Implementing the ExternalizableSecurityCheck

Overview

The abstract `ExternalizableSecurityCheck` class implements the `SecurityCheck` interface and handles two important aspects of the security check functionality: externalization and state management.

- Externalization - this class implements the `Externalizable` interface, so that the derived classes don't need to implement it themselves.
- State management - this class predefined a `STATE_EXPIRED` state that means the security check is expired and its state will not be preserved. The derived classes need to define other states supported by their security check.

Three methods are required to be implemented by the subclasses: `initStateDurations`, `authorize` and `introspect`.

This tutorial explains how to implement the class and demonstrates how to manage states.

Prerequisites: Make sure to read the Authorization concepts (../authorization-concepts/) and Creating a Security Check (../creating-a-security-check) tutorials.

Jump to:

- The initStateDurations Method
- The authorize Method
- The introspect Method

The initStateDurations Method

The `ExternalizableSecurityCheck` defines an abstract method called `initStateDurations`. The subclasses must implement that method providing the names and durations for all states supported by their security check. The duration values usually come from the security check configuration.

```
private static final String SUCCESS_STATE = "success";

protected void initStateDurations(Map<String, Integer> durations) {
    durations.put (SUCCESS_STATE, ((SecurityCheckConfig) config).successStateExpirationSec);
}
```

> For more information about security check configuration, see the configuration class section (../credentials-validation/security-check/#configuration-class) in the Implementing the CredentialsValidationSecurityCheck tutorial.

The authorize Method

The `SecurityCheck` interface defines a method called `authorize`. This method is responsible for implementing the main logic of the security check, managing states and sending a response to the client (success, challenge, or failure).

Use the following helper methods to manage states:

```
protected void setState(String name)
```

```
public String getState()
```

In the following example we simply check if the user is logged-in and return success or failure respectively:

```
public void authorize(Set<String> scope, Map<String, Object> credentials, HttpServletRequest request,
AuthorizationResponse response) {
    if (loggedIn){
        setState(SUCCESS_STATE);
        response.addSuccess(scope, getExpiresAt(), this.getName());
    } else {
        setState(STATE_EXPIRED);
        Map <String, Object> failure = new HashMap<String, Object>();
        failure.put("failure", "User is not logged-in");
        response.addFailure(getName(), failure);
    }
}
```

The `AuthorizationResponse.addSuccess` method adds the success scope and its expiration to the response object. It requires:

- The scope granted by the security check.
- The expiration of the granted scope.
  The `getExpiresAt` helper method returns the time when the current state will be expired, or 0 if the current state is null:

  ```
  public long getExpiresAt()
  ```

- The name of the security check.

The `AuthorizationResponse.addFailure` method adds a failure to the response object. It requires:

- The name of the security check.
- A failure `Map` object.

The `AuthorizationResponse.addChallenge` method adds a challenge to the response object. It requires:

- The name of the security check.
- A challenge `Map` object.

The introspect Method

The `SecurityCheck` interface defines a method called `introspect`. This method should make sure that the security check is in the state that grants the requested scope. If the scope is granted, the security check should report the granted scope, its expiration, and a custom introspection data to the result parameter. If the scope is not granted, the security check does noting.
This method may change the state of the security check and/or the client registration record.

```
public void introspect(Set<String> checkScope, IntrospectionResponse response) {
   if (getState().equals(SUCCESS_STATE)) {
      response.addIntrospectionData(getName(),checkScope,getExpiresAt(),null);
   }
}
```