# Implementing the User Authentication Security Check



## Overview

This abstract class extends `CredentialsValidationSecurityCheck` and builds upon it to fit the most common use-cases of simple user authentication. In addition to validating the credentials, it creates a user identity that will be accessible from various parts of the framework, allowing you to identity the current user.

Optionally, `UserAuthenticationSecurityCheck` also provides **Remember Me** capabilities.

This tutorial uses the example of Security Check asking for a username and password and uses the username to represent an authenticated user.

**Prerequisites:** Make sure to read the Credentials Validation Security Check (../../credentials-validation/) tutorial.

## UserAuthSecurityCheck

Create a Java adapter and add a Java class named `UserAuthSecurityCheck` that extends `UserAuthenticationSecurityCheck`.

```
public class UserAuthSecurityCheck extends UserAuthenticationSecurityCheck {

    @Override
    protected AuthenticatedUser createUser() {
        return null;
    }

    @Override
    protected boolean validateCredentials(Map<String, Object> credentials) {
        return false;
    }

    @Override
    protected Map<String, Object> createChallenge() {
        return null;
    }
}
```

## Creating the challenge

The challenge is exactly the same as the one described in Implementing the Credentials Validation Security Check (../../credentials-validation/security-check/).

```
@Override
protected Map<String, Object> createChallenge() {
    HashMap challenge = new HashMap();
    challenge.put("errorMsg",errorMsg);
    challenge.put("remainingAttempts",remainingAttempts);
    return challenge;
}
```