**Demo 1: Client Configuration**

**Steps to Configure Clients:**

1. **Navigate to the Clients Section:**
   * Open Keycloak Admin Console.
   * Select the realm where you want to configure the client.
   * Click on the "Clients" menu item on the left.
2. **Create a New Client:**
   * Click on the "Create" button at the top right.
   * Enter a unique **Client ID** (e.g., my-client).
   * Choose **Client Protocol** (e.g., openid-connect or saml).
3. **Configure Client Settings:**
   * Set **Root URL** if needed.
   * Enter **Redirect URIs** (e.g., http://localhost:8080/\* for testing).
   * Configure other settings like **Web Origins** if required.
4. **Save the Client Configuration:**
   * Click on the "Save" button.
5. **Review and Adjust Additional Settings:**
   * After saving, navigate through the tabs (Settings, Credentials, Roles, Mappers, etc.) to further customize the client as needed.

**Step-by-Step Process:**

**Part 1: Navigate to the Clients Section**

1. **Open Keycloak Admin Console:**
   * Open a web browser and navigate to your Keycloak server (e.g., http://localhost:8080/auth).
   * Log in with your administrator credentials.
2. **Select the Realm:**
   * In the top-left corner, click the dropdown to select the realm you want to work with.
   * If you don't have a realm, create one by clicking "Add realm", entering a name (e.g., demo-realm), and clicking "Create".
3. **Access Clients:**
   * In the left-hand menu, click on "Clients".
   * You will be taken to the Clients section where you can see a list of existing clients.

**Part 2: Create a New Client**

1. **Add a New Client:**
   * In the Clients section, click on the "Create" button at the top-right corner.
   * You will be directed to a form to create a new client.
2. **Enter Client Details:**
   * **Client ID:** Enter a unique identifier for the client (e.g., my-app-client).
   * **Client Protocol:** Choose the protocol you want to use. For most web applications, select openid-connect. If you are integrating with a SAML-based application, select saml.
   * Click "Save".

**Part 3: Configure Client Settings**

1. **Settings Tab:**
   * After saving, you will be taken to the client’s settings page.
2. **Basic Settings:**
   * **Enabled:** Ensure this is set to ON.
   * **Client Protocol:** Should be set to the protocol you selected during creation.
   * **Access Type:** Choose the appropriate access type:
     + **Confidential:** Requires client authentication (typically for server-side applications).
     + **Public:** No client authentication required (typically for browser-based applications).
     + **Bearer-only:** Used for securing back-end services where authentication is done via bearer tokens.
   * **Standard Flow Enabled:** Enable this for authorization code flow.
   * **Implicit Flow Enabled:** Enable if you want to use the implicit flow (less secure, typically for single-page applications).
   * **Direct Access Grants Enabled:** Enable if you want to allow direct access grants (e.g., resource owner password credentials grant).
   * **Service Accounts Enabled:** Enable if you need service account access for server-to-server communication.
3. **Client URIs:**
   * **Root URL:** Set the root URL of your application (e.g., http://localhost:8080).
   * **Valid Redirect URIs:** Add URIs where Keycloak can redirect the user after authentication (e.g., http://localhost:8080/\*).
   * **Base URL:** Base URL for the client if applicable (often same as Root URL).
   * **Admin URL:** URL to be used by Keycloak for back-channel logout.
   * **Web Origins:** Specify allowed CORS origins (e.g., http://localhost:8080 or \* for all origins).
4. **Advanced Settings (Optional):**
   * **Login Theme:** Choose a custom theme if you have one.
   * **Consent Required:** Enable if you want users to consent to client access.
   * **Client Authentication:** Enable if the client must authenticate to Keycloak (relevant for confidential clients).
5. **Save Settings:**
   * Click on the "Save" button to apply the settings.

**Part 4: Review and Adjust Additional Settings**

1. **Credentials Tab:**
   * This tab contains the credentials required for client authentication.
   * **Client Secret:** If your client is confidential, a secret will be generated here. Note this value as you’ll need it to configure your application.
   * **Regenerate Secret:** You can regenerate the client secret if necessary.
2. **Roles Tab:**
   * You can define roles specific to this client.
   * Click "Add Role" to create new roles (e.g., admin, user).
   * These roles can be assigned to users for this client.
3. **Mappers Tab:**
   * Mappers are used to map claims and attributes from the token to the user.
   * Click "Create" to add a new mapper.
   * **Name:** Give a name to the mapper (e.g., username-mapper).
   * **Mapper Type:** Choose the appropriate type (e.g., User Property, User Attribute).
   * **Property:** Specify the user property to map (e.g., username).
   * Save the mapper.
4. **Scope Tab:**
   * Define the scopes for the client if necessary.
   * Default scopes (e.g., openid, profile, email) are usually sufficient.
5. **Sessions Tab:**
   * Manage active sessions for the client.
   * View and revoke sessions if needed.
6. **Fine-Grain OpenID Connect Settings:**
   * **Access Token Lifespan:** Set the lifespan of the access token.
   * **Refresh Token Lifespan:** Set the lifespan of the refresh token.
   * **ID Token Signature Algorithm:** Choose the algorithm for ID token signing (e.g., RS256).
7. **Save Any Additional Changes:**
   * After making any changes in these tabs, always click "Save" to apply the changes.

**Part 5: Integrate Client with Application**

1. **Use Keycloak Configuration in Your Application:**
   * For JavaScript applications, use the Keycloak JavaScript adapter as described in Demo 8.
   * For backend applications like Spring Boot, use the relevant Keycloak adapters and configurations.
2. **Test the Integration:**
   * Run your application and attempt to authenticate using Keycloak.
   * Verify that the application redirects to Keycloak for authentication and then back to the application upon successful login.

**Demo 2: Managing Users and User Groups**

**Creating and Managing Users:**

1. **Navigate to the Users Section:**
   * Open Keycloak Admin Console.
   * Select the realm where you want to manage users.
   * Click on the "Users" menu item on the left.
2. **Create a New User:**
   * Click on the "Add User" button at the top right.
   * Enter required details such as **Username**, **Email**, **First Name**, and **Last Name**.
   * Click on the "Save" button.
3. **Set User Credentials:**
   * After saving, go to the "Credentials" tab.
   * Set a password for the user and ensure it is marked as "Temporary" or "Permanent" as required.
   * Click on the "Set Password" button.
4. **Assign Roles to the User:**
   * Go to the "Role Mappings" tab.
   * Assign roles to the user by selecting roles from the "Available Roles" list and clicking "Add selected".
5. **Assign User to Groups:**
   * Go to the "Groups" tab.
   * Add the user to relevant groups by selecting from the "Available Groups" list and clicking "Join".

**Step-by-Step Process:**

**Part 1: Managing Users**

1. **Navigate to the Users Section:**
   * Open the Keycloak Admin Console.
   * Select the realm where you want to manage users.
   * In the left-hand menu, click on "Users".
   * You will be taken to the Users section where you can see a list of existing users.
2. **Create a New User:**
   * In the Users section, click on the "Add user" button at the top-right corner.
   * Fill in the required fields:
     + **Username:** Enter a unique username for the user.
     + **Email:** (Optional) Enter the user’s email address.
     + **First Name:** Enter the user’s first name.
     + **Last Name:** Enter the user’s last name.
     + **User Enabled:** Ensure this is set to ON to enable the user.
   * Click "Save".
3. **Set User Credentials:**
   * After saving the user, you will be taken to the user’s settings page.
   * Go to the "Credentials" tab.
   * Click on "Set Password".
   * Enter and confirm the new password.
   * Optionally, you can toggle the "Temporary" switch to OFF if you do not want the user to change the password on first login.
   * Click "Save".
4. **Configure User Attributes:**
   * Go to the "Attributes" tab.
   * Click "Add" to create custom attributes for the user.
   * Enter the attribute Key and Value (e.g., department and HR).
   * Click "Save".
5. **Assign Roles to User:**
   * Go to the "Role Mappings" tab.
   * You can assign realm-level roles or client-level roles to the user.
   * **Realm Roles:**
     + Select roles from the "Available Roles" column.
     + Click "Add selected" to assign the roles to the user.
   * **Client Roles:**
     + Select the client from the "Client Roles" dropdown.
     + Select roles from the "Available Roles" column.
     + Click "Add selected" to assign the roles to the user.
6. **Review and Manage User Sessions:**
   * Go to the "Sessions" tab.
   * View active sessions and log the user out if necessary by clicking "Logout All".
7. **User Consents:**
   * Go to the "Consents" tab.
   * View and revoke user consents for clients if necessary.

**Part 2: Managing User Groups**

1. **Navigate to the Groups Section:**
   * In the left-hand menu, click on "Groups".
   * You will be taken to the Groups section where you can see a list of existing groups.
2. **Create a New Group:**
   * In the Groups section, click on the "New" button at the top-right corner.
   * Enter a Group Name (e.g., HR Team).
   * Click "Save".
3. **Manage Group Membership:**
   * After saving, you will be taken to the group’s settings page.
   * Go to the "Members" tab.
   * Click "Join" to add users to the group.
   * Select users from the list and click "Join".
4. **Assign Roles to Group:**
   * Go to the "Role Mappings" tab.
   * You can assign realm-level roles or client-level roles to the group.
   * **Realm Roles:**
     + Select roles from the "Available Roles" column.
     + Click "Add selected" to assign the roles to the group.
   * **Client Roles:**
     + Select the client from the "Client Roles" dropdown.
     + Select roles from the "Available Roles" column.
     + Click "Add selected" to assign the roles to the group.
5. **Configure Group Attributes:**
   * Go to the "Attributes" tab.
   * Click "Add" to create custom attributes for the group.
   * Enter the attribute Key and Value (e.g., location and Building A).
   * Click "Save".
6. **Nest Groups:**
   * You can create a hierarchy by nesting groups.
   * In the Groups section, select a parent group.
   * Click "New" to create a sub-group.
   * Enter a name for the sub-group and click "Save".
7. **Manage Group Permissions:**
   * Go to the "Permissions" tab if you want to manage fine-grained permissions for the group.
   * Enable "Admin Permissions" if needed.

**Part 3: User and Group Integration**

1. **Assign Groups to Users:**
   * In the Users section, select a user.
   * Go to the "Groups" tab.
   * Click "Join" to add the user to one or more groups.
   * Select groups from the list and click "Join".
2. **Review Effective Roles:**
   * Go to the "Role Mappings" tab for a user.
   * Click "Effective Roles" to view all roles assigned to the user, including those inherited from groups.

**Demo 3: Keycloak SSO**

**Setting Up Keycloak SSO:**

1. **Configure Identity Providers:**
   * Navigate to the "Identity Providers" section in the Admin Console.
   * Select the identity provider type (e.g., **OpenID Connect v1.0**, **SAML v2.0**).
   * Fill in the required settings such as **Alias**, **Authorization URL**, **Token URL**, **Client ID**, and **Client Secret**.
2. **Set Up Client Applications:**
   * Go to the "Clients" section.
   * Select the client application you want to set up for SSO.
   * Ensure **SSO settings** are correctly configured, especially **Redirect URIs** and **Web Origins**.
3. **Test the SSO Flow:**
   * Access your application.
   * Trigger the login process to see the SSO in action. Users should be redirected to Keycloak for authentication and back to the application upon successful login.

**Step-by-Step Process:**

**Part 1: Understanding Single Sign-On (SSO)**

**What is SSO?**

* **Definition:** Single Sign-On (SSO) is an authentication process that allows a user to access multiple applications with one set of login credentials.
* **Benefits:**
  + Simplifies user management.
  + Enhances user experience by reducing the need for multiple logins.
  + Increases security by centralizing authentication.

**Part 2: Setting Up Keycloak for SSO**

1. **Create a Realm:**
   * Open the Keycloak Admin Console.
   * In the top-left corner, click the dropdown to select or add a new realm.
   * Click "Add realm", enter a name (e.g., sso-realm), and click "Create".
2. **Create Clients for Applications:**
   * In the Keycloak Admin Console, navigate to the sso-realm.
   * Click on "Clients" in the left-hand menu.
   * Click "Create".
   * **For Application 1:**
     + **Client ID:** Enter a unique identifier (e.g., app1-client).
     + **Client Protocol:** Select openid-connect.
     + **Root URL:** Enter the URL of your first application (e.g., http://localhost:8081).
     + Click "Save".
   * **For Application 2:**
     + Repeat the steps above, using a different Client ID (e.g., app2-client) and the URL for the second application (e.g., http://localhost:8082).
3. **Configure Clients:**
   * **Common Settings for Both Clients:**
     + **Access Type:** Set to confidential for backend applications or public for frontend applications.
     + **Valid Redirect URIs:** Enter the redirect URIs for your applications (e.g., http://localhost:8081/\* for Application 1 and http://localhost:8082/\* for Application 2).
     + **Web Origins:** Set to \* or specify your application's origin (e.g., http://localhost:8081 and http://localhost:8082).
4. **Set Up Client Credentials:**
   * For confidential clients, a client secret will be generated.
   * Go to the "Credentials" tab of each client and note down the client secret.

**Part 3: Configuring Applications for SSO**

1. **Integrate Application 1:**
   * **JavaScript Application:**
     + Include Keycloak JavaScript adapter:

<script src="https://cdnjs.cloudflare.com/ajax/libs/keycloak/11.0.3/keycloak.min.js"></script>

* + - Initialize Keycloak in your JavaScript code:

var keycloak = new Keycloak({

url: 'http://localhost:8080/auth',

realm: 'sso-realm',

clientId: 'app1-client'

});

keycloak.init({ onLoad: 'login-required' }).then(function(authenticated) {

console.log(authenticated ? 'Authenticated' : 'Not authenticated');

}).catch(function() {

console.log('Failed to initialize');

});

* + **Spring Boot Application:**
    - Add Keycloak dependencies to pom.xml:

<dependency>

<groupId>org.keycloak</groupId>

<artifactId>keycloak-spring-boot-starter</artifactId>

<version>12.0.4</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

* + - Configure application.properties:

keycloak.realm=sso-realm

keycloak.auth-server-url=http://localhost:8080/auth

keycloak.resource=app1-client

keycloak.credentials.secret=<client-secret>

keycloak.use-resource-role-mappings=true

security.basic.enabled=false

* + - Create a security configuration class:

@Configuration

@EnableWebSecurity

public class SecurityConfig extends KeycloakWebSecurityConfigurerAdapter {

@Bean

@Override

protected SessionAuthenticationStrategy sessionAuthenticationStrategy() {

return new RegisterSessionAuthenticationStrategy(new SessionRegistryImpl());

}

@Override

protected void configure(HttpSecurity http) throws Exception {

super.configure(http);

http.authorizeRequests()

.anyRequest().authenticated();

}

}

1. **Integrate Application 2:**
   * Repeat the integration steps used for Application 1, adjusting the configuration for app2-client.

**Part 4: Testing SSO Integration**

1. **Access Application 1:**
   * Open a web browser and navigate to Application 1 (e.g., http://localhost:8081).
   * You should be redirected to the Keycloak login page.
   * Log in with your Keycloak credentials.
   * After a successful login, you should be redirected back to Application 1.
2. **Access Application 2:**
   * Open a new tab in the same browser and navigate to Application 2 (e.g., http://localhost:8082).
   * You should not be prompted to log in again, as the SSO session from Application 1 should allow seamless access to Application 2.
3. **Verify SSO Functionality:**
   * Ensure that both applications are accessible with the same login session.
   * Log out from one application and verify that the session is terminated for both applications.

**Part 5: Advanced SSO Configuration (Optional)**

1. **SSO Session Settings:**
   * Go to the realm settings in the Keycloak Admin Console.
   * Navigate to the "Tokens" tab.
   * Configure session timeouts and token lifespans according to your requirements.
2. **Custom Login Themes:**
   * Customize the login page and themes by navigating to the "Themes" tab in the realm settings.
   * Select or upload custom themes.
3. **Role-Based Access Control (RBAC):**
   * Define roles at the realm or client level.
   * Assign roles to users and configure role mappings in the applications.

By following these detailed steps, even beginners should be able to set up and test Single Sign-On (SSO) with Keycloak, ensuring a smooth and secure user experience across multiple applications.

4o

**Demo 4: Authentication with OpenID Connect**

**Configuring OpenID Connect:**

1. **Set Up Clients for OpenID Connect:**
   * Navigate to the "Clients" section.
   * Create or select an existing client.
   * Ensure **Client Protocol** is set to openid-connect.
2. **Configure Client Settings:**
   * Fill in **Redirect URIs** and **Web Origins**.
   * Go to the "Credentials" tab to get the **Client Secret**.
3. **Use OpenID Connect Endpoints:**
   * Use the following endpoints in your application for authentication flows:
     + **Authorization Endpoint:** http://<keycloak-server>/auth/realms/<realm>/protocol/openid-connect/auth
     + **Token Endpoint:** http://<keycloak-server>/auth/realms/<realm>/protocol/openid-connect/token
     + **UserInfo Endpoint:** http://<keycloak-server>/auth/realms/<realm>/protocol/openid-connect/userinfo
4. **Integrate in Application:**
   * Use an OpenID Connect library (e.g., oidc-client-js for JavaScript) to handle the authentication flow.
   * Configure the library with your Keycloak settings.

**Step-by-Step Process:**

**Part 1: Understanding OpenID Connect (OIDC)**

**What is OpenID Connect (OIDC)?**

* **Definition:** OpenID Connect is an authentication layer built on top of OAuth 2.0, providing a standard way for clients to obtain identity information from an authorization server.
* **Features:**
  + **Authentication:** Provides user authentication via an identity token.
  + **Authorization:** Allows clients to obtain authorization via access tokens.
  + **User Information:** Provides user profile information via UserInfo endpoint.

**Part 2: Setting Up Keycloak for OpenID Connect**

1. **Create a Realm:**
   * Open the Keycloak Admin Console.
   * In the top-left corner, click the dropdown to select or add a new realm.
   * Click "Add realm", enter a name (e.g., oidc-realm), and click "Create".
2. **Configure Clients:**
   * Follow the steps outlined in Demo 1 to create a client for your application.
   * Ensure that the client protocol is set to openid-connect.
3. **Configure Client Settings:**
   * Configure the client settings as required for your application, including redirect URIs and web origins.
4. **Set Up Client Credentials:**
   * Note down the client secret generated for the client, as it will be needed for integration with your application.

**Part 3: Integrating Application with OpenID Connect**

1. **Integrate OpenID Connect with JavaScript Application:**
   * Include the Keycloak JavaScript adapter in your HTML file:

<script src="https://cdnjs.cloudflare.com/ajax/libs/keycloak/11.0.3/keycloak.min.js"></script>

* + Initialize Keycloak in your JavaScript code:

var keycloak = new Keycloak({

url: 'http://localhost:8080/auth',

realm: 'oidc-realm',

clientId: 'your-client-id'

});

keycloak.init({ onLoad: 'login-required' }).then(function(authenticated) {

console.log(authenticated ? 'Authenticated' : 'Not authenticated');

}).catch(function() {

console.log('Failed to initialize');

});

1. **Integrate OpenID Connect with Spring Boot Application:**
   * Add Keycloak dependencies to pom.xml:

<dependency>

<groupId>org.keycloak</groupId>

<artifactId>keycloak-spring-boot-starter</artifactId>

<version>12.0.4</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

* + Configure application.properties:

keycloak.realm=oidc-realm

keycloak.auth-server-url=http://localhost:8080/auth

keycloak.resource=your-client-id

keycloak.credentials.secret=<client-secret>

keycloak.use-resource-role-mappings=true

security.basic.enabled=false

* + Create a security configuration class:

@Configuration

@EnableWebSecurity

public class SecurityConfig extends KeycloakWebSecurityConfigurerAdapter {

@Bean

@Override

protected SessionAuthenticationStrategy sessionAuthenticationStrategy() {

return new RegisterSessionAuthenticationStrategy(new SessionRegistryImpl());

}

@Override

protected void configure(HttpSecurity http) throws Exception {

super.configure(http);

http.authorizeRequests()

.anyRequest().authenticated();

}

}

**Part 4: Testing OpenID Connect Integration**

1. **Access Your Application:**
   * Open a web browser and navigate to your application.
   * You should be redirected to the Keycloak login page.
2. **Authenticate with Keycloak:**
   * Log in with your Keycloak credentials.
   * After successful authentication, you should be redirected back to your application.
3. **Verify Authentication:**
   * Ensure that the authentication process is successful and that you can access protected resources within your application.

**Demo 5: Authorization with OAuth 2.0**

**Implementing OAuth 2.0:**

1. **Configure OAuth 2.0 Clients:**
   * Navigate to the "Clients" section.
   * Create or select an existing client.
   * Ensure **Client Protocol** is set to openid-connect.
2. **Set Up Roles, Scopes, and Permissions:**
   * Go to the "Roles" section and define necessary roles.
   * Assign these roles to your client and users as needed.
3. **Use OAuth 2.0 Tokens:**
   * Utilize the **Authorization** and **Token** endpoints for obtaining tokens.
   * Integrate token usage in your application to access protected resources.

**Step-by-Step Process:**

**Part 1: Understanding OAuth 2.0**

**What is OAuth 2.0?**

* **Definition:** OAuth 2.0 is an authorization framework that enables third-party applications to obtain limited access to a user's resources without exposing their credentials.
* **Key Concepts:**
  + **Authorization Server:** The server that issues access tokens after successfully authenticating the user and obtaining their consent.
  + **Client:** The application requesting access to the user's resources.
  + **Resource Owner:** The user who owns the resources being accessed.
  + **Access Token:** A token issued by the authorization server that grants access to specific resources on behalf of the user.
  + **Scope:** Permissions granted to the client by the resource owner.

**Part 2: Setting Up Keycloak for OAuth 2.0 Authorization**

1. **Create a Realm:**
   * Open the Keycloak Admin Console.
   * Create a new realm or select an existing one.
   * Click on "Clients" in the left-hand menu and create a new client for your application.
   * Configure the client settings, including redirect URIs and client permissions.
2. **Define Scopes:**
   * Go to the "Clients" section and select your client.
   * Navigate to the "Scope" tab.
   * Define custom scopes that represent the permissions required by your application (e.g., read, write).
3. **Configure Client Roles:**
   * Define roles that represent the access levels granted to users.
   * Go to the "Roles" tab in the client settings and create roles such as user, admin, etc.

**Part 3: Integrating OAuth 2.0 Authorization with Your Application**

1. **JavaScript Application Integration:**
   * Include the Keycloak JavaScript adapter in your HTML file.
   * Initialize Keycloak and obtain an access token based on the desired scope.
2. **Spring Boot Application Integration:**
   * Add Keycloak dependencies to your pom.xml.
   * Configure Spring Security to use Keycloak for authentication and authorization.
   * Use annotations or configuration to secure endpoints based on required roles or scopes.

**Part 4: Testing OAuth 2.0 Authorization**

1. **Access Your Application:**
   * Open a web browser and navigate to your application.
2. **Authenticate with Keycloak:**
   * Log in with your Keycloak credentials.
3. **Authorize Access:**
   * Upon successful authentication, Keycloak will prompt the user to authorize access based on the requested scope and permissions.
4. **Access Protected Resources:**
   * After authorization, the application should be able to access protected resources based on the granted access token.

**Demo 6: SAML and Application Security**

**Using SAML for Application Security:**

1. **Configure SAML Clients:**
   * Navigate to the "Clients" section.
   * Create a new client and set **Client Protocol** to saml.
2. **Set Up SAML Assertions and Mappings:**
   * Configure the necessary **SAML attributes** and **mappings**.
   * Adjust the **Assertion Consumer Service URL** and other settings.
3. **Integrate Applications with Keycloak:**
   * Use the SAML metadata URL provided by Keycloak to configure your service provider.
   * Ensure your application is set up to accept and process SAML assertions from Keycloak.

**Part 1: Understanding SAML**

**1. What is SAML?**

* **Explanation:** SAML (Security Assertion Markup Language) is a standard protocol used for single sign-on (SSO) authentication. It enables users to access multiple applications using one set of credentials.
* **Example:** Imagine SAML as a ticket that you get from one place (the identity provider) to access different attractions (applications).

**Part 2: Setting Up Keycloak for SAML Integration**

**1. Configure Identity Provider:**

* **Explanation:** In Keycloak, we set it up to be the "identity provider," meaning it will handle user authentication.
* **Steps:**
  + Log in to Keycloak Admin Console.
  + Go to "Identity Providers" and select "SAML".
  + Enter the required information like metadata URL or XML file provided by the service you're integrating with.

**2. Configure Service Provider:**

* **Explanation:** The "service provider" is your application, which relies on Keycloak for user authentication.
* **Steps:**
  + Obtain metadata URL or XML file from Keycloak.
  + Configure your application's SAML settings with this metadata.

**3. Define Attribute Mappings:**

* **Explanation:** This step ensures that the information exchanged between Keycloak and your application is understood correctly.
* **Steps:**
  + Map attributes provided by Keycloak (like username, email) to your application's user attributes.

**4. Test SAML Integration:**

* **Explanation:** It's crucial to test the setup to ensure that users can authenticate successfully.
* **Steps:**
  + Log in to your application using SAML authentication.
  + Verify that user attributes (like username, email) are correctly populated.

**Part 3: Ensuring Application Security**

**1. Secure Endpoints:**

* **Explanation:** Once SAML integration is set up, ensure that only authenticated users can access your application's resources.
* **Steps:**
  + Implement access controls to restrict access to specific endpoints based on user roles or permissions.

**2. Enable Single Sign-On (SSO):**

* **Explanation:** SSO allows users to log in once and access multiple applications without re-entering credentials.
* **Steps:**
  + Configure Keycloak to enable SSO across your application ecosystem.
  + Verify that users can seamlessly access other applications after logging into one.

**3. Implement Role-Based Access:**

* **Explanation:** Use roles to control what users can do within your application.
* **Steps:**
  + Define roles in Keycloak based on your application's access requirements.
  + Assign roles to users, and enforce role-based access controls within your application.

**Demo 7: Access Control with Keycloak**

**Implementing Access Control Models:**

1. **Configure Access Control Policies:**
   * Navigate to the "Authorization" section under your client settings.
   * Create and configure **Policies** (e.g., role-based, time-based).
2. **Use Fine-Grained Authorization Features:**
   * Define **Permissions** and **Resources**.
   * Apply policies to resources to control access.
3. **Set Up and Manage Different Models:**
   * For ABAC, define attributes and use them in policies.
   * For RBAC, manage roles and assign them to users and clients.
   * For UBAC, control access based on individual user identities.
   * For CBAC, use context information in your policies.
   * For Rule-Based, define and apply rules.
   * For Time-Based, set up time constraints in your policies.

**1. Configuring Access Control Policies:**

**Step-by-Step Process:**

1. **Navigate to the Authorization Section:**
   * Open Keycloak Admin Console.
   * Select the realm and the specific client for which you want to configure access control.
   * Click on the "Authorization" tab under the client settings.
2. **Create a Resource:**
   * Go to the "Resources" tab.
   * Click "Create".
   * Fill in the **Resource Name** (e.g., Document).
   * Set the **Type** if necessary (e.g., urn:document).
   * Define **URIs** and **Scopes** if applicable.
   * Click "Save".
3. **Define Policies:**
   * Go to the "Policies" tab.
   * Click "Create".
   * Choose a policy type based on the model you are implementing (e.g., **Role**, **User**, **Aggregate**, **Time**).
   * Fill in the policy details:

**Example - Role-Based Policy:**

* + - **Name:** AdminRolePolicy
    - **Type:** Role
    - **Logic:** Positive
    - **Apply to Roles:** Select admin.

**Example - Time-Based Policy:**

* + - **Name:** BusinessHoursPolicy
    - **Type:** Time
    - **Logic:** Positive
    - **Day Time Conditions:** Set appropriate start and end times for business hours.

**Example - User Attribute-Based Policy:**

* + - **Name:** DepartmentPolicy
    - **Type:** User
    - **Logic:** Positive
    - **User Attributes:** Add conditions (e.g., department == "HR").
  + Click "Save".

1. **Create Permissions:**
   * Go to the "Permissions" tab.
   * Click "Create".
   * Choose a permission type (e.g., **Resource-based**, **Scope-based**).
   * Fill in the permission details:

**Example - Resource-Based Permission:**

* + - **Name:** DocumentPermission
    - **Resource:** Select Document.
    - **Policies:** Add policies (e.g., AdminRolePolicy).

**Example - Scope-Based Permission:**

* + - **Name:** DocumentReadPermission
    - **Resource:** Select Document.
    - **Scopes:** Select view.
    - **Policies:** Add policies (e.g., BusinessHoursPolicy).
  + Click "Save".

**2. Using Fine-Grained Authorization Features:**

**Step-by-Step Process:**

1. **Enable Authorization Services:**
   * Ensure that authorization services are enabled for your client.
2. **Define and Manage Policies:**
   * Create various types of policies as mentioned above.
3. **Set Up Permissions:**
   * Combine policies into permissions that are applied to resources or scopes.

**3. Setting Up and Managing Different Models:**

**Attribute-Based Access Control (ABAC):**

1. **Create User Attributes:**
   * Go to the "Users" section.
   * Select a user.
   * Go to the "Attributes" tab.
   * Add attributes (e.g., department, location).
2. **Create ABAC Policies:**
   * Go to the "Authorization" section.
   * Create a user policy with conditions based on attributes (e.g., department == "HR").
3. **Assign ABAC Policies to Permissions:**
   * Create permissions and assign ABAC policies to control access.

**Role-Based Access Control (RBAC):**

1. **Create Roles:**
   * Go to the "Roles" section.
   * Create roles (e.g., admin, user).
2. **Assign Roles to Users:**
   * Go to the "Users" section.
   * Select a user.
   * Go to the "Role Mappings" tab.
   * Assign roles to users.
3. **Create RBAC Policies:**
   * Go to the "Authorization" section.
   * Create role-based policies.
4. **Assign RBAC Policies to Permissions:**
   * Create permissions and assign role-based policies.

**User-Based Access Control (UBAC):**

1. **Create UBAC Policies:**
   * Go to the "Authorization" section.
   * Create user policies targeting specific users.
2. **Assign UBAC Policies to Permissions:**
   * Create permissions and assign user-based policies.

**Context-Based Access Control (CBAC):**

1. **Create CBAC Policies:**
   * Go to the "Authorization" section.
   * Create policies based on context (e.g., IP address, device type).
2. **Assign CBAC Policies to Permissions:**
   * Create permissions and assign context-based policies.

**Rule-Based Access Control:**

1. **Create Rule-Based Policies:**
   * Go to the "Authorization" section.
   * Create policies using rules (e.g., JavaScript-based conditions).
2. **Assign Rule-Based Policies to Permissions:**
   * Create permissions and assign rule-based policies.

**Time-Based Access Control:**

1. **Create Time-Based Policies:**
   * Go to the "Authorization" section.
   * Create time-based policies with conditions (e.g., specific times or days).
2. **Assign Time-Based Policies to Permissions:**
   * Create permissions and assign time-based policies.

**4. Testing and Verifying Access Control:**

1. **Simulate Access Requests:**
   * Use the **Authorization** tab to simulate access requests.
   * Verify that the correct policies and permissions are applied based on the user, roles, attributes, and context.
2. **Monitor and Audit:**
   * Use the **Events** section to monitor access attempts and audit logs.

**Demo 8: Integrating Applications with Keycloak**

**Integrating with JavaScript Applications:**

1. **Include Keycloak JavaScript Adapter:**
   * Add the Keycloak JavaScript adapter to your project.
   * Initialize Keycloak in your application code:

var keycloak = new Keycloak({

url: 'http://<keycloak-server>/auth',

realm: '<realm>',

clientId: '<client-id>'

});

1. **Initialize and Configure Keycloak:**
   * Initialize Keycloak on page load:

keycloak.init({ onLoad: 'login-required' }).success(function() {

console.log('Authenticated');

}).error(function() {

console.log('Authentication failed');

});

1. **Handle Authentication and Token Management:**
   * Use Keycloak's methods to manage tokens and user sessions in your application.

**Integrating with Spring Boot Applications:**

1. **Include Keycloak Spring Boot Adapter:**
   * Add the Keycloak Spring Boot adapter dependency to your pom.xml:

<dependency>

<groupId>org.keycloak</groupId>

<artifactId>keycloak-spring-boot-starter</artifactId>

<version>VERSION</version>

</dependency>

1. **Configure Keycloak Settings:**
   * Add Keycloak settings to your application.properties:

keycloak.realm=<realm>

keycloak.auth-server-url=http://<keycloak-server>/auth

keycloak.resource=<client-id>

keycloak.credentials.secret=<client-secret>

1. **Manage Authentication and Authorization:**
   * Configure Spring Security to use Keycloak:

@Configuration

@EnableWebSecurity

public class SecurityConfig extends KeycloakWebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

super.configure(http);

http.authorizeRequests()

.anyRequest().authenticated();

}

}

**Other Integrations:**

1. **Node.js:**
   * Use Keycloak Node.js adapter or Keycloak's REST API to integrate Node.js applications.
2. **Python:**
   * Use libraries like python-keycloak or Keycloak's REST API for integration.
3. **Custom REST API:**
   * Use Keycloak’s REST API for custom integrations, accessing endpoints for user, client, and token management.

**Part 1: Integrating a JavaScript Application with Keycloak**

**Step-by-Step Process:**

1. **Setting Up Keycloak:**
   * **Create a Realm:**
     + Open the Keycloak Admin Console.
     + Click on the dropdown in the top-left corner and select "Add realm".
     + Enter a name for your realm (e.g., demo-realm) and click "Create".
   * **Create a Client:**
     + In the Keycloak Admin Console, navigate to the demo-realm.
     + Click on "Clients" in the left-hand menu.
     + Click "Create".
     + Enter a Client ID (e.g., js-client) and select Client Protocol as openid-connect.
     + Click "Save".
   * **Configure Client:**
     + In the "Settings" tab, set the Root URL to your application's URL (e.g., http://localhost:3000).
     + Add Valid Redirect URIs (e.g., http://localhost:3000/\*).
     + Set Web Origins to \* or your specific URL.
     + Click "Save".
2. **Setting Up the JavaScript Application:**
   * **Include Keycloak JavaScript Adapter:**
     + Download the Keycloak JavaScript adapter from the Keycloak website or include it via CDN.
     + Add the following script to your HTML file:

<script src="https://cdnjs.cloudflare.com/ajax/libs/keycloak/11.0.3/keycloak.min.js"></script>

* + **Initialize Keycloak in Your JavaScript Code:**
    - Create a JavaScript file (e.g., app.js) and add the following code:

var keycloak = new Keycloak({

url: 'http://localhost:8080/auth',

realm: 'demo-realm',

clientId: 'js-client'

});

keycloak.init({ onLoad: 'login-required' }).success(function(authenticated) {

console.log('Authenticated');

}).error(function() {

console.log('Authentication failed');

});

* + **Add Authentication Check:**
    - Ensure that your application checks if the user is authenticated before allowing access to protected resources:

if (keycloak.authenticated) {

document.getElementById('welcome').innerText = 'Welcome, ' + keycloak.tokenParsed.preferred\_username;

} else {

keycloak.login();

}

1. **Running the JavaScript Application:**
   * Host your JavaScript application (e.g., using a simple HTTP server like http-server).
   * Open your application in a web browser (e.g., http://localhost:3000).
   * You should be redirected to the Keycloak login page. After logging in, you'll be redirected back to your application.

**Part 2: Integrating a Spring Boot Application with Keycloak**

**Step-by-Step Process:**

1. **Setting Up Keycloak:**
   * **Create a Realm:**
     + Open the Keycloak Admin Console.
     + Click on the dropdown in the top-left corner and select "Add realm".
     + Enter a name for your realm (e.g., demo-realm) and click "Create".
   * **Create a Client:**
     + In the Keycloak Admin Console, navigate to the demo-realm.
     + Click on "Clients" in the left-hand menu.
     + Click "Create".
     + Enter a Client ID (e.g., spring-boot-client) and select Client Protocol as openid-connect.
     + Click "Save".
   * **Configure Client:**
     + In the "Settings" tab, set the Root URL to your Spring Boot application’s URL (e.g., http://localhost:8080).
     + Add Valid Redirect URIs (e.g., http://localhost:8080/\*).
     + Set Web Origins to \* or your specific URL.
     + Click "Save".
2. **Setting Up the Spring Boot Application:**
   * **Include Keycloak Dependencies:**
     + Add the following dependencies to your pom.xml:

<dependency>

<groupId>org.keycloak</groupId>

<artifactId>keycloak-spring-boot-starter</artifactId>

<version>12.0.4</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

* + **Configure Keycloak Settings:**
    - Add the following configuration to your application.properties:

keycloak.realm=demo-realm

keycloak.auth-server-url=http://localhost:8080/auth

keycloak.resource=spring-boot-client

keycloak.credentials.secret=<client-secret>

keycloak.use-resource-role-mappings=true

# Spring Security properties

security.basic.enabled=false

* + **Create a Security Configuration Class:**
    - Create a class SecurityConfig.java to configure Spring Security with Keycloak:

import org.keycloak.adapters.springsecurity.KeycloakSecurityComponents;

import org.keycloak.adapters.springsecurity.config.KeycloakWebSecurityConfigurerAdapter;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.core.session.SessionRegistryImpl;

import org.springframework.security.web.authentication.session.RegisterSessionAuthenticationStrategy;

import org.springframework.security.web.authentication.session.SessionAuthenticationStrategy;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends KeycloakWebSecurityConfigurerAdapter {

@Bean

@Override

protected SessionAuthenticationStrategy sessionAuthenticationStrategy() {

return new RegisterSessionAuthenticationStrategy(new SessionRegistryImpl());

}

@Override

protected void configure(HttpSecurity http) throws Exception {

super.configure(http);

http.authorizeRequests()

.anyRequest().authenticated();

}

}

1. **Running the Spring Boot Application:**
   * Start your Spring Boot application.
   * Access your application in a web browser (e.g., http://localhost:8080).
   * You should be redirected to the Keycloak login page. After logging in, you'll be redirected back to your Spring Boot application.