Basic Authentication

This section provides details on how Spring Security provides support for Basic HTTP Authentication for servlet-based applications.

This section describes how HTTP Basic Authentication works within Spring Security. First, we see the WWW-Authenticate header is sent back to an unauthenticated client:

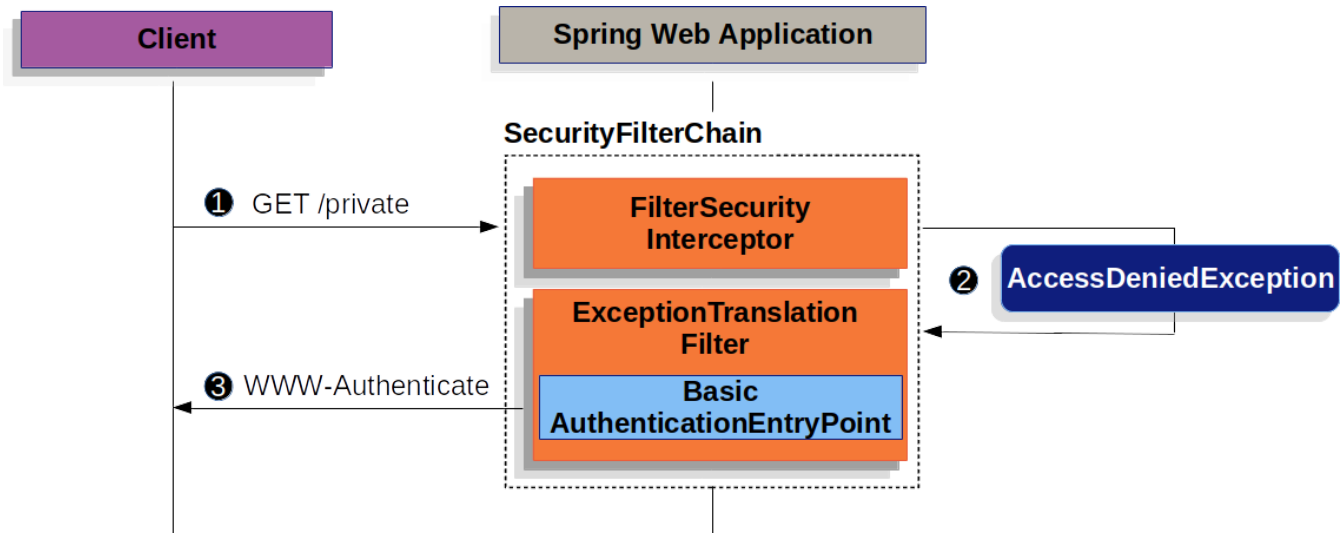

Figure 1. Sending WWW-Authenticate Header

The preceding figure builds off our `SecurityFilterChain` diagram.

1 First, a user makes an unauthenticated request to the resource `/private` for which it is not authorized.

2 Spring Security's `AuthorizationFilter` indicates that the unauthenticated request is *Denied* by throwing an `AccessDeniedException`.

3 Since the user is not authenticated, `ExceptionTranslationFilter` initiates *Start Authentication*. The configured `AuthenticationEntryPoint` is an instance of `BasicAuthenticationEntryPoint`, which sends a WWW-Authenticate header. The `RequestCache` is typically a `NullRequestCache` that does not save the request since the client is capable of replaying the requests it originally requested.

When a client receives the `WWW-Authenticate` header, it knows it should retry with a username and password. The following image shows the flow for the username and password being processed:

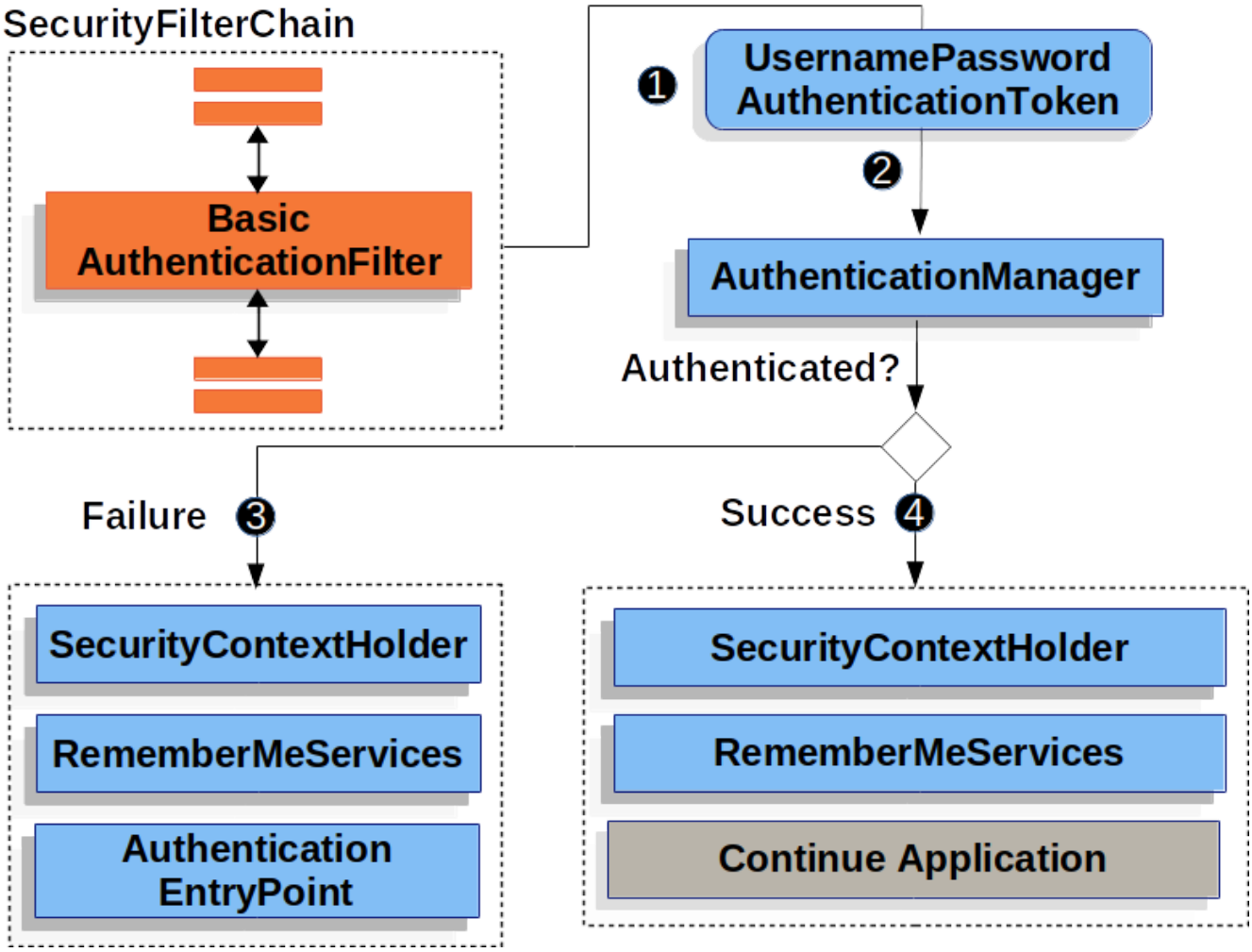

Figure 2. Authenticating Username and Password

The preceding figure builds off our `SecurityFilterChain` diagram.

1 When the user submits their username and password, the `BasicAuthenticationFilter` creates a `UsernamePasswordAuthenticationToken` , which is a type of `Authentication` by extracting the username and password from the `HttpServletRequest` .

2 Next, the `UsernamePasswordAuthenticationToken` is passed into the `AuthenticationManager` to be authenticated. The details of what `AuthenticationManager` looks like depend on how the user information is stored.

3 If authentication fails, then *Failure*.

1. The SecurityContextHolder is cleared out.
2. `RememberMeServices.loginFail` is invoked. If remember me is not configured, this is a no-op. See the `RememberMeServices` interface in the Javadoc.
3. `AuthenticationEntryPoint` is invoked to trigger the WWW-Authenticate to be sent again. See the `AuthenticationEntryPoint` interface in the Javadoc.

4 If authentication is successful, then *Success*.

1. The Authentication is set on the SecurityContextHolder.
2. `RememberMeServices.loginSuccess` is invoked. If remember me is not configured, this is a no-op. See the `RememberMeServices` interface in the Javadoc.
3. The `BasicAuthenticationFilter` invokes `FilterChain.doFilter(request,response)` to continue with the rest of the application logic. See the `BasicAuthenticationFilter` Class in the Javadoc

By default, Spring Security's HTTP Basic Authentication support is enabled. However, as soon as any servlet based configuration is provided, HTTP Basic must be explicitly provided.

The following example shows a minimal, explicit configuration:

Explicit HTTP Basic Configuration

Java | **XML** | **Kotlin**

JAVA

```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) {
    http
        // ...
        .httpBasic(withDefaults());
    return http.build();
}
```