

CA 2E / CA Plex Worldwide Developers Conference 2023

Securing CA Plex Web Apps

Andrew Leggett & Lily Taharudin – CM First

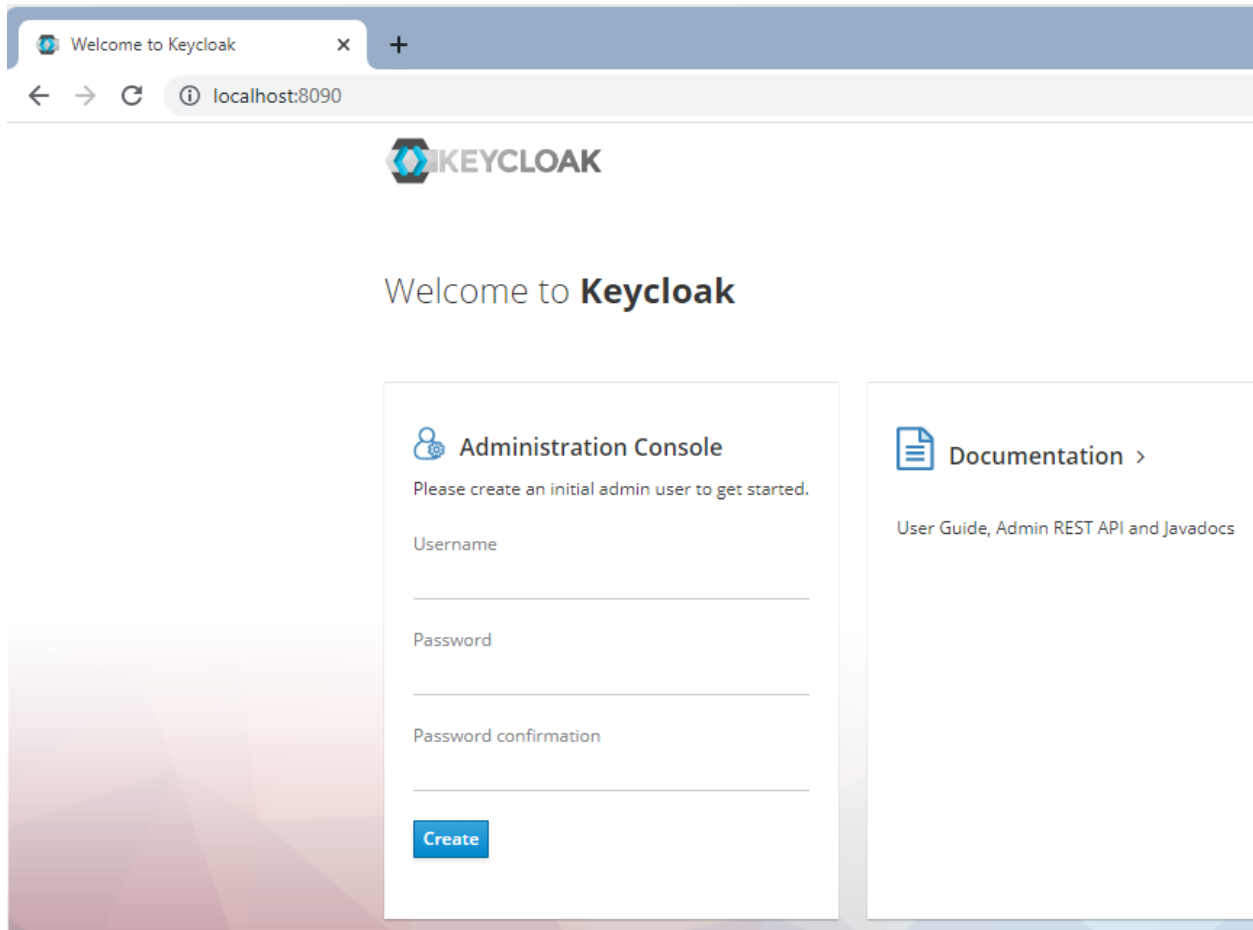
Exercise One

startKeycloak.bat

```
C:\WINDOWS\system32\cmd. X + v
C:\Conference 2023 Workshops>.\keycloak-21.0.2\bin\kc.bat start-dev --http-port=8090
2023-04-29 11:50:33,063 INFO [org.keycloak.quarkus.runtime.hostname.DefaultHostnameProvider] (main) Hostname settings:
Base URL: <unset>, Hostname: <request>, Strict HTTPS: false, Path: <request>, Strict BackChannel: false, Admin URL: <uns
et>, Admin: <request>, Port: -1, Proxied: false
2023-04-29 11:50:34,275 WARN [io.quarkus.agroal.runtime.DataSources] (main) Datasource <default> enables XA but transac
tion recovery is not enabled. Please enable transaction recovery by setting quarkus.transaction-manager.enable-recovery=
true, otherwise data may be lost if the application is terminated abruptly
2023-04-29 11:50:35,454 WARN [org.infinispan.PERSISTENCE] (keycloak-cache-init) ISPN000554: jboss-marshalling is deprec
ated and planned for removal
2023-04-29 11:50:35,507 WARN [org.infinispan.CONFIG] (keycloak-cache-init) ISPN000569: Unable to persist Infinispan int
ernal caches as no global state enabled
2023-04-29 11:50:35,535 INFO [org.infinispan.CONTAINER] (keycloak-cache-init) ISPN000556: Starting user marshaller 'org
.infinispan.jboss.marshalling.core.JBossUserMarshaller'
2023-04-29 11:50:39,193 INFO [org.keycloak.broker.provider.AbstractIdentityProviderMapper] (main) Registering class org
.keycloak.broker.provider.mappersync.ConfigSyncEventListener
2023-04-29 11:50:39,375 INFO [org.keycloak.connections.infinispan.DefaultInfinispanConnectionFactory] (main) No
de name: node_480562, Site name: null
2023-04-29 11:50:40,524 INFO [io.quarkus] (main) Keycloak 21.0.2 on JVM (powered by Quarkus 2.13.7.Final) started in 9.
207s. Listening on: http://0.0.0.0:8090
2023-04-29 11:50:40,525 INFO [io.quarkus] (main) Profile dev activated.
2023-04-29 11:50:40,525 INFO [io.quarkus] (main) Installed features: [agroal, cdi, hibernate-orm, jdbc-h2, jdbc-mariadb
, jdbc-mssql, jdbc-mysql, jdbc-oracle, jdbc-postgresql, keycloak, logging-gelf, micrometer, narayana-jta, reactive-route
s, resteasy, resteasy-jackson, smallrye-context-propagation, smallrye-health, vertx]
2023-04-29 11:50:40,528 WARN [org.keycloak.quarkus.runtime.KeycloakMain] (main) Running the server in development mode.
DO NOT use this configuration in production.
```

Running on port 8090

Create Keycloak Administrator



The screenshot shows a web browser window with the address bar displaying 'localhost:8090'. The page title is 'Welcome to Keycloak'. The Keycloak logo is prominently displayed at the top. Below the logo, the text 'Welcome to Keycloak' is shown. The main content area is divided into two columns. The left column contains the 'Administration Console' section, which includes a message: 'Please create an initial admin user to get started.' Below this message are three input fields labeled 'Username', 'Password', and 'Password confirmation'. A blue 'Create' button is positioned at the bottom of this section. The right column contains the 'Documentation' section, which includes a link to 'User Guide, Admin REST API and Javadocs'.

Welcome to Keycloak

Administration Console

Please create an initial admin user to get started.

Username

Password

Password confirmation

Create

Documentation >

User Guide, Admin REST API and Javadocs

First time in, create administrator

User: kadmin

Password: kadmin



Administration Console

Please create an initial admin user to get started.

Username

kcadmin

Password

Password confirmation

Create

Welcome to **Keycloak**

User created



Administration Console >

Centrally manage all aspects of the Keycloak server

Click on "Administration Console >"

Enter user/password



Sign in to your account

Username or email

kcadmin

Password

.....

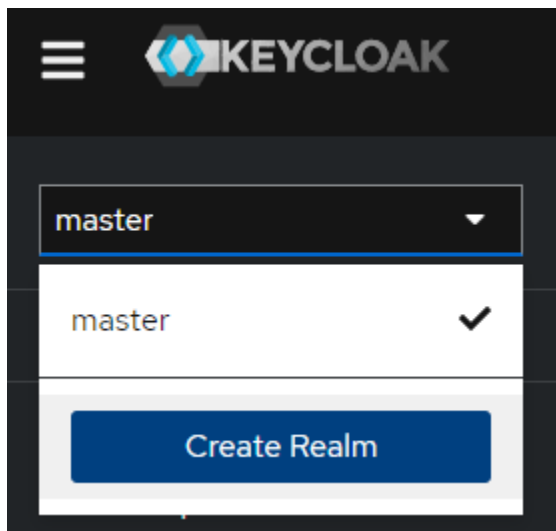
Sign In

Create “SecurityWorkshop2023” Realm

In Keycloak, a realm is a container for a set of users, applications, and security configurations. A realm defines a boundary for authentication and authorization of its users and clients, providing a secure environment for the components within it.

When a user logs in to Keycloak, they log in to a specific realm. Each realm has its own set of users, roles, clients, and other security components. By creating multiple realms, it's possible to separate different applications or user groups from each other, each with its own authentication and authorization rules.

By default, Keycloak provides a ‘master’ realm. We’ll create a new realm by clicking on the dropdown next to ‘master’, then click on the ‘Create Realm’ button.



Name the domain “SecurityWorkshop2023” and press Create.

Create realm

A realm manages a set of users, credentials, roles, and groups. A user belong

Resource file

Drag a file here or browse to upload

1

Upload a JSON file

Realm name *

SecurityWorkshop2023

Enabled



On

Create

Cancel



Welcome to SecurityWorkshop2023

If you want to leave this page and manage this realm, please click the corresponding menu items in the left navigation bar.

Create Client for Web Store

In the context of the Keycloak identity and access management system, a client is a representation of a web application or service that wants to use Keycloak for authentication and authorization.

A Keycloak client defines a set of configuration options and settings that allow the application to interact with Keycloak in a secure way. These options include things like the client ID, client secret, redirect URIs, authentication flow, and security settings.

Once a client is registered in Keycloak, it can obtain access tokens and refresh tokens that can be used to authenticate and authorize requests to protected resources. The client can also interact with Keycloak's APIs to perform various tasks, such as registering new users, managing roles and permissions, and revoking tokens.

We are going to create a client named 'webstore' to control access to our web application.

Clients

Clients are applications and services that can request authentication of a user

Clients list	Initial access token	Client registration
--------------	----------------------	---------------------

→ [Create client](#) [Import client](#)

Client ID	Name
account	\$_client_account}

Client type ?	OpenID Connect
Client ID * ?	WebStore
Name ?	WebStore
Description ?	Conference 2023 Product Store web application
Always display in UI ?	<input type="checkbox"/> Off

Set ClientID and Name to WebStore. Click on '?' to learn what any setting is.

Click 'Next'

Leave every other option as its default value.

Client authentication ?	<input type="checkbox"/> Off	
Authorization ?	<input type="checkbox"/> Off	
Authentication flow	<input checked="" type="checkbox"/> Standard flow ?	<input checked="" type="checkbox"/> Direct access grants ?
	<input type="checkbox"/> Implicit flow ?	<input type="checkbox"/> Service accounts roles ?
	<input type="checkbox"/> OAuth 2.0 Device Authorization Grant ?	
	<input type="checkbox"/> OIDC CIBA Grant ?	

Click 'Next'

Root URL ?	<input type="text"/>
Home URL ?	<input type="text"/>
Valid redirect URIs ?	<div><input type="text" value="http://localhost:8080/storeweb/*"/> <input type="text" value="http://localhost:8080/storeAPI/*"/> + Add valid redirect URIs</div>
Valid post logout redirect URIs ?	<div><input type="text"/> + Add valid post logout redirect URIs</div>
Web origins ?	<div><input type="text" value="http://localhost:8080/*"/> + Add web origins</div>

[Save](#) [Back](#) [Cancel](#)

In Keycloak, redirect URLs are used to define where the user should be redirected after authentication or logout. These URLs can be set at the client level or globally for a realm.

When a user tries to access a protected resource on a client application that requires authentication, the application will redirect the user to the Keycloak server login page. After the user logs in successfully, Keycloak will redirect the user back to the original URL requested by the application.

Redirect URLs are also used for logout. When a user logs out of a client application, the application will redirect the user to the Keycloak server logout page. After the user is successfully logged out, Keycloak will redirect the user back to the URL specified by the application.

Redirect URLs can be configured to include query parameters and other information, allowing for a more customized user experience.

We have 2 applications to secure, 'storeweb' and 'storeAPI'.

Add the following URLs to the 'Valid redirect URIs' value'

- http://localhost:8080/storeweb/*
- http://localhost:8080/storeAPI/*

Web origins are URLs from which requests to the Keycloak server can be made. When a client application redirects a user to the Keycloak server for authentication or authorization, Keycloak will only redirect the user back to URLs that are listed as valid redirect URLs or web origins for that client. This is an important security measure to prevent unauthorized access to client applications.

Set the web origin to http://localhost:8080/*

Click 'Save'.

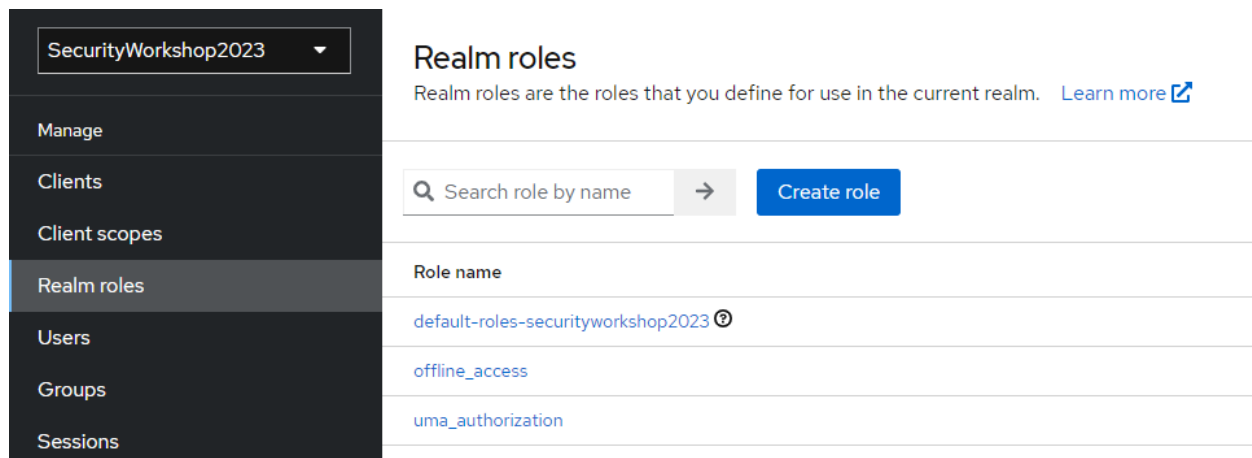
Create Roles

We want to control access to various resources within our web applications. For example, if we have a web application that allows someone to edit an employee's salary, we would only want certain members of the HR Department to access that. We can control that by assigning 'Roles', in this case we could create a role named 'HR Administrator' and assign it to certain users. We can then configure our web application to only allow users with specific roles to access specific resources.

In Keycloak, there are two types of roles: realm roles and client roles. Realm roles are global and can be assigned to users across all client applications within the realm. Client roles, on the other hand, are specific to a particular client application within a realm.

Roles can be managed and assigned through the Keycloak Admin Console or through the Keycloak REST API. Once a user or a client application is assigned a role, they inherit the permissions associated with that role. This allows for fine-grained access control and enables administrators to easily manage permissions and access levels for users and client applications.

For this exercise, we'll manage our roles from the 'Realm roles' menu option.



SecurityWorkshop2023

Manage

Clients

Client scopes

Realm roles

Users

Groups

Sessions

Realm roles

Realm roles are the roles that you define for use in the current realm. [Learn more](#)

Search role by name → Create role

Role name
default-roles-securityworkshop2023
offline_access
uma_authorization

We'll create the following roles:

- SiteAdmin – Access all
- Shopper – Access Products
- Registrant – Can register customers and view products.

[Realm roles](#) > [Create role](#)

Create role

Role name *	<input type="text" value="Shopper"/>
Description	<input type="text" value="Can browse products"/>

Enter the details above and click Save.

[Realm roles](#) > [Create role](#)

Create role

Role name *	<input type="text" value="Registrant"/>
Description	<input type="text" value="Can view and register new customers"/>

Enter the details above and click Save.

[Realm roles](#) > [Role details](#)

SiteAdmin

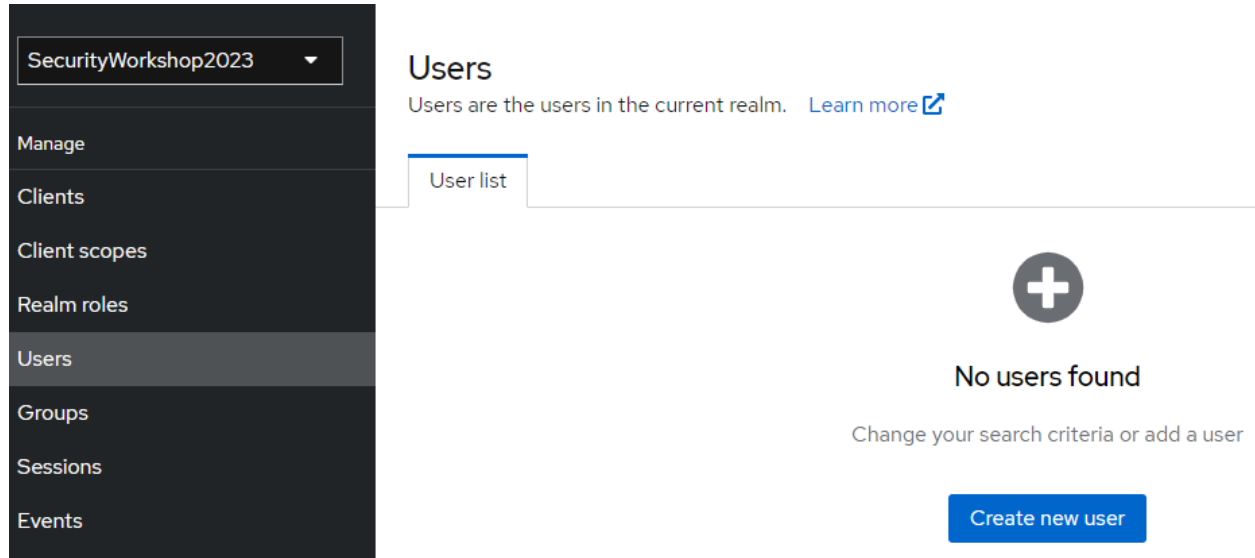
	<div>Details</div>	<div>Attributes</div>	<div>Users in role</div>
Role name	<input type="text" value="SiteAdmin"/>		
Description	<input type="text" value="Access to the whole site"/>		

Enter the details above and click Save.

Create Users

Keycloak users can be created manually or imported from external sources, such as LDAP or Active Directory. We'll create some users manually.

Select 'Users' from the menu and select 'Create new user'.



Create the user 'dave' and enter the values below.

[Users](#) > [Create user](#)

Create user

Required user actions <small>?</small>	Select action
Username *	dave
Email	
Email verified <small>?</small>	<input type="checkbox"/> No
First name	Dave
Last name	Dolittle
Groups <small>?</small>	Join Groups

[Create](#) [Cancel](#)

You will see the details confirmed. Note that keycloak has assigned an internal ID for the user.

dave

	Details	Attributes	Credentials	Role mapping	Groups	Co
--	---------	------------	-------------	--------------	--------	----


ID *	ba0096c8-03b7-4488-a75f-88f972011c0f
Created at *	4/29/2023, 1:05:13 PM
Required user actions ?	Select action
Username *	dave
Email	
Email verified ⓘ	<input type="checkbox"/> No
First name	Dave
Last name	Dolittle

[Save](#) [Revert](#)

To be able to sign-on, the user needs credentials. We'll assign a password.

dave

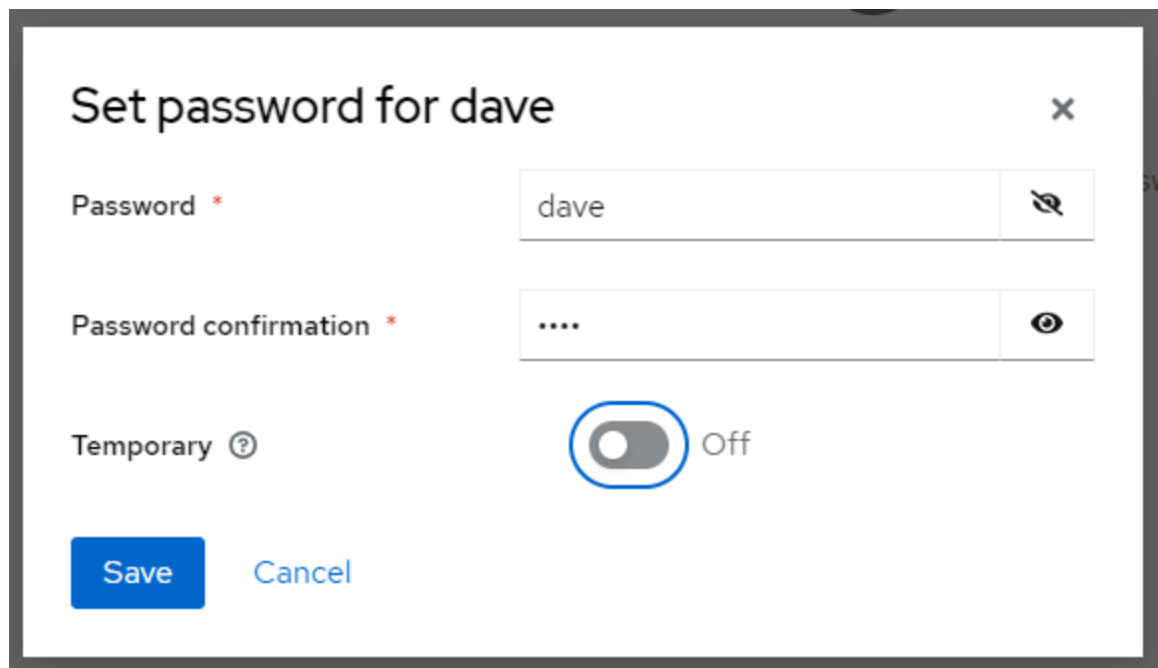
	Details	Attributes	Credentials	Role mapping	Groups	Consents	Identity provider links	Sessions
--	---------	------------	-------------	--------------	--------	----------	-------------------------	----------



No credentials

This user does not have any credentials. You can set password for this user.

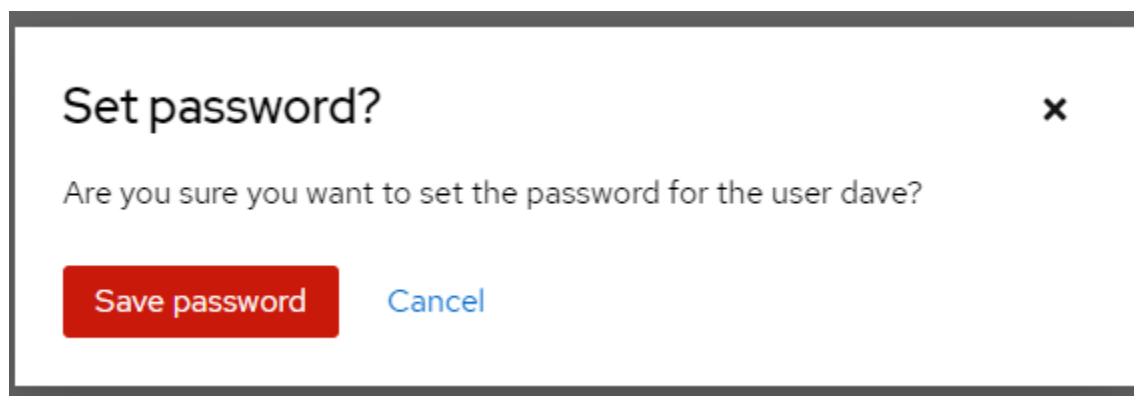
[Set password](#)



The screenshot shows a dialog box titled "Set password for dave" with a close button (X) in the top right corner. It contains two input fields: "Password *" with the text "dave" and a toggle icon, and "Password confirmation *" with four dots and a toggle icon. Below these is a "Temporary" toggle switch, which is currently turned off and labeled "Off". At the bottom are two buttons: "Save" (blue) and "Cancel" (blue text).

To keep things simple for the workshop, set the password to 'dave' as well. (This is *never* recommended in the real world)

Set Temporary to Off to prevent asking to change when signing on.



The screenshot shows a confirmation dialog box titled "Set password?" with a close button (X) in the top right corner. The text inside asks, "Are you sure you want to set the password for the user dave?". At the bottom are two buttons: "Save password" (red) and "Cancel" (blue text).

Confirm and save.

Add user 'Emma Einstein' with password 'emma'

Add user 'Fred Fandango' with password 'fred'.

Create user

Required user actions ?

Select action

Username *

emma

Email

Email verified ?

No

First name

Emma

Last name

Einstein

Groups ?

Join Groups

Create

Cancel

User list

Search user

→

Add user

Delete user

<input type="checkbox"/>	Username	Email	Last name	First name
<input type="checkbox"/>	dave	<div></div>	Dolittle	Dave
<input type="checkbox"/>	emma	<div></div>	Einstein	Emma
<input type="checkbox"/>	fred	<div></div>	Fandango	Fred

Assign User Roles

From the user list, click on 'dave', then select 'Role mapping' then 'Assign role'.

dave

Details

Attributes

Credentials

Role mapping

Groups

Consents

Identi

Q Search by name

→

☒ Hide inherited roles

Assign role

Unassign

<input type="checkbox"/>	Name	Inherited
<input type="checkbox"/>	default-roles-securityworkshop2023	False

Assign role 'Registrant' and 'Shopper'.

Assign roles to dave

Filter by realm roles

▼

Q Search by role name

→

<input type="checkbox"/>	Name	Description
<input type="checkbox"/>	offline_access	`\${role_offline-access}`
<input checked="" type="checkbox"/>	Registrant	Can view and register new customers
<input checked="" type="checkbox"/>	Shopper	Can browse products
<input type="checkbox"/>	SiteAdmin	Access to the whole site
<input type="checkbox"/>	uma_authorization	`\${role_uma_authorization}`

Assign

Cancel

Set Emma to 'SiteAdmin' role,

Set Fred to 'Shopper'

Exercise Two

Configure the 'storeweb' application to be secured by Keycloak.

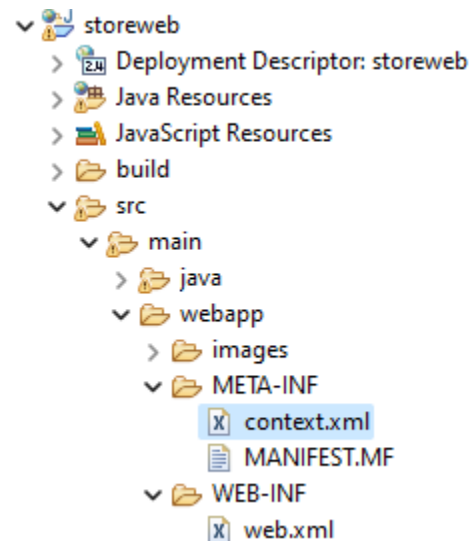
This approach is being deprecated, but alternative is not available yet.

https://www.keycloak.org/docs/latest/securing_apps/#_tomcat_adapter

On VM keycloak tomcat adapter has already been installed, so just configure Tomcat to use it.

Configure storeweb application

Create storeweb\src\main\webapp\META-INF\context.xml

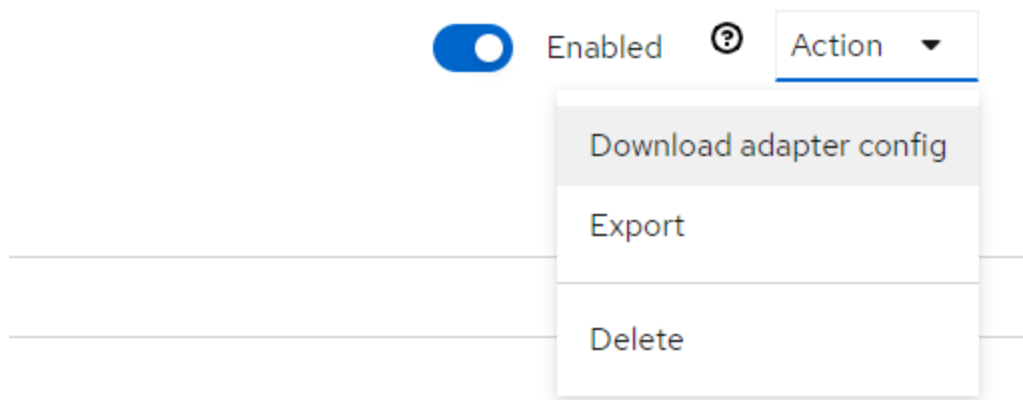


Set contents to the following.


```
<?xml version="1.0" encoding="UTF-8"?>
<Context>
  <Valve className="org.keycloak.adapters.tomcat.KeycloakAuthenticatorValve"/>
</Context>
```

We need information about our keycloak client, so we can authenticate against the users and roles that we have set up. This can be downloaded from Keycloak in a file named keycloak.json


Navigate to your WebStore client in keycloak, you may need to sign in as 'kadmin' again if the session has expired.



Download adaptor configs

Format option 

Keycloak OIDC JSON

Details 

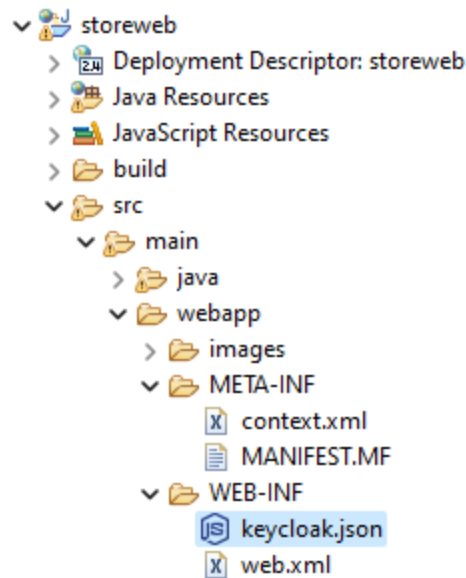
```
{
  "realm": "SecurityWorkshop2023",
  "auth-server-url": "http://localhost:8090/",
  "ssl-required": "external",
  "resource": "WebStore",
  "public-client": true,
  "confidential-port": 0
}
```

Download

Cancel

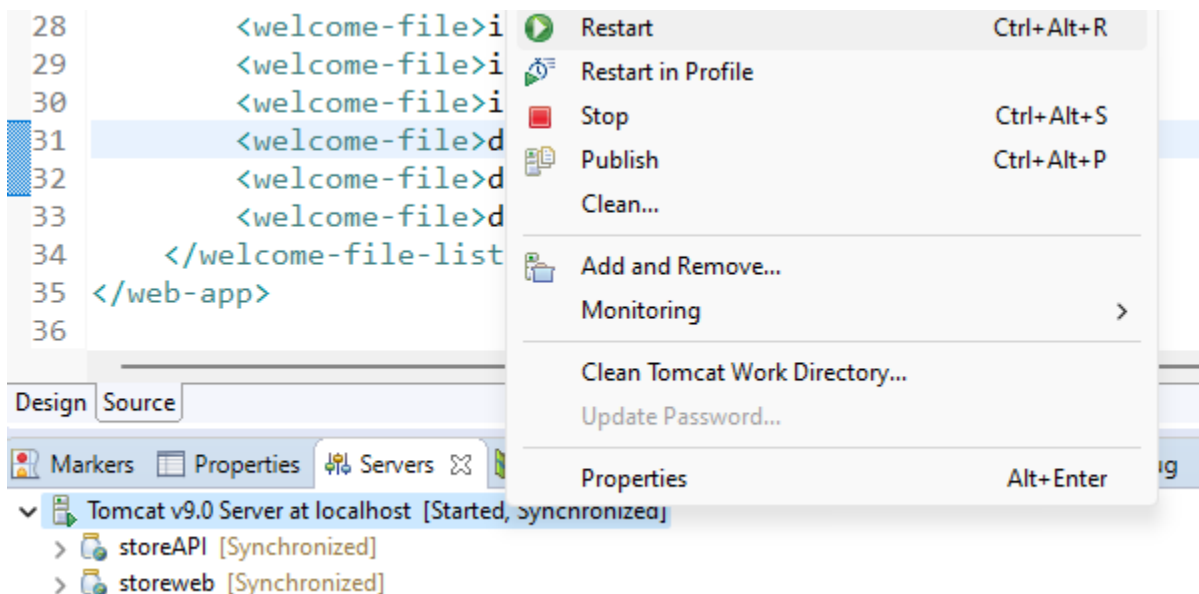
Click 'Download', and save in C:\Conference 2023 Workshops\Web Technology\Workspace\storeweb\src\main\webapp\WEB-INF

In Eclipse, press F5 to refresh, and the file should show up there.

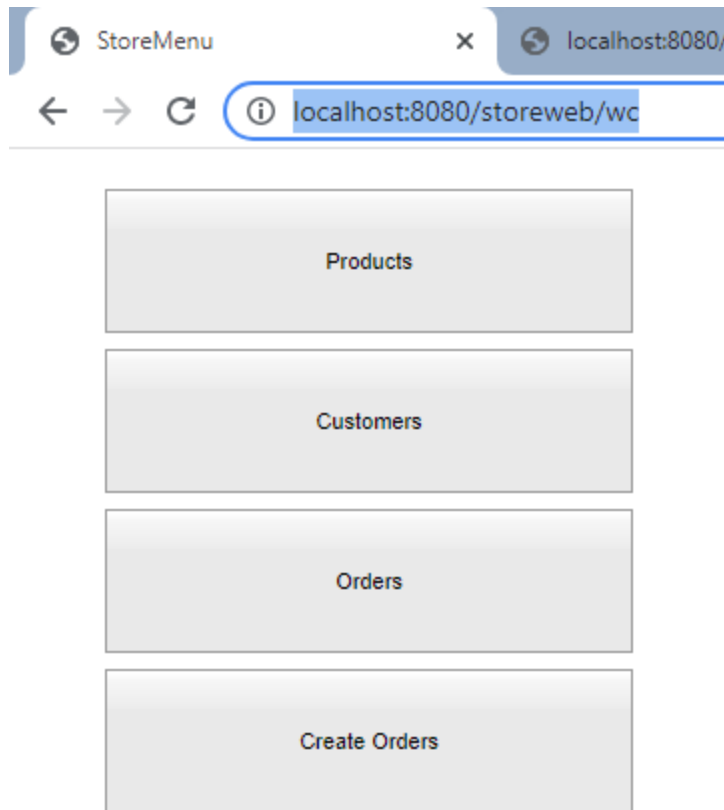


We've now configured Tomcat to use the Keycloak adapter and use our SecurityWorkshop2023 settings.

Let's see if our site has been secured yet. Restart our Tomcat server,



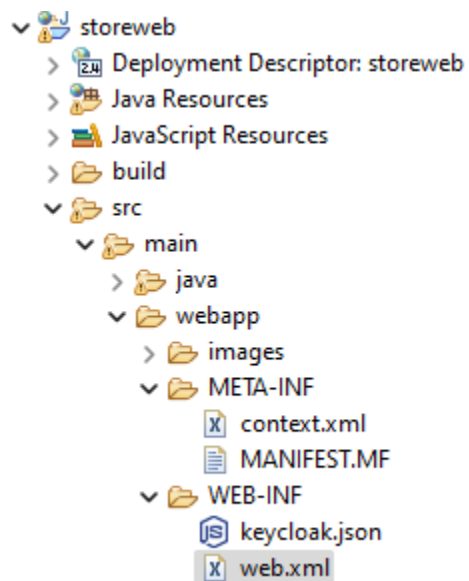
When Tomcat is showing as 'Started', visit the site <http://localhost:8080/storeweb/wc>



The site loads, just as before. The reason for this is that we haven't specified which resources are secured, and who has access to those resources yet.

Set site Security Constraints

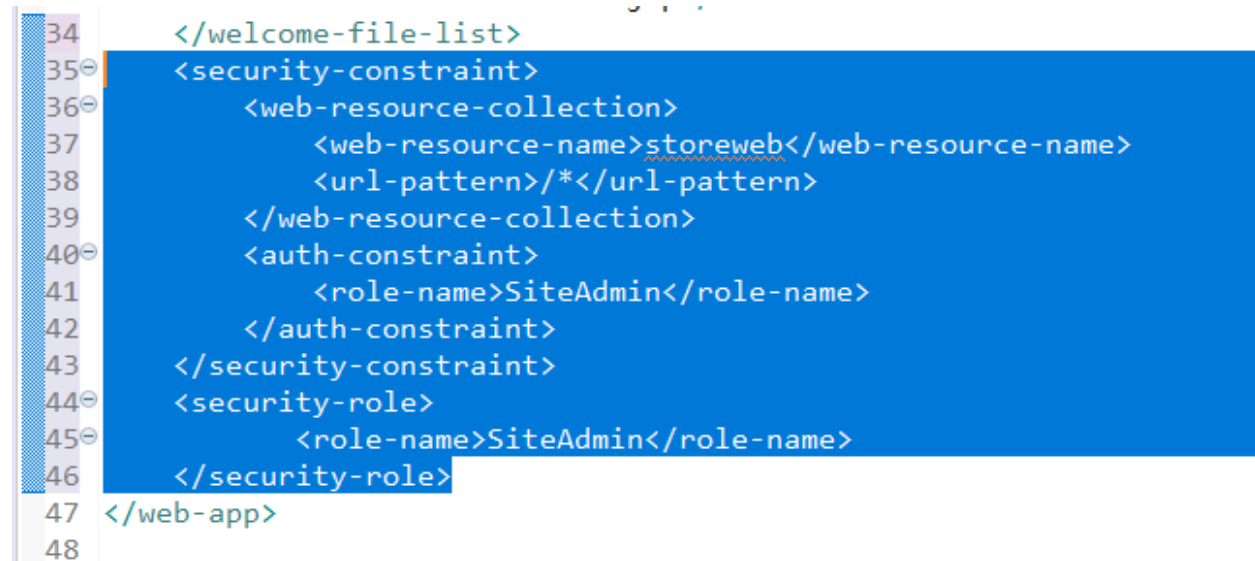
We can specify this in the web application's WEB-INF/web.xml file.



Double-click on the web.xml file to open it in the text editor, and paste the following section between the `</welcome-file-list>` and `</web-app>` tags near the end of the file.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>storeweb</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>SiteAdmin</role-name>
  </auth-constraint>
</security-constraint>
<security-role>
  <role-name>SiteAdmin</role-name>
</security-role>
```

Your file should look like this. Save the file with Ctrl+S.



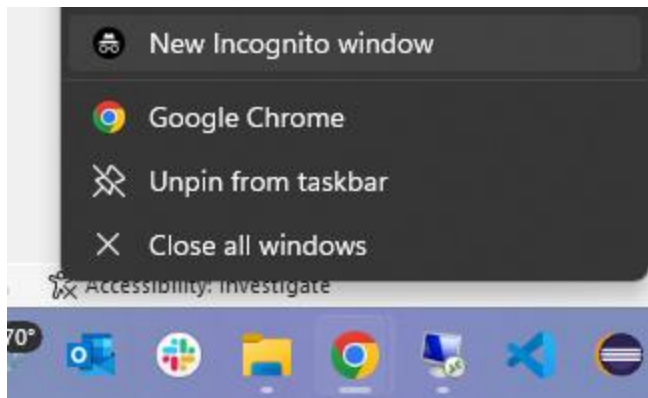
```
34 </welcome-file-list>
35 <security-constraint>
36   <web-resource-collection>
37     <web-resource-name>storeweb</web-resource-name>
38     <url-pattern>/*</url-pattern>
39   </web-resource-collection>
40   <auth-constraint>
41     <role-name>SiteAdmin</role-name>
42   </auth-constraint>
43 </security-constraint>
44 <security-role>
45   <role-name>SiteAdmin</role-name>
46 </security-role>
47 </web-app>
48
```

This states that the whole 'storeweb' web application is secured, and only available to users with the 'SiteAdmin' role.

Restart Tomcat in Eclipse.

We are going to test the security setting by signing in as a number of different users. Normally, the login information is remembered in a browser cookie, making it more difficult to sign in as someone else. To get around this, we can access the site in 'Incognito Mode'.

Right-click on the Chrome icon in the task bar and select 'New Incognito window', then visit the site <http://localhost:8080/storeweb/wc> again.



This time we will see a sign-on window. Try to sign in as 'fred', with password 'fred'.

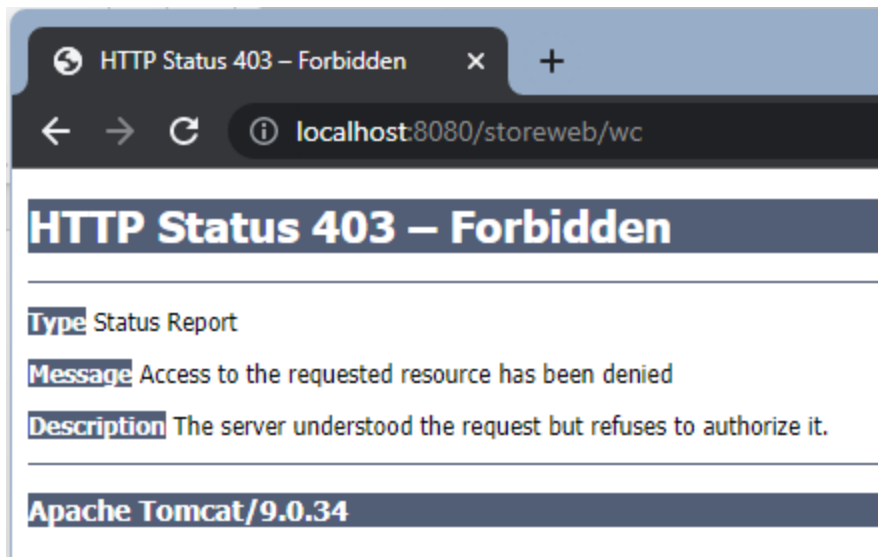
SECURITYWORKSHOP2023

Sign in to your account

Username or email

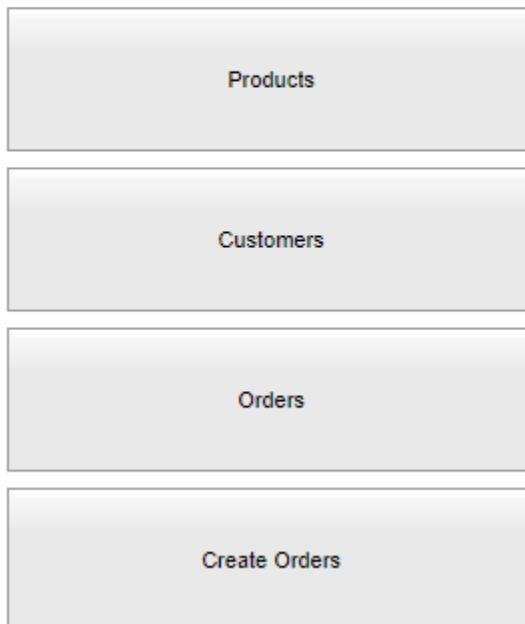
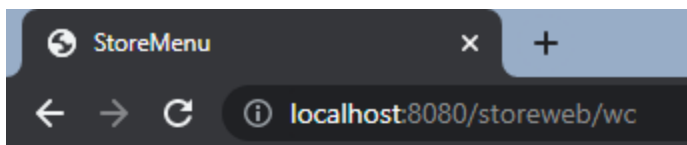
Password

Sign In



Fred hasn't been assigned the 'SiteAdmin' role, so he doesn't have access to the web application.

Close the browser and restart it in Incognito mode again. Access the site <http://localhost:8080/storeweb/wc>, this time sign on as 'emma' with password 'emma'. This time she has access to the whole site as the administrator.



Set API Security constraints

In the workspace, we also have an API generated by HSync. This allows a user to access products or customers with the following URLs.

<http://localhost:8080/storeAPI/api/v1/products>

<http://localhost:8080/storeAPI/api/v1/customers>

We can configure the API in a similar way to only allow 'Shopper' roles to view products, and 'Registrant' roles to view Customers. We also want users with the SiteAdmin role to view both.

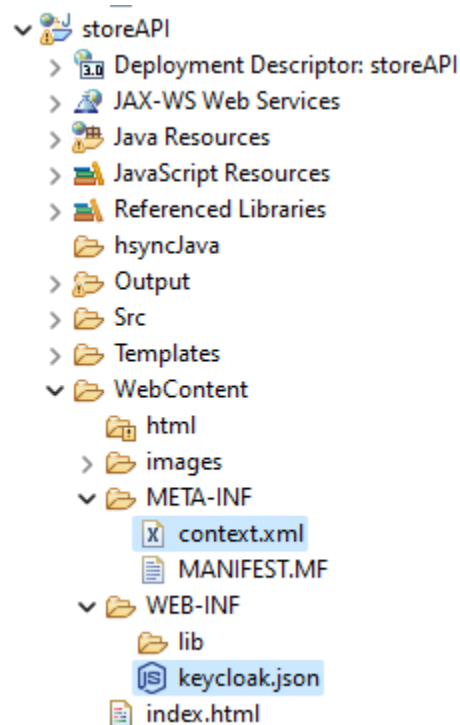
We can copy a couple of the config files we set up in this exercise to the storeAPI project.

Copy file: storeweb\src\main\webapp\META-INF\context.xml

To folder: storeAPI\WebContent\META-INF

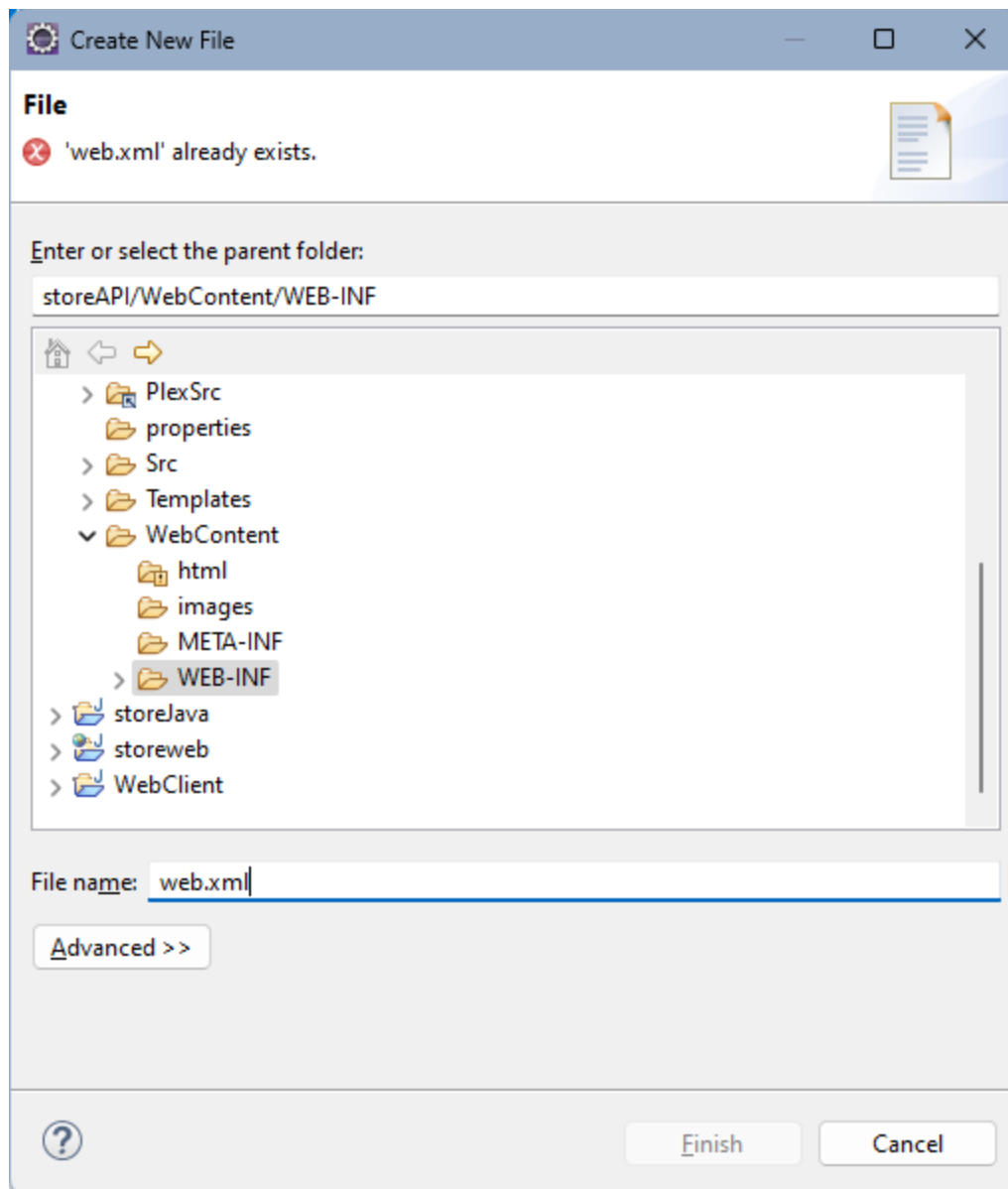
Copy file: storeweb\src\main\webapp\WEB-INF\keycloak.json

To folder: storeAPI\WebContent\WEB-INF



To specify the security constraints, we can add a web.xml file.

Right-click on storeAPI\WebContent\WEB-INF, then select New -> File. Enter the name "web.xml"



Double-click on the new file to edit it, then paste the following XML. This constrains the Products API to users with role 'Shopper' or 'SiteAdmin', and Customers API to users with role 'Registrant' or 'SiteAdmin'.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4"
xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
  <display-name>storeAPI</display-name>
  <security-constraint>
    <web-resource-collection>
```

```

        <web-resource-name>storeAPI</web-resource-name>
            <url-pattern>/api/v1/products</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>Shopper</role-name>
            <role-name>SiteAdmin</role-name>
        </auth-constraint>
    </security-constraint>
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>storeAPI</web-resource-name>
            <url-pattern>/api/v1/customers</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>Registrant</role-name>
            <role-name>SiteAdmin</role-name>
        </auth-constraint>
    </security-constraint>
    <security-role>
        <role-name>SiteAdmin</role-name>
        <role-name>Registrant</role-name>
        <role-name>Shopper</role-name>
    </security-role>
</web-app>

```

Try accessing the APIs as different users.

<http://localhost:8080/storeAPI/api/v1/products>

<http://localhost:8080/storeAPI/api/v1/customers>

The expected results are:

- “dave” can access both as he is both a “Shopper” and a “Registrant”.
- “emma” can access both as she is a “SiteAdmin”
- “fred” can only access customers, as he is just a “Registrant”.

Exercise Three

Control access from Plex application

Menu, enable buttons for user

Get attributes.

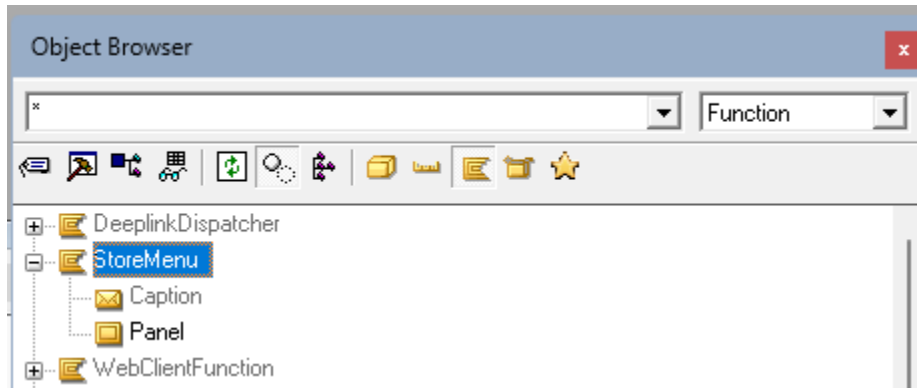
Starting Point

Restore "Security-PlexStart.7z" – REDO THIS TO INCLUDE KC JARS

Set up Plex function

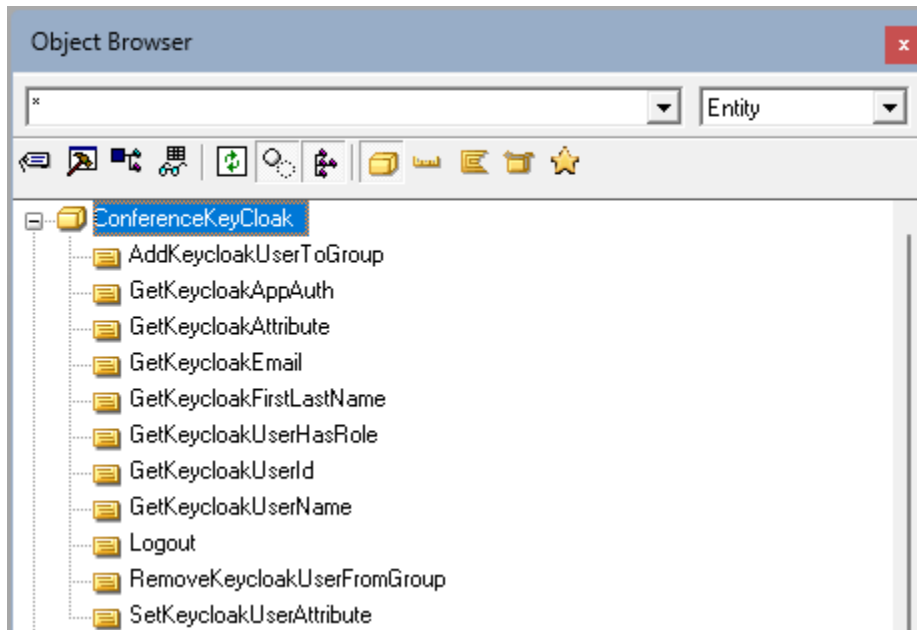
Open store.mdl

Open the StoreMenu function in the Action Diagram Editor



Add a new Subroutine “Sub Get User Information”

In the Object Browser, look for an Entity named ‘ConferenceKeyCloak’. This contains several source code objects we can use to work with our KeyCloak setup.



First, we’ll check our user roles and enable menu options based on these.

Source code object ConferenceKeyCloak.GetKeycloakUserHasRole allows us to check if the User is assigned a specific role.

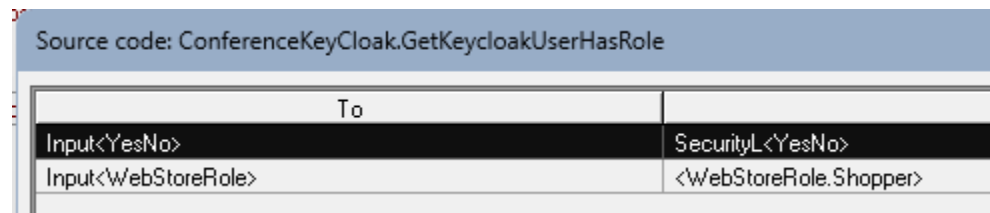
Add the statement **API Call** Source code: `ConferenceKeyCloak.GetKeycloakUserHasRole`

Map the following parameters:

Input<YesNo>

SecurityL<YesNo>

Input< WebStoreRole > <WebStoreRole.Shopper>



Source code: ConferenceKeyCloak.GetKeycloakUserHasRole	
To	
Input<YesNo>	SecurityL<YesNo>
Input<WebStoreRole>	<WebStoreRole.Shopper>

Conditionally hide the Products button.

If SecurityL<YesNo> == <YesNo.No>

Set State Protected, Event: Products

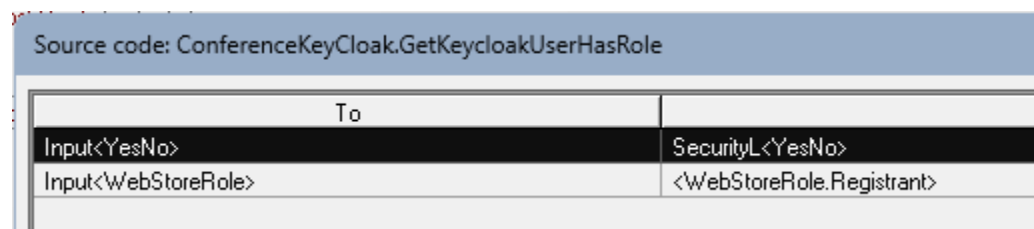
Now do the same for the Registrant Role to enable/disable the Customers button.

API Call Source code: ConferenceKeyCloak.GetKeycloakUserHasRole

Map the following parameters:

Input<YesNo> SecurityL<YesNo>

Input< WebStoreRole > <WebStoreRole.Registrant>



Source code: ConferenceKeyCloak.GetKeycloakUserHasRole	
To	
Input<YesNo>	SecurityL<YesNo>
Input<WebStoreRole>	<WebStoreRole.Registrant>

If SecurityL<YesNo> == <YesNo.No>

Set State Protected, Event: Customers

Now we can get the user attributes from KeyCloak for the Welcome message.

We'll use the Source code object *ConferenceKeyCloak.GetKeycloakAttribute* for this.

Add the statement:

API Call Source code: ConferenceKeyCloak.GetKeycloakAttribute

And map the following parameters:

Input<Attribute> <WebStoreAttribute.WelcomeMessage>

Input<Attribute Value> SecurityL<Attribute Value>

Source code: ConferenceKeyCloak.GetKeycloakAttribute

To	From
Input<Attribute>	<WebStoreAttribute.WelcomeMessage>
Input<Attribute Value>	SecurityL<Attribute Value>

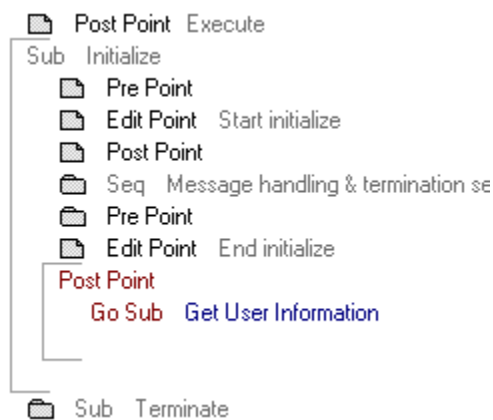
If SecurityL<Attribute Value> != <LongDescription.*Blank>

Cast Environment<*Message text>, SecurityL<Attribute Value>

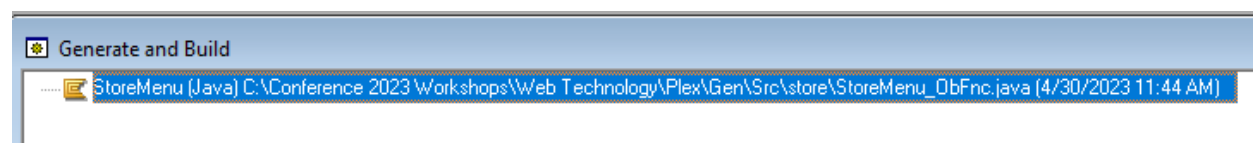
Go Sub Send message

Finally, add the following line to the End Initialize Post Point

Go Sub Get User Information



Save your model, close the action diagram and generate the StoreMenu function:



Open the C:\Conference 2023 Workshops\Web Technology\Workspace workspace in Eclipse.

Press F5 to refresh the workspace with your latest changes. The WebClient generator will build your function.

>>> Starting WebClient Build for storeJava

> Using CM WebClient v1.8.8-pre13284
> Licensed to CMFirst Technologies
> Valid until Mon Jan 01 00:00:00 CST 2024

Generating Web templates.

>>> WebClient Build complete.

1 panels processed. 0 panels not processed.

In the previous exercise we restricted access to our web application by assigning access to the 'SiteAdmin' role. Let's change this to allow access to all users.

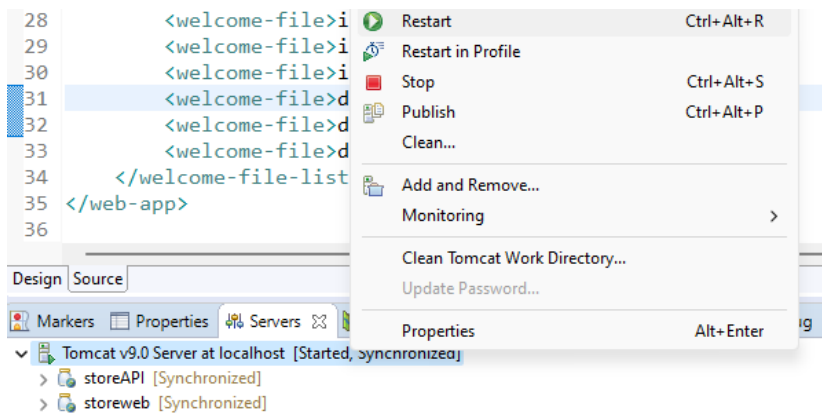
```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>storeweb</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>SiteAdmin</role-name>
  </auth-constraint>
</security-constraint>
<security-role>
  <role-name>SiteAdmin</role-name>
</security-role>
```

Change the 'SiteAdmin' values to '*' for all access.

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>storeweb</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>*</role-name>
  </auth-constraint>
</security-constraint>
<security-role>
  <role-name>*</role-name>
</security-role>
```

Save the web.xml file.

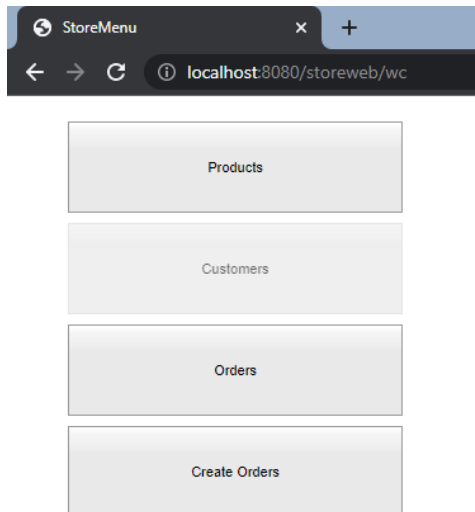
Restart the Tomcat Server on the Servers tab.



Make sure that Keycloak is running and access the web application again.

<http://localhost:8080/storeweb/wc>

Sign-on as 'fred' with password 'fred'.



The 'Customers' menu option is disabled, as Fred does not have the 'Registrant' role assigned.

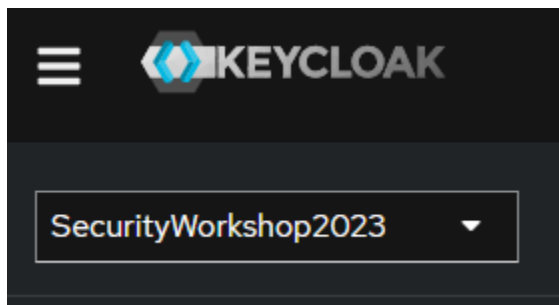
We don't see a welcome message as we haven't defined that yet. We'll do that in the next step.

Create OpenID Connect Mappers

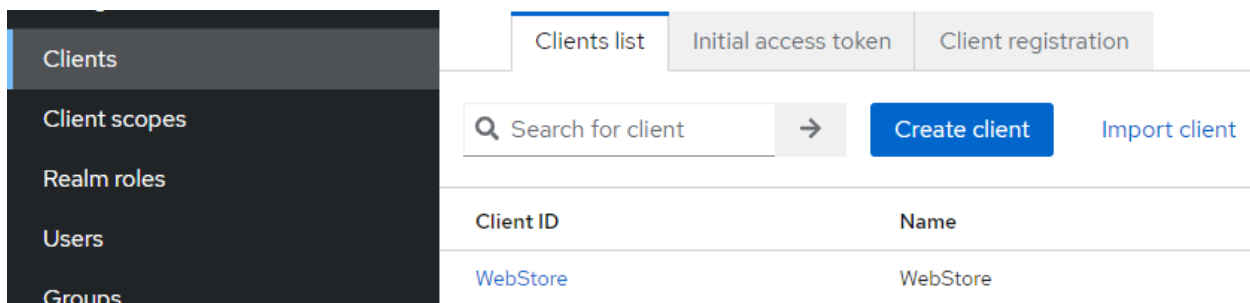
When we set up Keycloak, we defined it (by default) to use the OpenID Connect (OIDC) protocol. This is an extension to the OAuth 2.0 protocol that allows it to attach user identity data to an access token, e.g. an email address or nationality. This information is known as a “claim”.

To allow these attributes to be attached to the token, we first need to set up mappers for these attributes. There are many types of mappers, but we’ll just deal with a User Attribute mapper for this exercise.

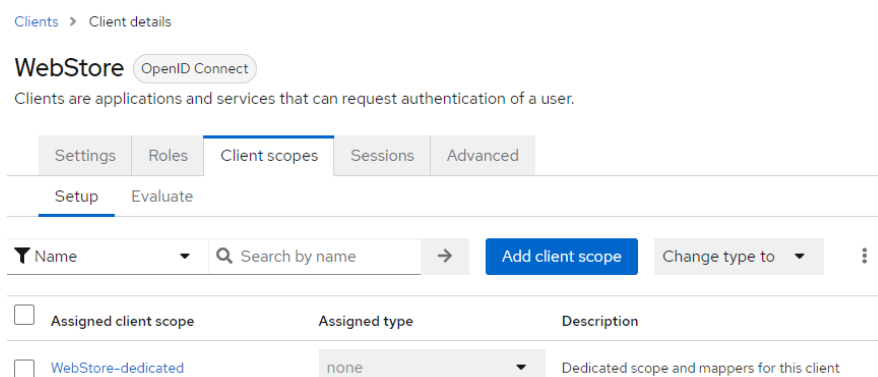
Log on to the Keycloak administration console at <http://localhost:8090/> as ‘kadmin’ with password ‘kadmin’ and select the “SecurityWorkshop2023” Realm.



Select Client, then click on our WebStore client.



Click on the Client Scopes tab. In Keycloak, a client scope is a set of predefined permissions or attributes that can be associated with a client application. Client scopes provide a way to manage and share sets of permissions or attributes across multiple client applications in a standardized way.



Click on the “WebStore-dedicated” link.

WebStore

This is a client scope which includes the dedicated mappers and scope

Mappers

Scope

Q

Search for mapper

→

Add mapper ▾

Click on “Add mapper”. The first time in you’ll see the “No mappers” message.



No mappers

If you want to add mappers, please click the button below to add some predefined mappers or to configure a new mapper.

Add predefined mapper

Configure a new mapper

Click on the “Configure a new mapper” button. There are several mapper types listed. Select “User Attribute”

	'address' claim.
User Attribute	Map a custom user attribute to a token claim.
User Client Role	Map a user client role to a token claim.

[Clients](#) > [Client details](#) > [Dedicated scopes](#) > [Mapper details](#)

Add mapper

If you want more fine-grain control, you can create protocol mapper on this client

Mapper type	User Attribute
Name * ?	Welcome Message
User Attribute ?	WelcomeMessage
Token Claim Name ?	<u>WelcomeMessage</u>
Claim JSON Type ?	String
Add to ID token ?	<input type="checkbox"/> Off
Add to access token ?	<input checked="" type="checkbox"/> On
Add to userinfo ?	<input type="checkbox"/> Off
Multivalued ?	<input type="checkbox"/> Off
Aggregate attribute values ?	<input type="checkbox"/> Off

[Save](#) [Cancel](#)

On the “Add Mapper” panel, set the values as above and click “Save”.

Now that we have the mapper, we can add an individual welcome message to each user.

Select ‘Users’ from the menu, and click on ‘emma’.

Realm roles	
Users	<input type="checkbox"/> Username
Groups	<input type="checkbox"/> dave
Sessions	<input type="checkbox"/> emma
Events	<input type="checkbox"/> fred

Users > User details

emma

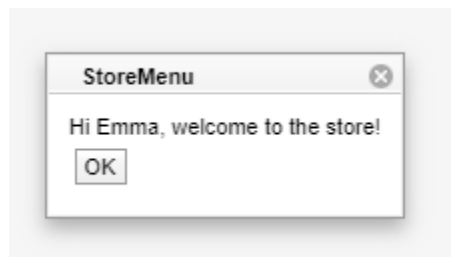
Details	Attributes	Credentials	Role mapping	Groups	Consents	Identity provider links	Sessions						
<table><thead><tr><th>Key</th><th>Value</th></tr></thead><tbody><tr><td>WelcomeMessage</td><td>Hi Emma, welcome to the store!</td></tr><tr><td>Type a key</td><td>Type a value</td></tr></tbody></table>								Key	Value	WelcomeMessage	Hi Emma, welcome to the store!	Type a key	Type a value
Key	Value												
WelcomeMessage	Hi Emma, welcome to the store!												
Type a key	Type a value												

Click on the Attributes tab and enter the key “WelcomeMessage”, then enter a message in the Value column. Click Save.

In Eclipse, start the Tomcat server if it’s not already running, and visit the web application in a new browser in Incognito mode at <http://localhost:8080/storeweb/wc>

Sign in as “Emma” with password “Emma”.

The message should now show up.



Add WelcomeMessage attributes to users Fred and Dave. Test out those users.