

# Java User Group Saarland

Keycloak Extension Development  
Overview and Best Practices

Thomas Darimont

56. Meeting

*06. April 2021*



 codecentric

*Sponsored by*

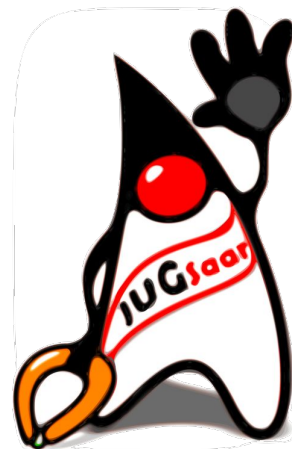
 eurodata

  
INFOSERVE  
eurodata-Gruppe

# Thomas Darimont

---

- Fellow @codecentric
- Pivotal Spring Team Alumni
- Open Source Enthusiast
- Java User Group Saarland
- Keycloak Contributor for over 5 years



@thomasdarimont  
@jugsaar

# The Journey

---



Keycloak Overview



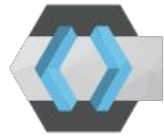
Extending Keycloak



Development Setups



Q&A



# **Keycloak**

## **Overview**



# Open Source Identity and Access Management

## For Modern Applications and Services

Get Started with Keycloak

Add authentication to applications and secure services with minimum fuss. No need to deal with storing users or authenticating users. It's all available out of the box.

You'll even get advanced features such as User Federation, Identity Brokering and Social Login.

For more details go to [about](#) and [documentation](#), and don't forget to try [Keycloak](#). It's easy by design!

### NEWS

<https://www.keycloak.org>

**01 Mar**

Keycloak 12.0.4 released

**16 Feb**

Keycloak 12.0.3 released

**19 Jan**

Keycloak 12.0.2 released



#### Single-Sign On

Login once to multiple applications



#### Standard Protocols

OpenID Connect, OAuth 2.0  
and SAML 2.0



#### Centralized Management

For admins and users



#### Adapters

Secure applications and services  
easily

# Project

---



- Java based **Authentication & Authorization** Server
- Started in 2013, broad adoption since 2015
- Apache License, **Red Hat** Developers
- Vital **Community** with **480+** Contributors
- **Keycloak Community** Free Version (current 12.0.4)
- **Red Hat SSO** Commercial Offering
- Very **robust**, good **documentation**, many **examples**

# Authentication & Authorization

---

- Authentication (AuthN)
  - Determines ***who** the user **is***
  - Internal & Federated User Storage Kerberos, LDAP, Custom
  - Customizable
- Authorization (AuthZ)
  - Determines ***what** the user **is allowed** to do*
  - Hierarchical Role-based Access Control (HRBAC)
  - Authorization Services
    - Flexible [Access Control Management](#)
    - More Variants like ABAC, UBAC, CBAC supported

# Features

---

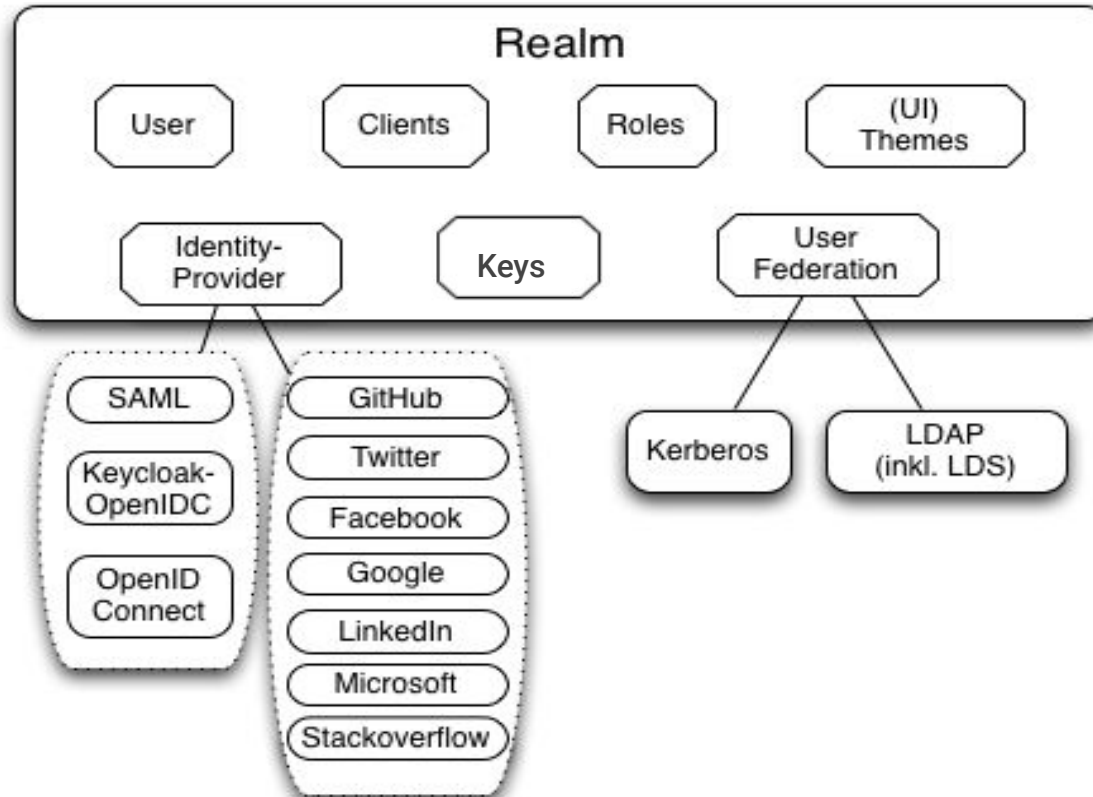


- **Single Sign-on** and Single Sign-out
- **Standard Protocols** OAuth 2.0, OIDC 1.0, SAML 2.0
- Flexible **Authentication** and **Authorization**
- **Multi-Factor Authentication** One-time Passwords
- **Social Login** Google, Facebook, Twitter,..., AzureAD, ADFS, Auth0
- Supports **Directory Services** LDAP, AD, Kerberos
- Provides centralized **User Management**
- **Customizable** and **Extensible**
- **Easy** Setup and Integration



# Main Concepts

---





# Keycloak Quick Tour

[Admin Console](#)

# Admin Console



KEYCLOAK

Admin

Acme

Configure

Realm Settings

Clients

Client Scopes

Roles

Identity Providers

User Federation

Authentication

Manage

Groups

Users

Sessions

Events

Import

Export

Acme

General

Login

Keys

Email

Themes

Cache

Tokens

Client Registration

Security Defenses

\* Name

acme

Display name

Acme Inc.

HTML Display name

<strong>Acme Inc.</strong>

Enabled ?

ON

User-Managed Access ?

OFF

Endpoints ?

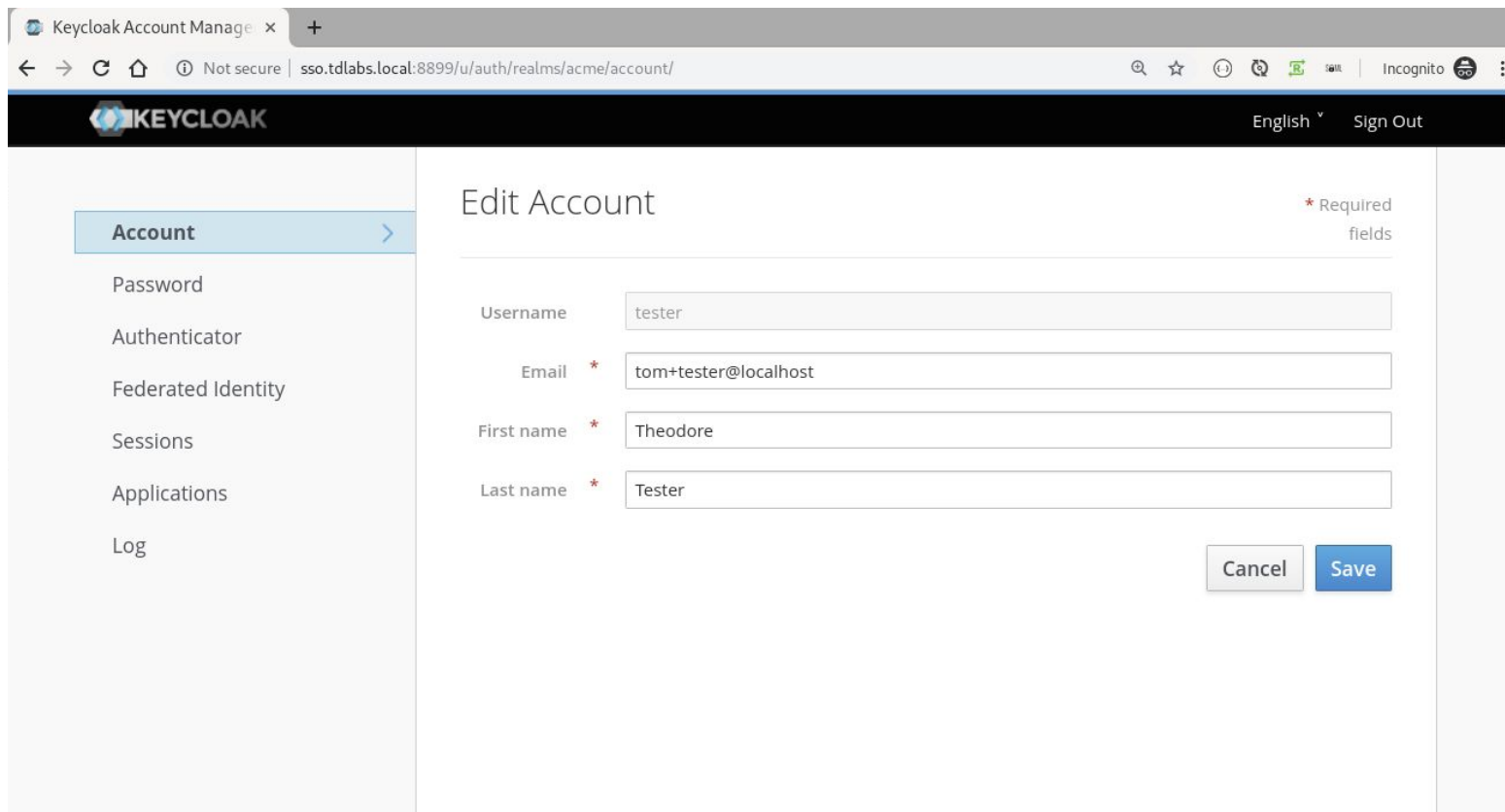
OpenID Endpoint Configuration

SAML 2.0 Identity Provider Metadata

Save

Cancel


# Self-Service Account Management



The screenshot displays the Keycloak Account Management interface in a web browser. The browser's address bar shows the URL `sso.tdlabs.local:8899/u/auth/realms/acme/account/`. The page features a dark header with the Keycloak logo, the text "English", and a "Sign Out" link. A left-hand sidebar contains a menu with the following items: "Account" (highlighted with a blue bar and a right-pointing arrow), "Password", "Authenticator", "Federated Identity", "Sessions", "Applications", and "Log". The main content area is titled "Edit Account" and includes a legend indicating that red asterisks (\*) denote required fields. The form contains four input fields: "Username" with the value "tester", "Email" with the value "tom+tester@localhost", "First name" with the value "Theodore", and "Last name" with the value "Tester". At the bottom right of the form are two buttons: a grey "Cancel" button and a blue "Save" button.

Keycloak Account Manage x +

← → ↻ ⌂ ⓘ Not secure | sso.tdlabs.local:8899/u/auth/realms/acme/account/ 🔍 ☆ ⓘ ⌂ 📄 📄 Incognito 🧑

 KEYCLOAK English Sign Out

**Account** >

- Password
- Authenticator
- Federated Identity
- Sessions
- Applications
- Log

## Edit Account

\* Required fields

Username

Email \*

First name \*

Last name \*

Cancel Save

# Keycloak Technology Stack

## Admin Console

- Angular JS
- PatternFly
- Bootstrap

## Keycloak Server

- Wildfly / Undertow
- JAX-RS (Resteasy)
- JPA (Hibernate)
- Infinispan (JGroups)
- Freemarker
- Jackson 2.x
- Liquibase
- JBoss Logging
- Apache Directory API
- Commons HTTP Client



# Keycloak-X (Preview) Technology Stack

## Admin Console

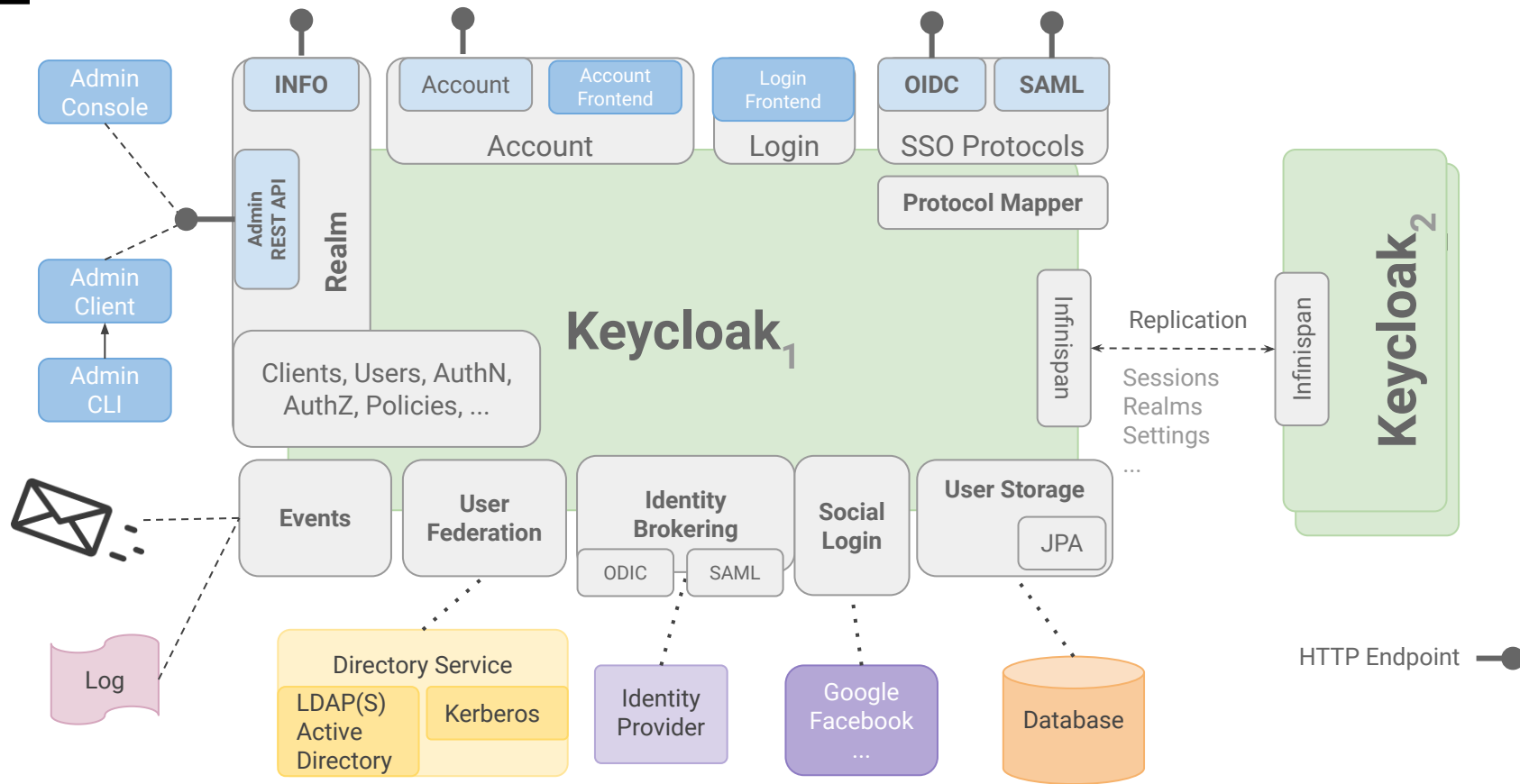
- Angular JS
- PatternFly
- Bootstrap

## Keycloak-X Server

- Quarkus / Vert.x
- JAX-RS (Resteasy)
- JPA (Hibernate)
- Infinispan (JGroups)
- Freemarker
- Jackson 2.x
- Liquibase
- JBoss Logging
- Apache Directory API
- Commons HTTP Client



# Server Architecture







# **Extending Keycloak**







# Keycloak-Extensions-Talk Github Repository

 thomasdarimont / **keycloak-extensions-talk** Unwatch

[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Settings](#)

main 1 branch 0 tags  +  ↓

 **thomasdarimont** [Revise readme](#) 9247ade now 🕒 7 commits 


## Easy Keycloak Extension Development

This repository contains the code & slides of my talk `Keycloak Extension Development - Overview & Best Practices`.

This example shows how to develop and deploy a set of Keycloak extensions, custom themens and configuration to a Keycloak docker container. In addition to that, the example also shows how to write integration tests via Keycloak-Testcontainers and how to package all extensions and themes as a custom docker image.

The example contains the following Keycloak extensions:

- OIDC ProtocolMapper to compute age-class information based on a `birthdate` user attribute: `AgeInfoMapper`
- Audit Event Listener sketch to forward certain Keycloak user and admin events to an external service: `AcmeAuditListener`
- Custom REST Endpoint the can expose additional custom APIs: `AcmeResource`



Java	62.4%	HTML	32.9%
Shell	2.0%	CSS	1.4%
Other	1.3%		

# Keycloak Extension Points

---

- Service Provider Interfaces (SPI) based Extension Mechanism
  - Extension Point = SPI
  - Extension = Provider and ProviderFactory for an SPI
- Many SPIs built-in
  - ~ 100 Service Provider Interfaces (SPIs) + ~ 500 Providers
  - Based on Java Service Loader (since Java 6)
- Highlights
  - Authentication Mechanisms
  - SAML / OIDC Protocol Mapper
  - Event Listener (Auditing, Provisioning)
  - User Storage (JDBC, REST, etc.) & Credential Hashing Mechanisms
- Possible to create own Extension Points
  - Requires three classes (Spi Implementation class, Provider and ProviderFactory interface)

# Keycloak SPI Overview (Keycloak 12.0.4)

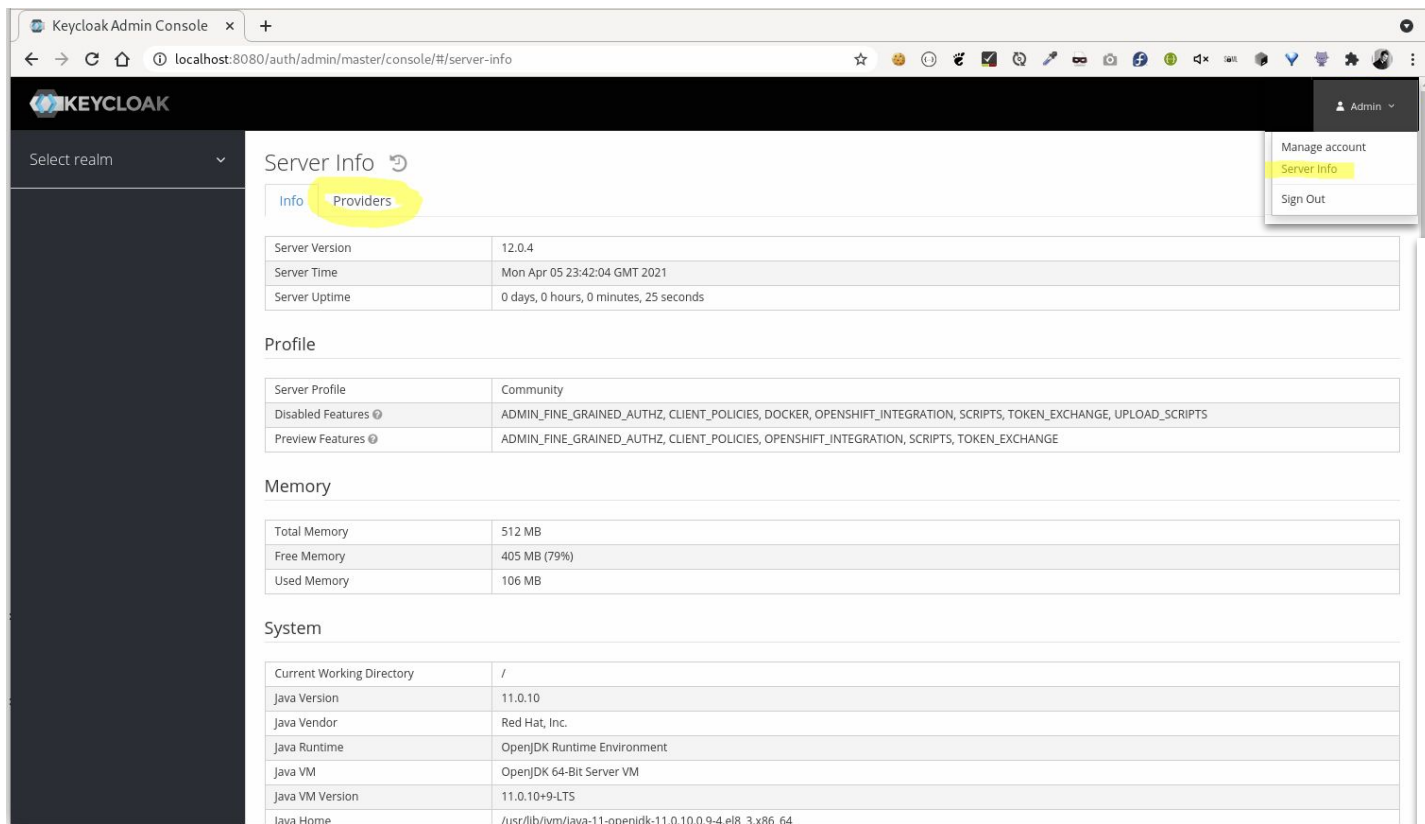
- ClientScopeSpi (org.keycloak.models)
- ImportSpi (org.keycloak.exportimport)
- ExecutorsSpi (org.keycloak.executors)
- IdentityProviderSpi (org.keycloak.broker.provider)
- ActionTokenStoreSpi (org.keycloak.models)
- EmailTemplateSpi (org.keycloak.email)
- PasswordPolicyManagerSpi (org.keycloak.policy)
- RequiredActionSpi (org.keycloak.authentication)
- RoleStorageProviderSpi (org.keycloak.storage.role)
- OIDCExtSpi (org.keycloak.protocol.oidc.ext)
- ActionTokenHandlerSpi (org.keycloak.authentication.actiontoken)
- SecurityHeadersSpi (org.keycloak.headers)
- SignatureSpi (org.keycloak.crypto)
- ClientDescriptionConverterSpi (org.keycloak.exportimport)
- InfinispanConnectionSpi (org.keycloak.connections.infinispan)
- HostnameSpi (org.keycloak.urls)
- ExceptionConverterSpi (org.keycloak.provider)
- ThemeSpi (org.keycloak.theme)
- AuthenticatorSpi (org.keycloak.authentication)
- ProtocolMapperSpi (org.keycloak.protocol)
- FormActionSpi (org.keycloak.authentication)
- PublicKeyStorageSpi (org.keycloak.keys)
- ClientPolicyConditionSpi (org.keycloak.services.clientpolicy.condition)
- ExportSpi (org.keycloak.exportimport)
- MapStorageSpi (org.keycloak.models.map.storage)
- ClientInstallationSpi (org.keycloak.protocol)
- ClientRegistrationPolicySpi (org.keycloak.services.clientregistration.policy)
- EventListenerSpi (org.keycloak.events)
- ClientValidationSpi (org.keycloak.models.cache.authorization)
- CachedStoreFactorySpi (org.keycloak.models)
- RealmSpi (org.keycloak.models)
- ClientPolicySpi (org.keycloak.services.clientpolicy)
- LocaleUpdaterSpi (org.keycloak.locale)
- ClientSignatureVerifierSpi (org.keycloak.crypto)
- TransactionManagerLookupSpi (org.keycloak.transaction)
- UserSessionSpi (org.keycloak.models)
- TokenRevocationStoreSpi (org.keycloak.models)
- SocialProviderSpi (org.keycloak.broker.social)
- OAuth2DeviceUserCodeSpi (org.keycloak.models)
- ResourceEncodingSpi (org.keycloak.encoding)
- RoleSpi (org.keycloak.models)
- ClientStorageProviderSpi (org.keycloak.storage.client)
- CodeToTokenStoreSpi (org.keycloak.models)
- EmailSenderSpi (org.keycloak.email)
- LoginFormsSpi (org.keycloak.forms.login)
- WellKnownSpi (org.keycloak.wellknown)
- PasswordHashSpi (org.keycloak.credential.hash)
- HashSpi (org.keycloak.crypto)
- CredentialSpi (org.keycloak.credential)
- UserFederatedStorageProviderSpi (org.keycloak.storage.federated)

- ClientRegistrationSpi (org.keycloak.services.clientregistration)
- KeySpi (org.keycloak.keys)
- StickySessionEncoderSpi (org.keycloak.sessions)
- UserStorageProviderSpi (org.keycloak.storage)
- LiquibaseConnectionSpi (org.keycloak.connections.jpa.updater.liquibase.conn)
- ComponentFactorySpi (org.keycloak.component)
- X509ClientCertificateLookupSpi (org.keycloak.services.x509)
- OAuth2DeviceTokenStoreSpi (org.keycloak.models)
- DBLockSpi (org.keycloak.models.dblock)
- HttpClientSpi (org.keycloak.connections.httpclient)
- TrustStoreSpi (org.keycloak.truststore)
- ClusterSpi (org.keycloak.cluster)
- ContentEncryptionSpi (org.keycloak.crypto)
- UserSpi (org.keycloak.models)
- TimerSpi (org.keycloak.timer)
- VaultSpi (org.keycloak.vault)
- SamlAuthenticationPreprocessorSpi (org.keycloak.protocol.saml.preprocessor)
- PolicySpi (org.keycloak.authorization.policy.provider)
- JpaUpdaterSpi (org.keycloak.connections.jpa.updater)
- TestAmphibianSpi (org.keycloak.testsuite.components.amphibian)
- TokenIntrospectionSpi (org.keycloak.protocol.oidc)
- LoginProtocolSpi (org.keycloak.protocol)
- ClientAuthenticatorSpi (org.keycloak.authentication)
- SingleUseTokenStoreSpi (org.keycloak.models)
- ClientSpi (org.keycloak.models)
- CacheRealmProviderSpi (org.keycloak.models.cache)
- ServerInfoSpi (org.keycloak.models)
- LDAPStorageMapperSpi (org.keycloak.storage.ldap.mappers)
- UserProfileSpi (org.keycloak.userprofile)
- StoreFactorySpi (org.keycloak.authorization.store)
- ClientScopeStorageProviderSpi (org.keycloak.storage.clientscope)
- ThemeSelectorSpi (org.keycloak.theme)
- CdkManagementSpi (org.keycloak.crypto)
- PasswordPolicySpi (org.keycloak.policy)
- GroupSpi (org.keycloak.models)
- FormAuthenticatorSpi (org.keycloak.authentication)
- IdentityProviderMapperSpi (org.keycloak.broker.provider)
- ScriptingSpi (org.keycloak.scripting)
- EventStoreSpi (org.keycloak.events)
- ExampleSpi (org.keycloak.testsuite.domainextension.spi)
- RealmResourceSpi (org.keycloak.services.resource)
- BruteForceProtectorSpi (org.keycloak.services.managers)
- AuthorizationSpi (org.keycloak.authorization)
- LocaleSelectorSpi (org.keycloak.locale)
- TestSpi (org.keycloak.testsuite.components)
- AuthenticationSessionSpi (org.keycloak.sessions)
- ClientPolicyExecutorSpi (org.keycloak.services.clientpolicy.executor)
- AccountSpi (org.keycloak.forms.account)
- ThemeResourceSpi (org.keycloak.theme)
- MigrationSpi (org.keycloak.migration)

- UserSessionPersisterSpi (org.keycloak.models.session)
- JpaEntitySpi (org.keycloak.connections.jpa.entityprovider)
- JpaConnectionSpi (org.keycloak.connections.jpa)
- CacheUserProviderSpi (org.keycloak.models.cache)
- ExampleSpi (org.keycloak.examples.domainextension.spi)
- GroupStorageProviderSpi (org.keycloak.storage.group)

~100 SPIs

# List Installed SPI Providers in Admin Console



The screenshot shows the Keycloak Admin Console interface. The browser address bar indicates the URL is `localhost:8080/auth/admin/master/console/#/server-info`. The left sidebar contains a 'Select realm' dropdown. The main content area is titled 'Server Info' and has two tabs: 'Info' and 'Providers'. The 'Providers' tab is currently selected and highlighted with a yellow circle. A dropdown menu is open in the top right corner, showing options: 'Manage account', 'Server Info' (highlighted), and 'Sign Out'.

**Server Info**

Info Providers

Server Version	12.0.4
Server Time	Mon Apr 05 23:42:04 GMT 2021
Server Uptime	0 days, 0 hours, 0 minutes, 25 seconds

**Profile**

Server Profile	Community
Disabled Features	ADMIN_FINE_GRAINED_AUTHZ, CLIENT_POLICIES, DOCKER, OPENSIFT_INTEGRATION, SCRIPTS, TOKEN_EXCHANGE, UPLOAD_SCRIPTS
Preview Features	ADMIN_FINE_GRAINED_AUTHZ, CLIENT_POLICIES, OPENSIFT_INTEGRATION, SCRIPTS, TOKEN_EXCHANGE

**Memory**

Total Memory	512 MB
Free Memory	405 MB (79%)
Used Memory	106 MB

**System**

Current Working Directory	/
Java Version	11.0.10
Java Vendor	Red Hat, Inc.
Java Runtime	OpenJDK Runtime Environment
Java VM	OpenJDK 64-Bit Server VM
Java VM Version	11.0.10+9-LTS
Java Home	/usr/lib/jvm/java-11-openjdk-11.0.10.0-9-4.el8_3.x86_64

## Server Info

Info Providers

Search...

SPI	Providers
account	freemarker
actionToken	infinispan
actionTokenHandler	verify-email execute-actions reset-credentials idp-verify-account-via-email
authenticationSessions	infinispan
authenticator	no-cookie-redirect auth-cookie console-username-password reset-credentials-choose-user direct-grant-validate-password webauthn-authenticator auth-spnego direct-grant-auth-x509-username reset-password auth-password-form docker-http-basic-authenticator idp-username-password-form auth-x509-client-username-form idp-auto-link idp-email-verification basic-auth conditional-user-role direct-grant-validate-username identity-provider-redirector reset-otp

# Keycloak SPI Example: AuthenticatorSpi

```
package org.keycloak.authentication;

import org.keycloak.provider.Provider;
import org.keycloak.provider.ProviderFactory;
import org.keycloak.provider.Spi;

/** @author <a href="mailto:sthorger@redhat.com">Stian Thorgeresen</a> */
public class AuthenticatorSpi implements Spi {

    ... public static final String SPI_NAME = "authenticator";

    ... @Override
    public boolean isInternal() { return true; }

    ... @Override
    public String getName() { return SPI_NAME; }

    ... @Override
    public Class<? extends Provider> getProviderClass() { return Authenticator.class; }

    ... @Override
    public Class<? extends ProviderFactory> getProviderFactoryClass() { return AuthenticatorFactory.class; }

}
```

The "identifier" of the SPI used for lookups.

SPI Implementors need to implement this interface.  
→ **Provider**

SPI Implementors need to implement this interface.  
→ **Factory** to create Provider instances

# How to create a new Extension?

---

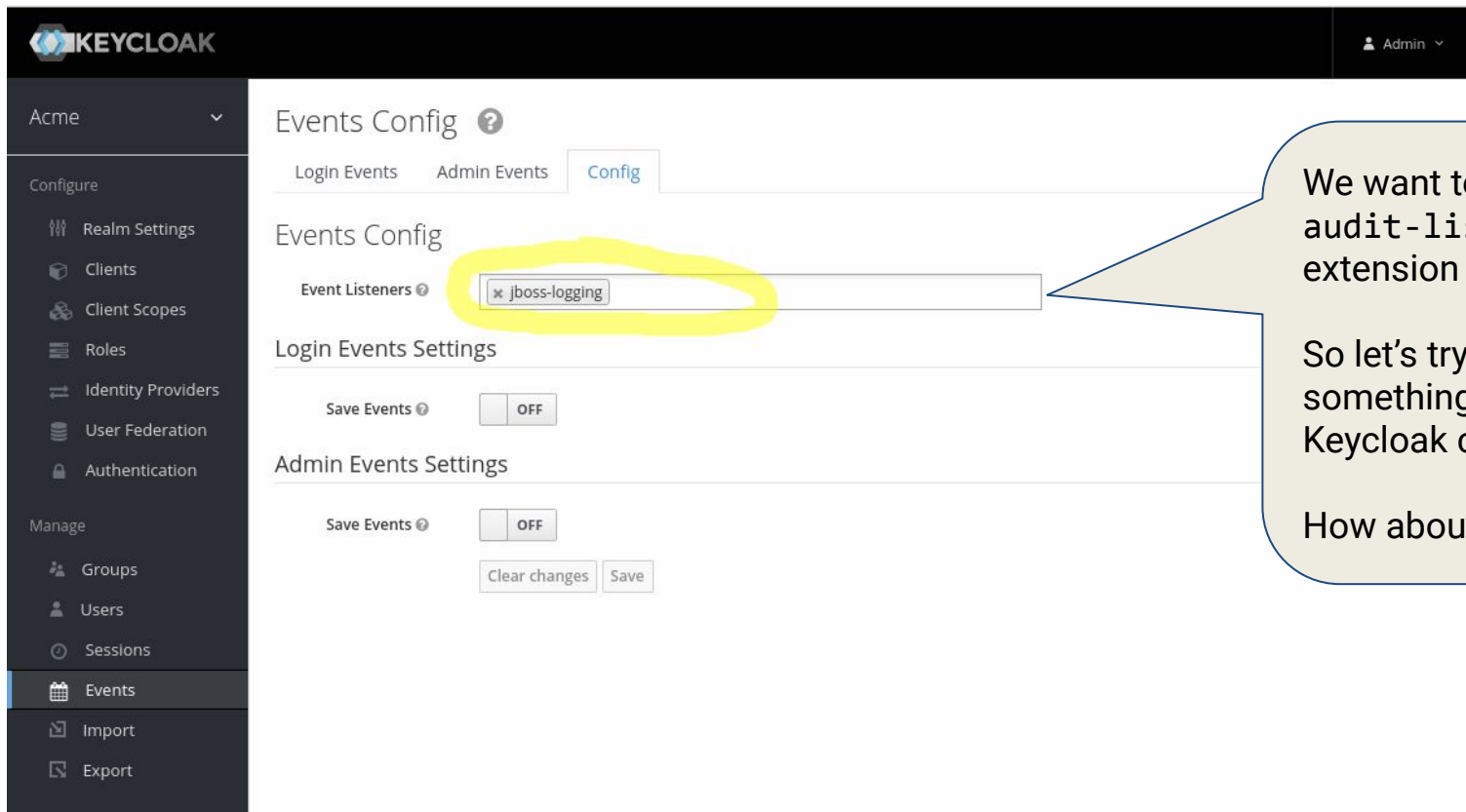
- Read [Keycloak Server Developer Documentation](#)
- Find SPI → `org.keycloak.provider.Spi`
- Find which Interfaces to implement
- Implement Interfaces
- Create or generate Service Manifest
- Package Extension
- Deploy Extension
- Configure Extension
- Profit!

# Example: AuditListener Extension

## Use Case

*“We want to forward relevant user events (e.g. login, logout, password changes) to an external audit-service.”*

# How to start? Which SPI do I need?



The screenshot shows the Keycloak Admin Console interface. The top navigation bar includes the Keycloak logo and a user profile labeled 'Admin'. The left sidebar contains a menu with sections: 'Acme' (selected), 'Configure' (containing Realm Settings, Clients, Client Scopes, Roles, Identity Providers, User Federation, and Authentication), and 'Manage' (containing Groups, Users, Sessions, Events, Import, and Export). The 'Events Config' page is displayed, with tabs for 'Login Events', 'Admin Events', and 'Config' (selected). The 'Event Listeners' field is highlighted with a yellow circle and contains the text 'x jboss-logging'. Below this are sections for 'Login Events Settings' and 'Admin Events Settings', each with a 'Save Events' toggle set to 'OFF'. At the bottom of the 'Admin Events Settings' section are 'Clear changes' and 'Save' buttons.

We want to add our audit-listener extension here.

So let's try to find something similar in the Keycloak codebase.

How about "jboss-login"?



# Custom Extension Example: AuditListener

---

- Spi: `org.keycloak.events.EventListenerSpi`
- Spi requires two interfaces
  - `EventListenerProvider`
  - `EventListenerProviderFactory`
- Implement interfaces
- Create or generate Service Manifest
  - in `src/main/resources/META-INF/services`
  - Create a file `org.keycloak.events.EventListenerProviderFactory`
  - With the FQCN of the `EventListenerProviderFactory` implementation class
- Package and Deploy Extension
  - Copy `extension.jar` to `$KEYCLOAK_HOME/standalone/deployments`
- Profit
  - Configure AuditListener in Admin-Console -> Realm -> Events -> Config

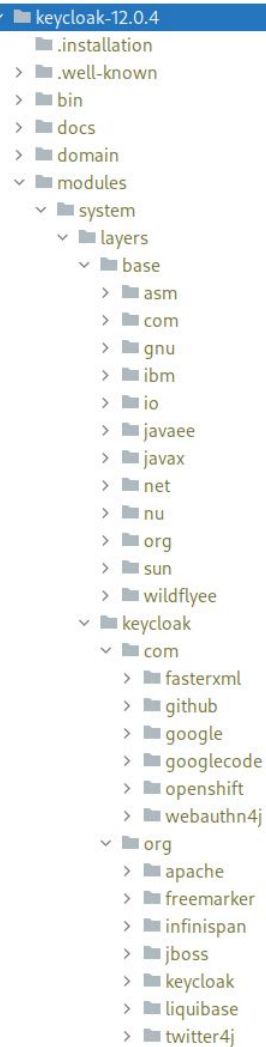
# AuditListener Extension Dependencies

- Uses provided Keycloak Dependencies
  - keycloak-server-spi
  - keycloak-server-spi-private
  - keycloak-services
  - ...
- Keycloak ships with many built-in libraries
  - \$KEYCLOAK\_HOME/modules
  - Sufficient for most extensions

```
<dependencies>
<dependency>
<groupId>org.keycloak</groupId>
<artifactId>keycloak-server-spi</artifactId>
<version>${keycloak.version}</version>
<scope>provided</scope>
</dependency>

<!-- Required for Event Listener SPI -->
<dependency>
<groupId>org.keycloak</groupId>
<artifactId>keycloak-server-spi-private</artifactId>
<version>${keycloak.version}</version>
<scope>provided</scope>
</dependency>

<dependency>
<groupId>org.keycloak</groupId>
<artifactId>keycloak-services</artifactId>
<version>${keycloak.version}</version>
<scope>provided</scope>
</dependency>
```



# AuditListener Extension Implementation Sketch

---

```
package demo.acme.keycloak.audit;

import org.keycloak.events.*;

public class AcmeAuditListener implements EventListenerProvider {

    public static final String ID = "acme-audit-listener";

    @Override
    public void onEvent(Event event) {
        // called for each User-Event
    }

    @Override
    public void onEvent(AdminEvent event, boolean includeRep) {
        // called for each AdminEvent
    }

    @Override
    public void close() {
        // NOOP
    }
}
```

```
package demo.acme.keycloak.audit;

import org.keycloak.Config;
import org.keycloak.events.*;
import org.keycloak.models.*;

public class AcmeAuditListenerFactory implements EventListenerProviderFactory {

    private static final AcmeAuditListener INSTANCE = new AcmeAuditListener();

    @Override
    public String getId() { return AcmeAuditListener.ID; }

    @Override // return singleton instance, or create new AcmeAuditListener(session)
    public EventListenerProvider create(KeycloakSession session) { return INSTANCE; }

    @Override // we could read settings from the provider config in standalone(-ha).xml
    public void init(Config.Scope config) { }

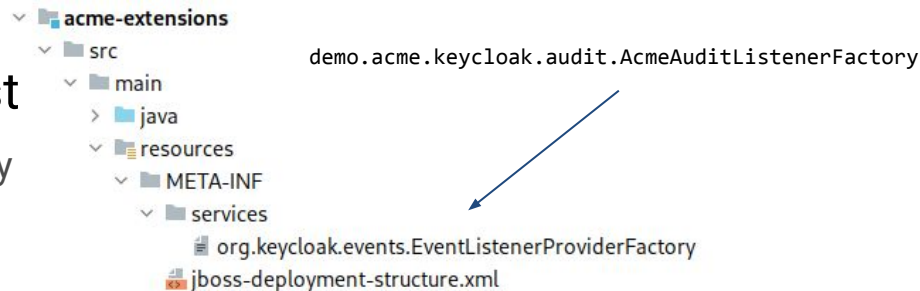
    @Override // we could init our provider with information from other providers
    public void postInit(KeycloakSessionFactory factory) { }

    @Override // close resources if necessary
    public void close() { }
}
```

# AuditListener Extension Packaging Config

- Keycloak Extension Service Manifest

- Needed by Keycloak for Extension Discovery
- in META-INF/services
- with the FQCN of the Implementation



- Keycloak Module Dependencies

- We need to declare the used Keycloak Libraries

```
<?xml version="1.0" encoding="UTF-8"?>
<jboss-deployment-structure>
  <!-- a generous collection of dependencies
  to simplify the development of extensions -->
  <dependencies>
    <module name="org.keycloak.keycloak-common"/>
    <module name="org.keycloak.keycloak-core"/>
    <module name="org.keycloak.keycloak-server-spi"/>
    <module name="org.keycloak.keycloak-server-spi-private"/>
    <module name="org.jboss.resteasy.resteasy-jaxrs"/>
    <module name="org.apache.httpcomponents"/>
    <module name="org.keycloak.keycloak-services"/>
    <module name="org.bouncycastle"/>
    <module name="com.google.guava"/>
    <module name="org.jboss.logging"/>
    <module name="org.apache.commons.io"/>
    <module name="javax.api"/>
    <module name="javax.transaction.api"/>
    <module name="com.fasterxml.jackson.core.jackson-core"/>
    <module name="com.fasterxml.jackson.core.jackson-annotations"/>
    <module name="com.fasterxml.jackson.core.jackson-databind"/>
    <module name="com.fasterxml.jackson.core.jackson-databind"/>
  </dependencies>
</jboss-deployment-structure>
```

jboss-deployment-structure.xml

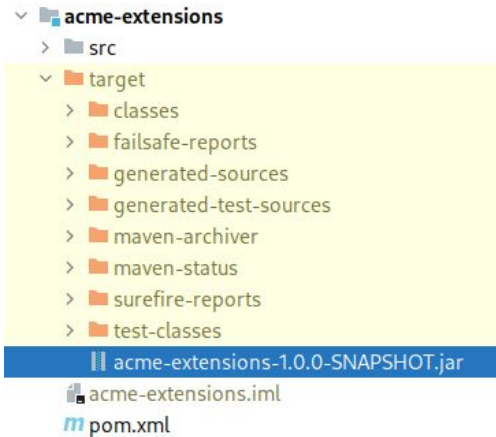
Lists the Keycloak Libraries needed by the extension. These are the **jboss-modules** from the /modules Folder in \$KEYCLOAK\_HOME.

The **module-name** can be found in the respective **module.xml** file.

# AuditListener Extension Packaging

---

- Extensions can be packaged as .jar or .ear
  - .jar for Extensions that solely rely on Keycloak Libs
  - .ear for Extensions that use custom Libs
- Custom libraries are possible with .jar, e.g. as...
  - Library deployed as jboss-module into Keycloak
  - Library repackaged in a “shaded” uber-jar
- We use a combined acme-extensions.jar
  - we only use Keycloak libraries and deploy multiple extensions in one .jar



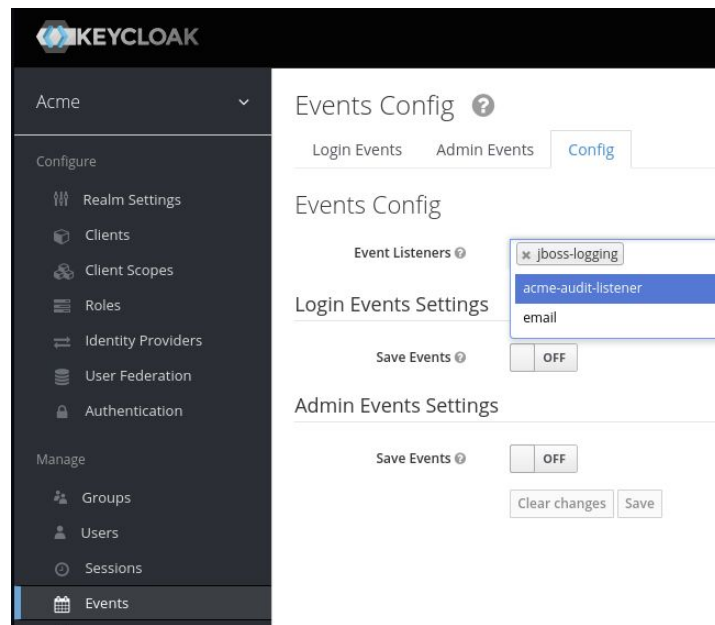
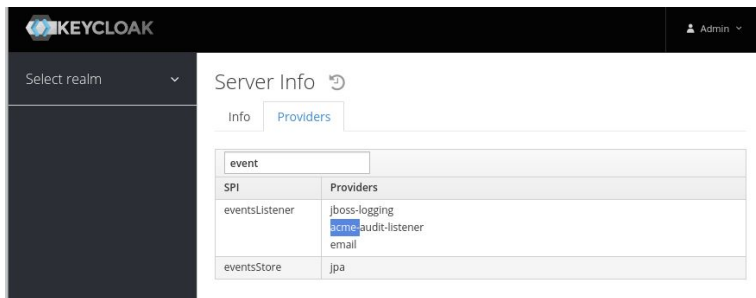
# AuditListener Extension Deployment

---

- Extensions can be deployed as a .jar or .ear (also in exploded format!)
- Simple copy to Keycloak Deployments folder
  - `cp acme-extensions.jar $KEYCLOAK_HOME/standalone/deployments`

# AuditListener Extension Configuration

- The acme-audit-listener appears in the providers list
- We can select the acme-audit-listener in the Events Config



# AuditListener Extension Demo



# Testing Keycloak Extensions

---

- Testing Keycloak Extensions can be tricky...
  - You need start the Keycloak environment
  - Setup Keycloak with test data → Import predefined realm.json
  - Provide Stubs / Mocks for external services
- Some Options
  - Classic Unit-Tests with [Mocks](#)
  - Integration Tests with [Testcontainers](#) and [Keycloak-Testcontainers](#)
  - UI / Acceptance Tests with [Selenide](#)

# AuditListener Testing Demo

# AuditListener Testing Implementation Sketch

```
<dependency>
<groupId>com.github.dasniko</groupId>
<artifactId>testcontainers-keycloak</artifactId>
<version>1.6.0</version>
<scope>test</scope>
</dependency>
```

```
public static KeycloakContainer keycloak;
```

```
@BeforeAll
public static void beforeAll() throws Exception {

    keycloak = createKeycloakContainer(REALM_IMPORT_FILE);
    keycloak.withReuse(true);
    keycloak.start();
    keycloak.followOutput(new Slf4jLogConsumer(log));
}
```

```
public static KeycloakContainer createKeycloakContainer() {

    return new KeycloakContainer("quay.io/keycloak/keycloak:12.0.4")
        // import Keycloak realm file
        .withCopyFileToContainer(localPath("target/classes/realm.json"), "...")
        .withEnv("KEYCLOAK_IMPORT", "/tmp/realm.json")
        // adjust Keycloak config on startup
        .withCopyFileToContainer(localPath("cli/init.cli"), "...")
        .withExtensionClassesFrom("target/classes");
}
```

```
@Test
```

```
public void auditListenerShouldPrintLogMessage() throws Exception {
```

```
    ToStringConsumer consumer = new ToStringConsumer();
    keycloak.followOutput(consumer);
```

```
    TokenService tokenService = KeycloakTestSupport.getTokenService(keycloak);
```

```
    // trigger user login via ROPC
```

```
    AccessTokenResponse accessTokenResponse = tokenService.grantToken(ACME_REALM,
        new Form()
            .param("grant_type", "password")
            .param("username", "tester")
            .param("password", TEST_USER_PASSWORD)
            .param("client_id", TEST_CLIENT)
            .param("scope", "openid acme.profile acme.ageinfo")
            .asMap());
```

```
    // Allow the container log to flush
```

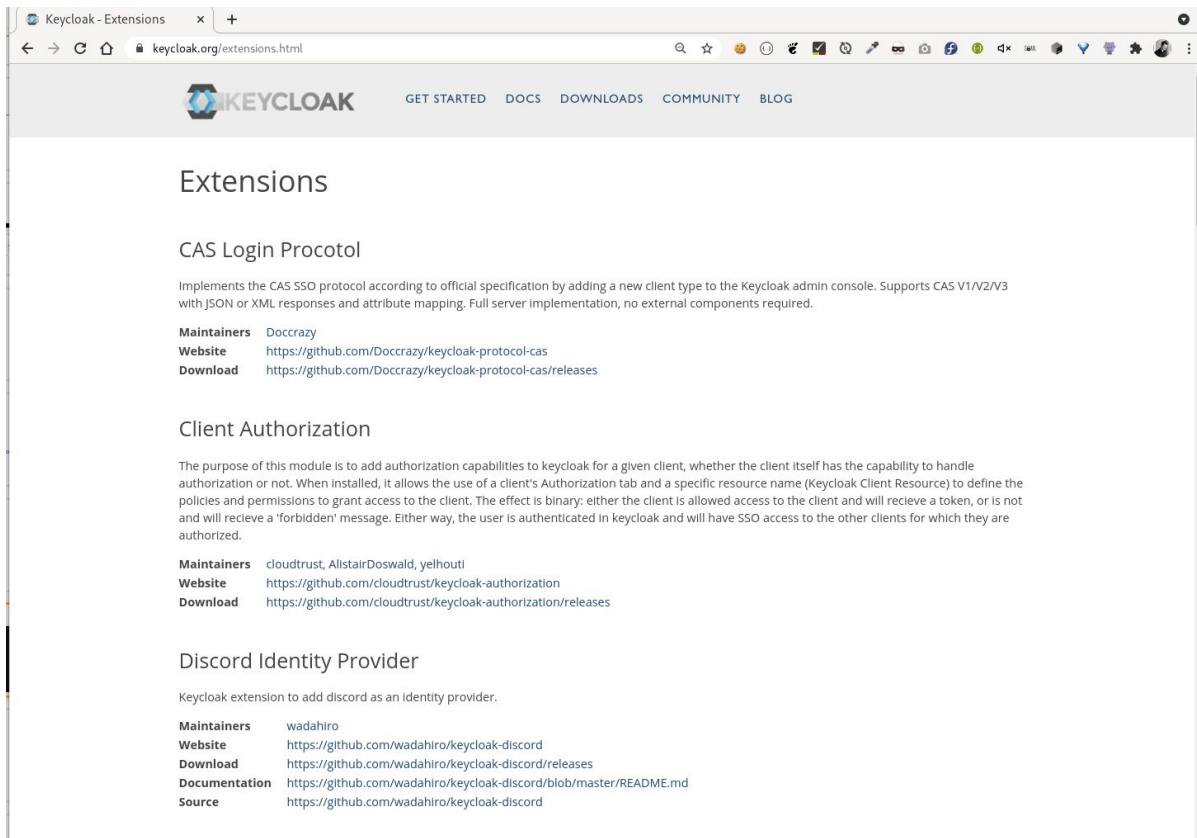
```
    TimeUnit.MILLISECONDS.sleep(750);
```

```
    // Check for audit log entry
```

```
    assertThat(consumer.toUtf8String()).contains("audit userEvent");
```

```
}
```

# Keycloak Community Extensions

A screenshot of a web browser displaying the Keycloak Community Extensions page. The browser's address bar shows 'keycloak.org/extensions.html'. The page has a navigation bar with links: GET STARTED, DOCS, DOWNLOADS, COMMUNITY, and BLOG. The main content area is titled 'Extensions' and lists three community extensions: 'CAS Login Protocol', 'Client Authorization', and 'Discord Identity Provider'. Each extension entry includes a description, maintainers, website, and download links.

Keycloak - Extensions x +

keycloak.org/extensions.html

KEYCLOAK GET STARTED DOCS DOWNLOADS COMMUNITY BLOG

## Extensions

### CAS Login Protocol

Implements the CAS SSO protocol according to official specification by adding a new client type to the Keycloak admin console. Supports CAS V1/V2/V3 with JSON or XML responses and attribute mapping. Full server implementation, no external components required.

**Maintainers** Doccrazy  
**Website** <https://github.com/Doccrazy/keycloak-protocol-cas>  
**Download** <https://github.com/Doccrazy/keycloak-protocol-cas/releases>

### Client Authorization

The purpose of this module is to add authorization capabilities to keycloak for a given client, whether the client itself has the capability to handle authorization or not. When installed, it allows the use of a client's Authorization tab and a specific resource name (Keycloak Client Resource) to define the policies and permissions to grant access to the client. The effect is binary: either the client is allowed access to the client and will receive a token, or is not and will receive a 'forbidden' message. Either way, the user is authenticated in keycloak and will have SSO access to the other clients for which they are authorized.

**Maintainers** cloudtrust, AlistairDowd, yelhouti  
**Website** <https://github.com/cloudtrust/keycloak-authorization>  
**Download** <https://github.com/cloudtrust/keycloak-authorization/releases>

### Discord Identity Provider

Keycloak extension to add discord as an identity provider.

**Maintainers** wadahiro  
**Website** <https://github.com/wadahiro/keycloak-discord>  
**Download** <https://github.com/wadahiro/keycloak-discord/releases>  
**Documentation** <https://github.com/wadahiro/keycloak-discord/blob/master/README.md>  
**Source** <https://github.com/wadahiro/keycloak-discord>

# Keycloak Extension Playground

thomasdarimont / keycloak-extension-playground

Sponsor

Unwatch

Star 191

Fork

<> Code

Issues 7

Pull requests 1

Actions

Projects

Wiki

Settings

master

3 branches

0 tags

thomasdarimont WIP FaceAuth

2be406d 22 hours ago

164 commits

Keycloak Extension Playground

Simple project environment for developing custom Keycloak extensions.

This example uses the in-process `KeycloakServer` from the `keycloak-testsuite-utils` project to ease development.

Note that you might need to build the `keycloak` project locally with the version referenced in this `pom.xml` since the required library `keycloak-testsuite-utils` is not distributed to maven central.

Building Keycloak

To build the Keycloak Project simply clone it into a directory like

```
# The Keycloak version referenced in the pom.xml
KC_VERSION=12.0.4
git clone https://github.com/keycloak/keycloak
git fetch origin --tags
```

Simple project environment for creating custom Keycloak extensions

Apache-2.0 License

Sponsor this project

thomasdarimont Thomas Darimont

Sponsor

Learn more about GitHub Sponsors

Contributors 5

Java 80.8%

FreeMarker 13.3%

HTML 3.6%

JavaScript 1.3%

CSS 0.9%

Shell 0.1%



# **Development Setups**

# Example Development Setups

---

- Keycloak Server as Standalone Installation or Docker Container
- Single Maven Project deployed to “Stock” Keycloak Server
  - Single .jar deployment
- Multi-Module Maven Project deployed to “Stock” Keycloak Server
  - Single/Multi .ear/.jar deployment
- Multi-Module Maven Project and custom built Keycloak Server
  - Single/Multi .ear/.jar deployment
  - Custom Keycloak Distribution (extended “Stock”)

# Demo: Multi-Module Maven Project with Docker

The screenshot displays an IDE window for a project named `keycloak-extensions-talk`. The project structure on the left shows a multi-module Maven project with the following hierarchy:

- `keycloak-extensions-talk`
  - `.history`
  - `.idea`
  - `acme-dist`
  - `acme-extensions`** (selected)
  - `acme-themes`
  - `cli`
  - `demoapp`
  - `etc`
  - `imex`
  - `requests`
  - `scratch`
  - `testrun`
  - `.gitignore`
  - `docker-compose.yml`
  - `keycloak-extensions-talk.iml`
  - `notes.md`
  - `pom.xml`
  - `readme.md`
- External Libraries
- Scratches and Consoles

The main editor shows the `AgeInfoMapper.java` file in the `acme-extensions` module. The code defines an `AgeInfoMapper` class that implements `OIDCAccessTokenMapper`, `OIDCIDTokenMapper`, and `UserInfoTokenMapper`. It includes imports for `org.keycloak.representations.IDToken`, `java.time.LocalDate`, `java.time.LocalDateTime`, `java.time.format.DateTimeParseException`, `java.time.temporal.ChronoUnit`, `java.util.ArrayList`, `java.util.List`, and `java.util.Optional`.

The bottom panel shows the `Services` tab, displaying a log of the application's startup. The log indicates that the `keycloak-server.war` is deployed and the `keycloak-extensions-talk` application is running. The log also shows the `keycloak-extensions-talk` application's configuration, including the `keycloak-extensions-talk` module and the `keycloak-extensions-talk` module's dependencies.

The status bar at the bottom shows the IDE's configuration: `75:6 LF UTF-8 4 spaces`, the current branch is `master`, and the file encoding is `UTF-8`. The IDE is running on a system with 21916f5120M of memory.



# Demo: Keycloak Extension Playground

- Idea: Provide a simple sandbox to develop new extensions quickly
- [keycloak-extension-playground](#) on Github

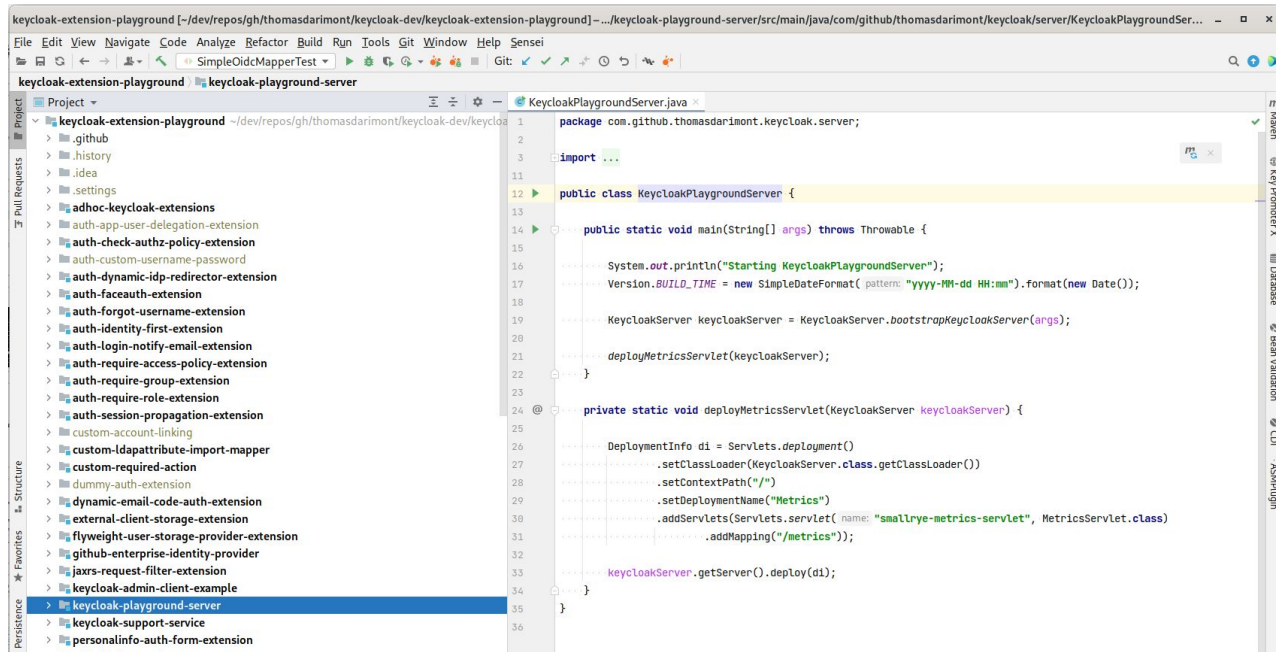
```
<artifactId>keycloak-playground-server</artifactId>

<dependencies>
  <dependency>
    <groupId>${project.groupId}</groupId>
    <artifactId>auth-faceauth-extension</artifactId>
    <version>${project.version}</version>
  </dependency>

  <dependency>
    <groupId>${project.groupId}</groupId>
    <artifactId>saml-group-ldap-mapper</artifactId>
    <version>${project.version}</version>
  </dependency>

  <dependency>
    <groupId>${project.groupId}</groupId>
    <artifactId>custom-ldapattribute-import-mapper</artifactId>
    <version>${project.version}</version>
  </dependency>

  <dependency>
    <groupId>org.keycloak</groupId>
    <artifactId>keycloak-testsuite-utils</artifactId>
    <version>${keycloak.version}</version>
  </dependency>
```



# Demo: Spring Boot Embedded Keycloak

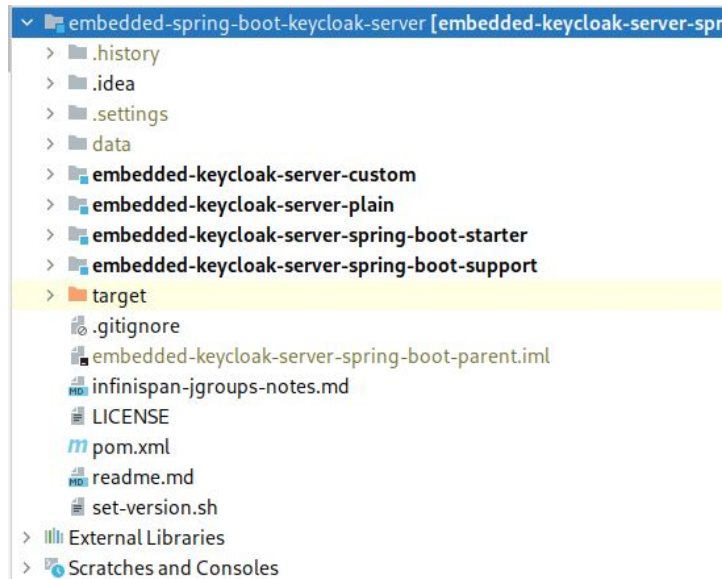
- [embedded-spring-boot-keycloak-server](#) on Github
- Embedded Keycloak Server as consumable spring-boot starter
- [Example as Gist](#)

```
<dependency>
  <groupId>com.github.thomasdarimont.keycloak</groupId>
  <artifactId>embedded-keycloak-server-spring-boot-starter</artifactId>
  <version>3.0.0-SNAPSHOT</version>
</dependency>
```

```
<repositories>
  <repository>
    <id>jitpack.io</id>
    <url>https://jitpack.io</url>
  </repository>
</repositories>
```

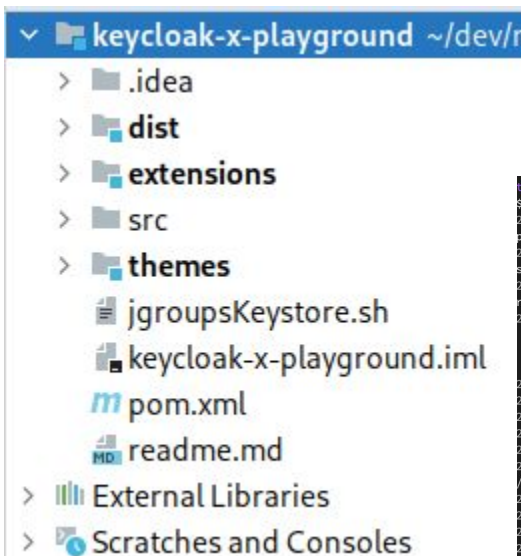


```
Run: Main
2021-04-07 22:56:49.312 INFO 2048597 --- [main] atKeycloakPropertiesApplicationListener
file:/home/tom/dev/repos/gh/thomasdarimont/keycloak-dev/embedded-spring-boot-keycloak-server/embedded-spring-boot-keycloak-server-3.0.0-SNAPSHOT.jar
Spring Boot Version: (v2.4.4)
Keycloak Version: (v@keycloak.version@)
```

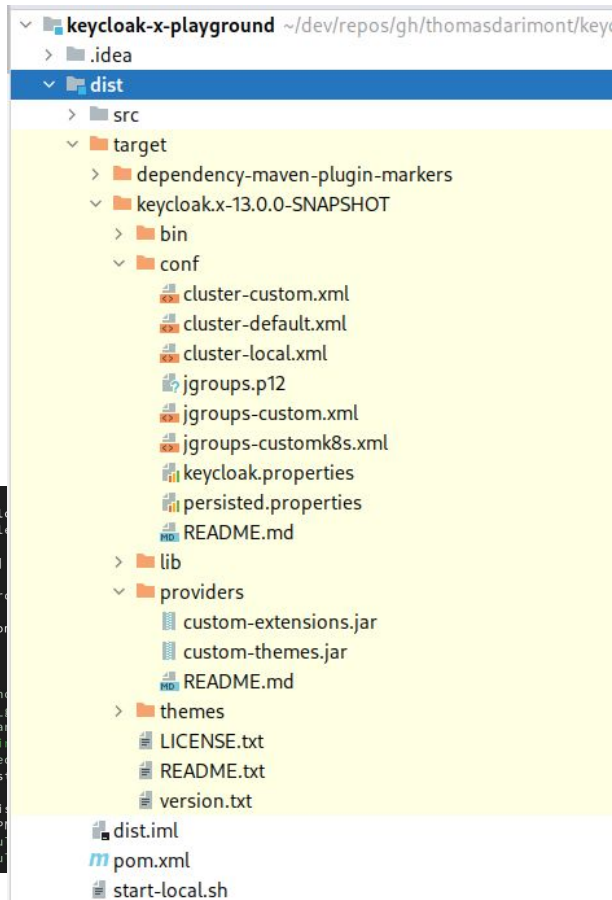


# Demo: Multi-Module Project with Keycloak-X

- Custom Keycloak Distribution based on Keycloak-X
- Extensions and Themes deployed as .jar's



```
com@neumann ~/dev/repos/gh/thomasdarimont/keycloak-dev/keycloak-x-playground/dist
$ ./start-local.sh start --verbose --http-port=8080 --http-enabled=true --Dkc.db.password=keycloak
2021-04-07 22:58:12,330 WARN [org.key.qua.KeycloakRecorder] (main) New property [kc.http.enabled] persisted into the server image.
2021-04-07 22:58:12,343 WARN [org.key.qua.KeycloakRecorder] (main) New property [kc.http-port] persisted into the server image.
2021-04-07 22:58:12,368 WARN [org.key.qua.KeycloakRecorder] (main) New property [kc.db.password] persisted into the server image.
2021-04-07 22:58:12,378 WARN [org.key.qua.KeycloakRecorder] (main) Please, run the 'config' command.
kc.sh config --http-enabled=true --http-port=8080 --db-password=keycloak
2021-04-07 22:58:13,922 WARN [io.qua.run.ConfigChangeRecorder] (main) Build time property cannot be changed.
2021-04-07 22:58:14,203 INFO [org.key.com.Profile] (main) Preview feature enabled: admin_fine_grained_permissions
2021-04-07 22:58:14,204 INFO [org.key.com.Profile] (main) Preview feature enabled: token_exchange
2021-04-07 22:58:14,810 INFO [com.git.tho.key.ext.eve.CustomEventListenerProviderFactory] (main) Loading cluster configuration
2021-04-07 22:58:14,811 INFO [org.key.url.DefaultHostnameProviderFactory] (main) Frontend: <redacted>
2021-04-07 22:58:14,896 INFO [org.key.pro.qua.QuarkusCacheManagerProvider] (main) Loading cluster configuration
2021-04-07 22:58:15,329 INFO [org.inf.CONTAINER] (main) ISPN000128: Infinispan version: Infinispan-12.1.0
2021-04-07 22:58:15,442 INFO [org.inf.CLUSTER] (main) ISPN000078: Starting JGroups channel ISPN000078
2021-04-07 22:58:15,714 WARN [org.jgr.pro.UDP] (main) JGRP000015: the send buffer of socket Mu
2021-04-07 22:58:15,715 WARN [org.jgr.pro.UDP] (main) JGRP000015: the send buffer of socket Mu
```





# KEYCLOAK Summary

---

- Easy to get started
  - unzip & run, [Keycloak Docker Images](#)
- Provides many features out of the box
  - SSO, Social Login, Federation, User Management,...
- Very extensible and easy to integrate
  - Many extension points & customization options
- Multiple Developer Setups possible
  - Tailor your workflow

# Thanks!




@thomasdarimont

# Links

---

- [Keycloak Website](#)
- [Keycloak Docs](#)
- [Keycloak Blog](#)
- [Keycloak User Mailing List](#)
- [Keycloak Developer Mailing List](#)
- [OpenID Connect](#)
- [Keycloak Community Extensions](#)
- [SAML](#)
- [JSON Web Tokens](#)
- [Awesome Keycloak](#)
- [Keycloak Quickstart Projects](#)
- [Keycloak Extension Playground](#)
- [Spring Boot Keycloak Server](#)

# Keycloak Extension Example

 dteleguin / beercloak

Watch 7

Unstar 39

Fork 9

<> Code

Issues 0

Pull requests 0

Projects 0

Wiki

Insights

BeerCloak: a comprehensive Keycloak extension example

keycloak

23 commits

1 branch

0 releases

0 contributors

Branch: master


New pull request

Create new file

Upload files

Find file

Clone or download

 **Dmitry Telegin** Update theme for latest Keycloak

Latest commit 4d7f14a on Jan 9

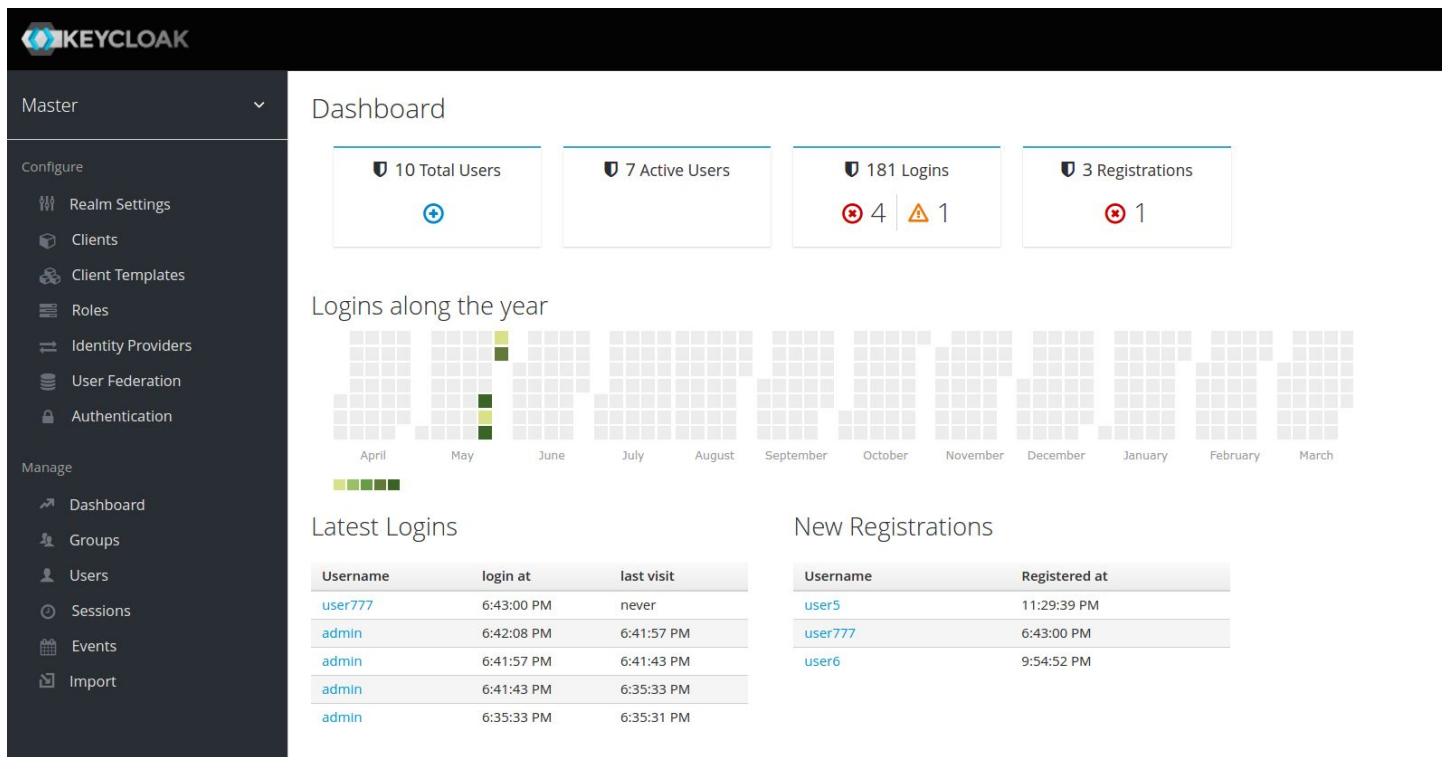
**Total lines in the current directory: 92**  
**By extensions:**  
gitignore - 2,  
md - 70,  
xml - 20

beercloak-core	Split some "business logic" into a separate module	a year ago
beercloak-ear	Split some "business logic" into a separate module	a year ago
beercloak-module	Update theme for latest Keycloak	2 months ago
.gitignore 2 lines	Split some "business logic" into a separate module	a year ago
README.md 70 lines	Update README.md	a year ago
pom.xml 20 lines	Split some "business logic" into a separate module	a year ago

README.md

BeerCloak: a comprehensive Keycloak extension example

# Custom Dashboard Extension



Please vote :) <https://issues.jboss.org/browse/KEYCLOAK-1840>