Pertemuan 10

Spring Security

Keamanan Dulu vs Sekarang

- Jaman dahulu orang-orang tidak perlu mengunci pintu/jendela mereka karena tidak ada barang berharga yang bisa dicuri
- Jaman sekarang banyak orang mempunyai barang berharga, sehingga mereka mulai menggunakan kunci rumah

Lanjutan

- Dalam dunia digital, barang yang sangat berharga bagi tiap-tiap orang adalah informasi
- Informasi **dapat** diperjualbelikan, sehingga seremeh apapun informasi tersebut orang bisa menggunakannya
- Informasi yang paling sensitif adalah informasi pribadi

Spring dan Keamanan

- Dalam bidang keamanan, Spring juga memerlukan pengamanan.
- Aplikasi yang dibuat dengan menggunakan Spring pastinya memiliki informasi yang tersimpan di dalamnya
- Oleh karena itu Spring menyediakan fitur Spring Security untuk pengamanan

Lanjutan

- Spring Security cukup mudah ditanamkan ke dalam aplikasi yang ingin dibuat. Namun harus dilakukan saat pembuatan projek dilakukan menggunakan dependency

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Spring Security

- Ketika dijalankan, Konfigurasi Otomatis akan mendeteksi Spring Security dan melakukan pembuatan keamanan dasar
- Hal ini bisa diuji dengan menjalankan aplikasi dan membuka websitenya, secara otomatis web akan meminta user dan password yang sudah dibuat otomatis di bagian log

Lanjutan

- Ketika menambahkan dependensi keamanan maka kita akan mendapatkan fitur berupa:
  - Semua permintaan HTTP memerlukan otentifikasi
  - Tidak ada aktor spesifik atau otoritas yang diperlukan
  - Tidak ada halaman login
  - Hanya ada satu pengguna yaitu user

Konfigurasi Tambahan

- Hal yang bisa dilakukan setelah konfigurasi dasar adalah
- Mengganti otentifikasi dengan halaman login
- Menambahkan pengguna melalui halaman pendaftaran
- Menambahkan peraturan keamanan untuk permintaan yang berbeda

Konfigurasi User

- Dalam konfigurasi keamanan user pastinya memerlukan tempat penyimpanan di mana informasi user disimpan.
- Ada beberapa cara untuk menyimpan informasi user secara aman.
- Spring menyediakan pengguna khusus untuk pengujian sistem

Jenis Penyimpanan User

- In-memory user store
- JDBC-based user store
- LDAP-backed user store
- A custom user details service

In-memory user store

- Penyimpanan ini hanya cocok untuk pengujian saja atau untuk aplikasi simpel
- Untuk melakukan pengubahan, penambahan user harus dengan cara mengkode ulang aplikasi

Contoh Kode

- .inMemoryAuthentication()
- .withUser("buzz")
- .password("infinity")
- .authorities("ROLE_USER")
- .and()
- .withUser("woody")
- .password("bullseye")
- .authorities("ROLE_USER");

JDBC-based user store

- Informasi pengguna sering dijaga di sebuah database dan sebuah penyimpanan berbasis JDBC dianggap sangat tepat.
- Dalam pengkodeannya terdapat elemen-elemen untuk memanggil data-data tertentu layaknya halaman login pada biasanya

Contoh Kode

- public static final String DEF_USERS_BY_USERNAME_QUERY =

  "select username,password,enabled " +

  "from users " + "where username = ?";
- public static final String DEF_AUTHORITIES_BY_USERNAME_QUERY =

  "select username,authority " +

  "from authorities " + "where username = ?";

LDAP-backed user store

- Sebuah buku telepon (halaman putih) yang berisi kontak informasi.
- LDAP menyediakan layanan ini untuk berbagai kebutuhan
- Server LDAP tidak terletak di dalam Spring ataupun database yang digunakannya
- Melainkan terletak diinternet

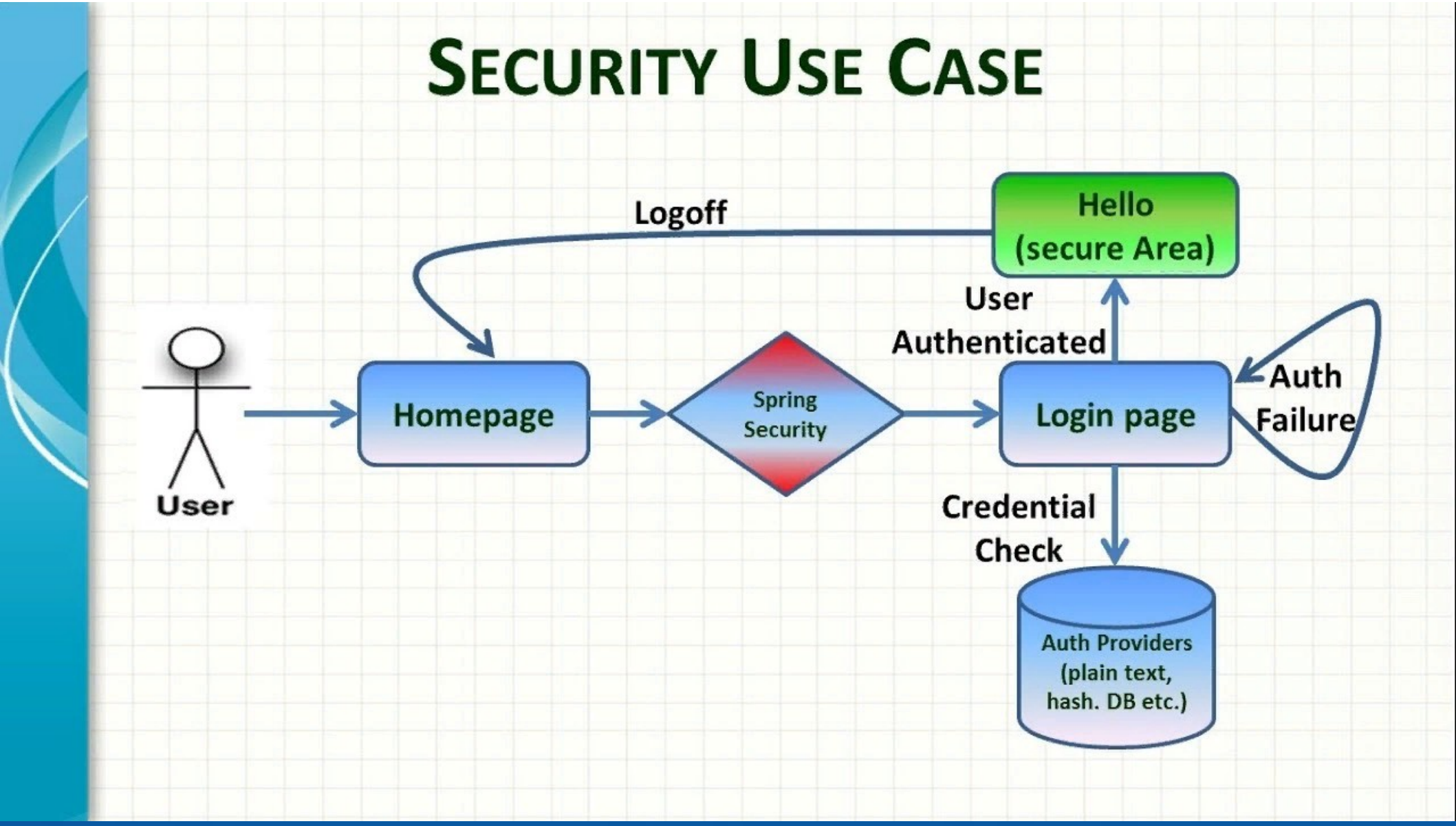
Customizing user authentication

- Cara ini dengan membuat konfigurasi tersendiri bagaimana penyimpanan user dilakukan oleh programmer
- Dalam pembuatannya bisa menggunakan Database yang telah disiapkan oleh programmer sebelumnya

Mengamankan Permintaan

- Setiap kita browsing web yang dibangun menggunakan Spring, maka setiap yang kita lakukan dianggap sebagai request
- Tidak semua request/browsing ke semua halaman itu bisa diakses seenaknya
- Beberapa halaman memerlukan otentifikasi untuk masuk ke halaman tersebut

Ilustrasi


SECURITY USE CASE
Logoff
Hello
(secure Area)
User
Authenticated
Homepage
Spring
Security
Login page
Auth
Failure
User
Credential
Check
Auth Providers
(plain text,
hash. DB etc.)

Konfigurasi Request

- access(String) = Allows access if the given SpEL expression evaluates to true
- Anonymous() = Allows access to anonymous users
- Authenticated() = Allows access to authenticated users
- DenyAll() = Denies access unconditionally
- PermitAll() = Allows access unconditionally

Kesimpulan

- Spring Security konfigurasi otomatis merupakan sebuah cara hebat untuk memulai dengan keamanan, namun tetap harus ada konfigurasi lanjutnya
- Pengguna bisa diatur menggunakan relational database, LDAP, ataupun buatan programmer sendiri
- Spring Security secara otoamtis melindungi dari senarangan CSRF

Thank you