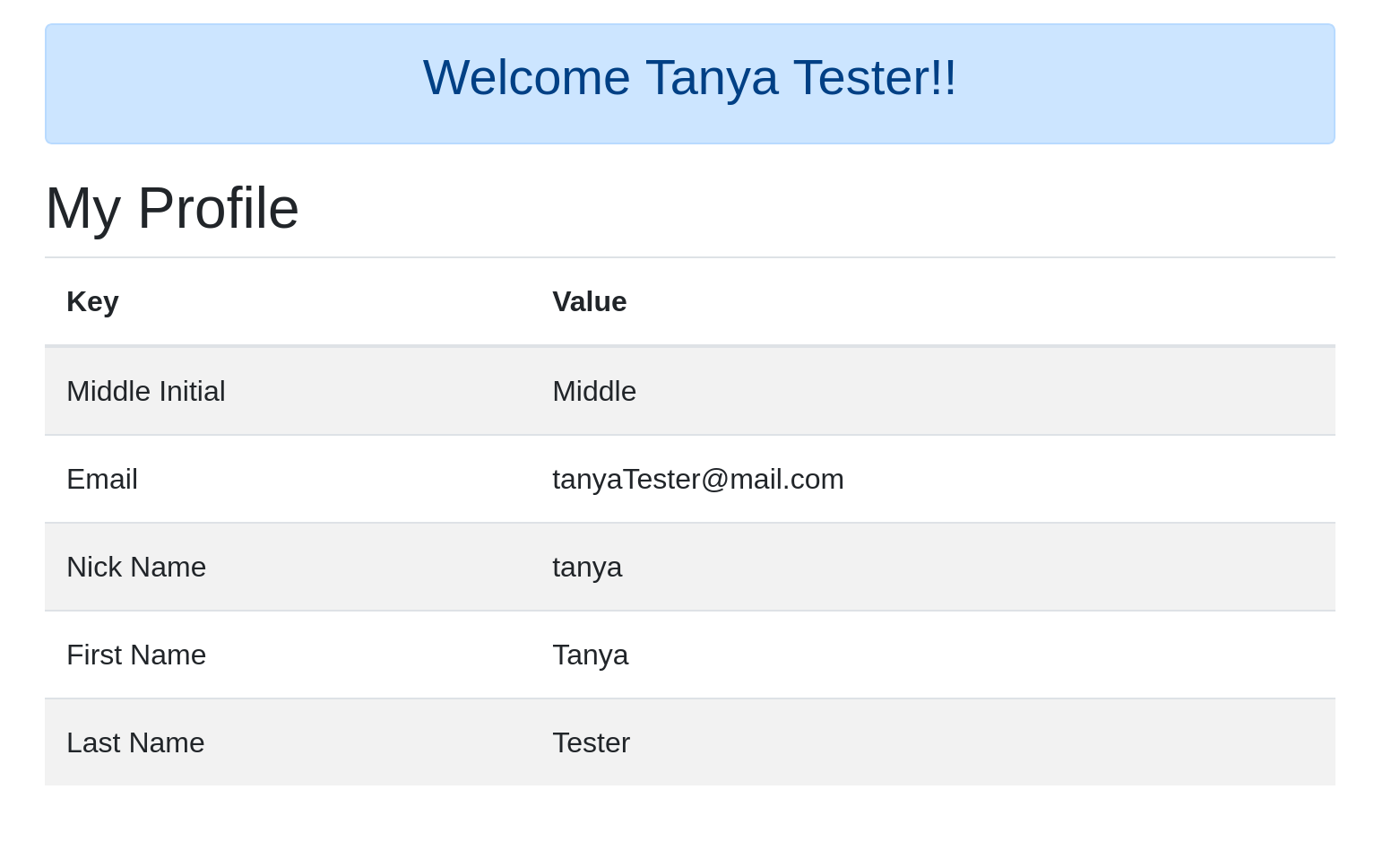
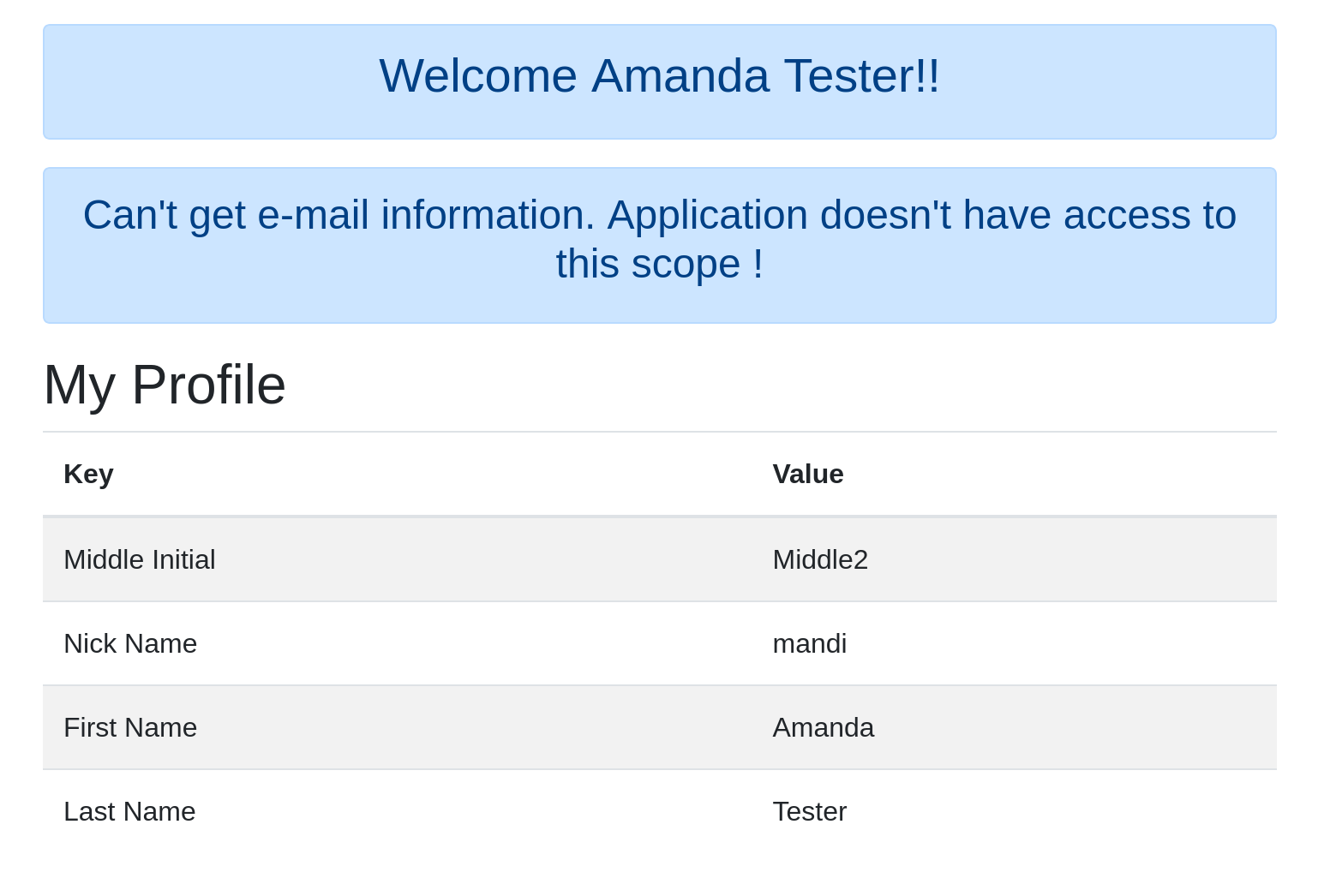
**Test Your Java Single Sign-On**

Open a new incognito browser window and enter the URL http://localhost:8080. This is the URL to the first application OIDC App 1. Log in with the tanyaTester@mail.com user. You should be able to successfully log in!

Next, you can change the URL to http://localhost:8081. This is the URL to the second application OIDC App 2. You will notice that you don’t have to log in again. This is because you already logged into OIDC App 1 and this is single sign-on!

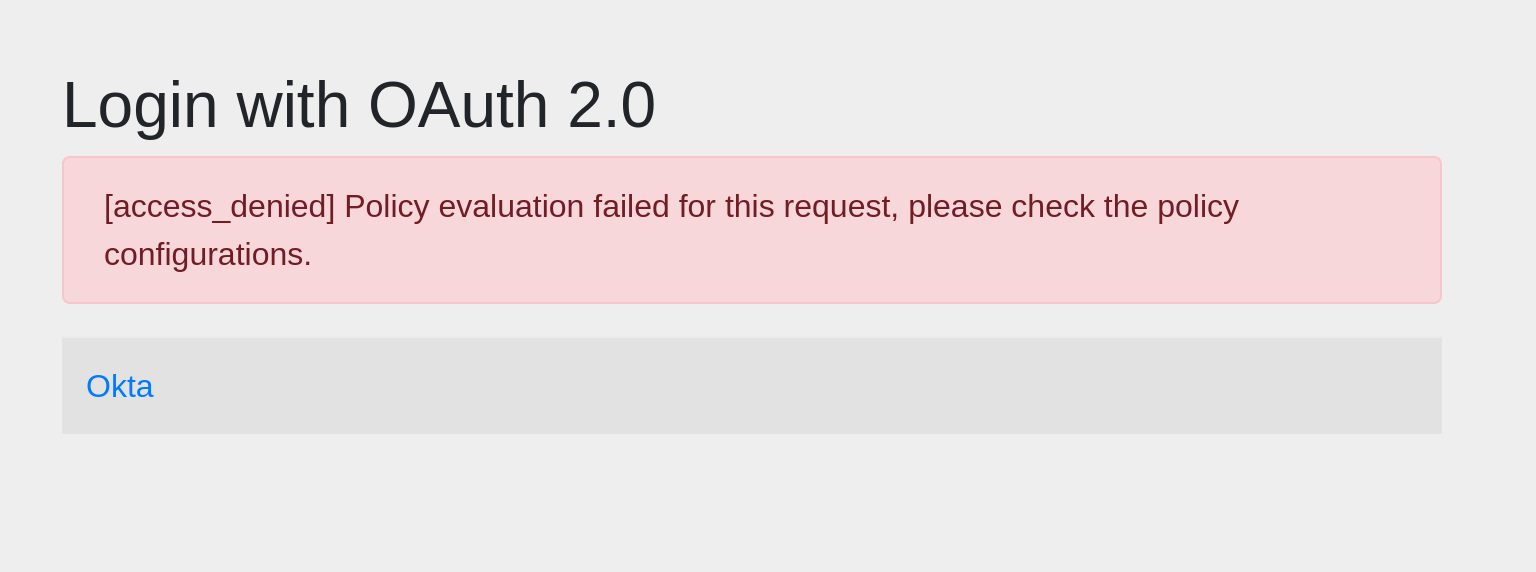


**Test Your Access Policy**



Log in to each application with the amandaTester@mail.com user. Open a new incognito browser window and enter the URL http://localhost:8080. Log in with the amandaTester@mail.com user. You should be able to successfully login!

Close that browser window and open a new incognito browser window and enter the URL <http://localhost:8081>. Log in with the amandaTester@mail.com user. You will get an “access denied” error.

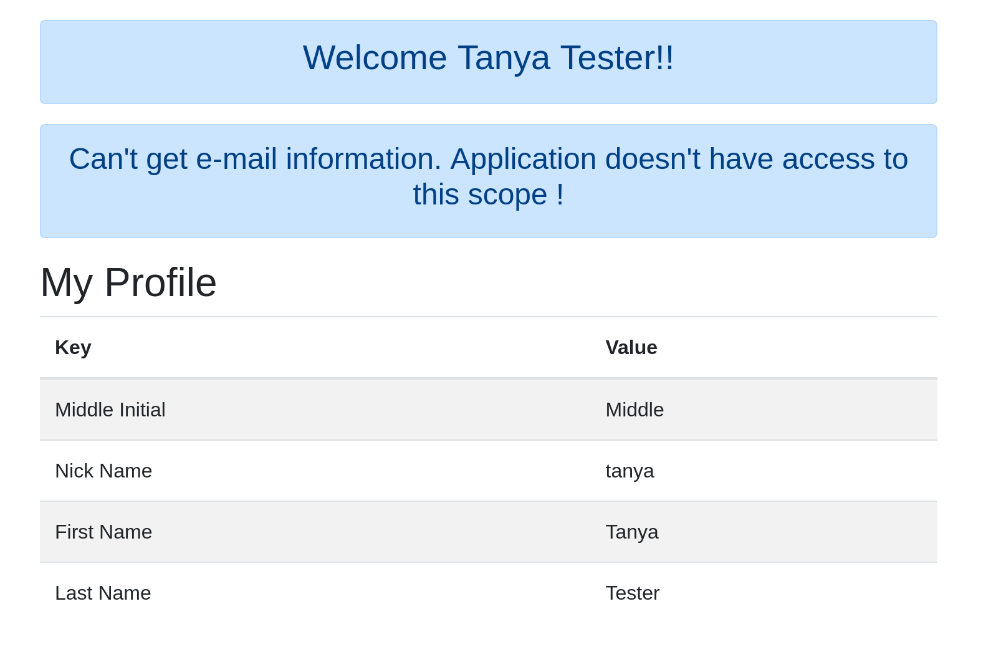


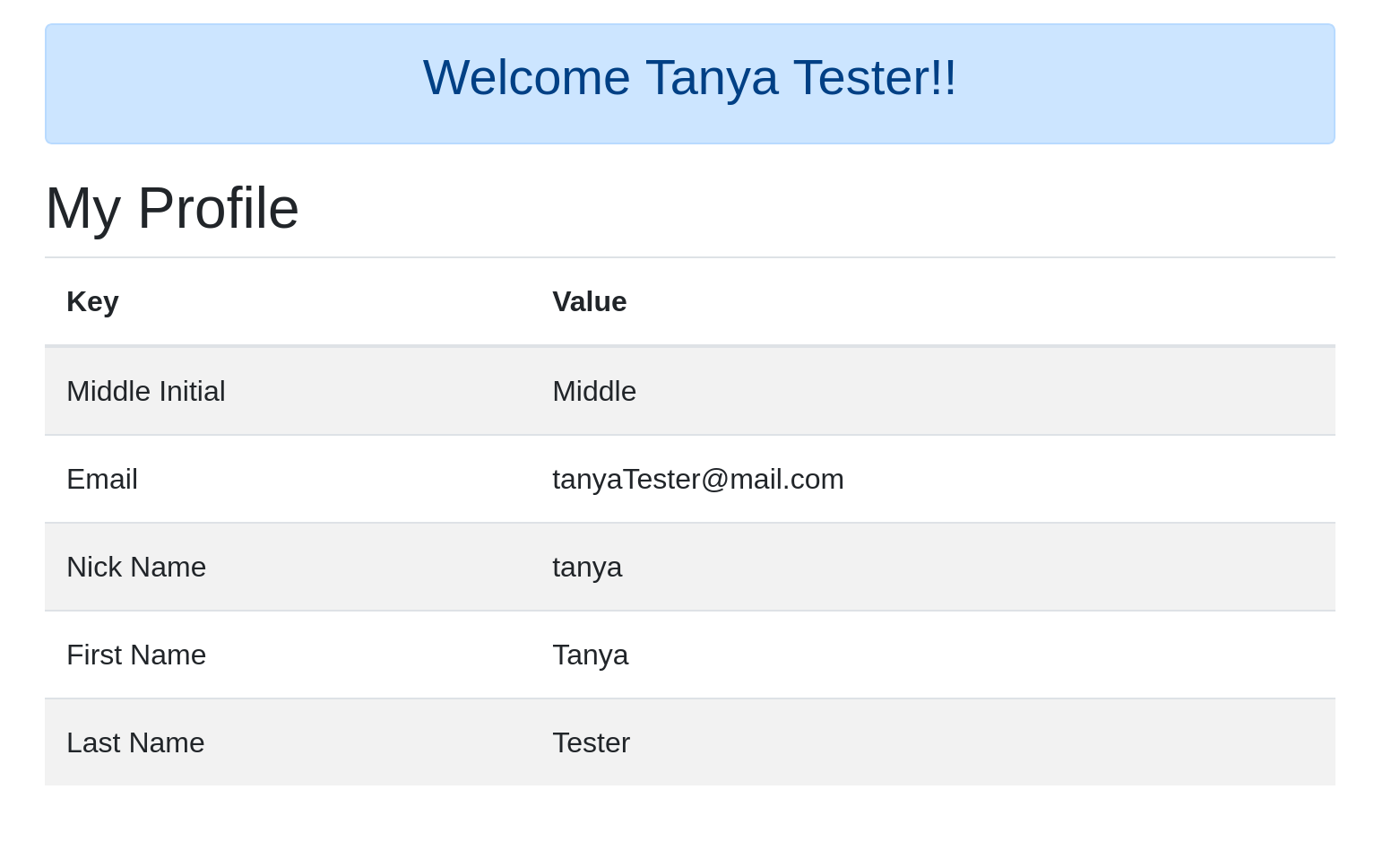
This error is because you set up the access policy so that only Tanya Tester could log in to OIDC App 2.

**Test Your Scope Authorization**

Lastly, you are going to test how the resource server handled the authorization of each of the applications. Open a new incognito browser window and enter the URL http://localhost:8080. Log in with the tanyaTester@mail.com user. You should be able to successfully log in.

Notice a message saying that the application does NOT have access to the email information. This is because this instance of the application is NOT running with the email scope.





Next, you can change the URL to <http://localhost:8081>. When you go to this page, you will notice that you do NOT see the message about not having access to the email. Instead, you will see the email as part of the profile information.

Reference:

https://developer.okta.com/blog/2020/01/29/java-single-sign-on