Projekt WASD

Projektinformationen

Spring

Zur Entwicklung dieser Web-Applikation wird das Open Source Java Framework Spring angewendet. Damit dieses Framework jedoch einfacher in der Verwendung ist und der Aufwand nicht aus den Fugen gerät, kommt ebenfalls Spring Boot zum Einsatz. Spring Boot ist eine vereinfachte Schnittstelle zwischen Spring Framework und Benutzer, damit man als Entwickler im Projekt schneller und vor allem einfacher durchstarten kann.

PostgreSQL

Für den Aufbau der Datenbank wird PostgreSQL verwendet, da es weit verbreitet als die am meisten fortgeschrittene relationale Open Source Datenbank bekannt ist. Genau genommen ist es ein objektrelationales Datenbankmanagementsystem. PostgreSQL richtet sich nach dem SQL-Standard und wird zur Verwendung im WASD-Projekt auf dem Server installiert.

Login

Das Login ist per HTTP-Post gelöst. Über die Spring-Security-Config wird sichergestellt, dass der Benutzer für die abgesicherten Bereiche authentifiziert sein muss. Ist man bei der Applikation in diesem Projekt nicht angemeldet, hat man nur Zugriff auf die Login-Maske. Versucht man auf eine andere Seite zuzugreifen, wird man von einem Filter auf die Login-Maske weitergeleitet.

Einloggen kann man sich über die Login-Maske mit den folgenden Benutzerdaten:

Benutzername: lb3
Passwort: gibbiX12345

Session-Handling

Sessions können durch das Spring-Framework verwaltet werden. Wenn sich ein Benutzer anmeldet, wird gleich eine neue Session ID erstellt und in ein Cookie gespeichert. In diesem Projekt wird eine sogenannte JSESSIONID erstellt.

Systemkommando

Systemkommandos lassen sich über die im Projekt integrierte ProcessBuilder Library ausführen. Damit jedoch keine x-beliebigen User Systemkommandos über das Interface der Web-Applikation absetzen können, müssen präventive Vorkehrungen getroffen werden. Dafür werden alle Parameter vor Übergabe an den ProcessBuilder durch eine Escaping Library in harmlosen Text übersetzt. Ausserdem schützt sich der ProcessBuilder schon von Haus gegen ungewollte Zugriffe auf die Betriebssystemseite, indem er Kommando von Argumenten trennt und ein Übergriff via Parametereingabe dadurch verhindert werden kann. Die gewünschten Systemkommandos, die über das Interface der Web-Applikation eingegeben werden, werden über HTTP-GET abgesetzt.

Log

Auch beim Logging wird in die Tasche des Spring-Frameworks gegriffen, welches durch vordefinierte Funktionen ein einfaches Logging ermöglicht. Im Beispiel des WASD-Projekts, werden relevante Informationen anhand dieser Funktionen in die Log-Datei *wasd.log* geschrieben.

Damit das Ganze auch richtig abgespeichert wird, wurde folgendes in der *application.properties*-Datei definiert:

```
logging.file=wasd.log
```

Nachfolgend noch einige Log-Meldungen, wie sie in der wasd.log-Datei abgespeichert werden:

```
2018-06-13 14:33:54.063  INFO 30921 --- [localhost-startStop-1] .s.DelegatingFilterProxyRegistrationBean : Mapping filter: 'springSecurityFilterChain' to: [/*]
2018-06-13 14:33:54.063  INFO 30921 --- [localhost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean  : Servlet dispatcherServlet mapped to [/]
2018-06-13 14:33:54.241  INFO 30921 --- [restartedMain] com.zaxxer.hikari.HikariDataSource       : wasd4 - Starting...
2018-06-13 14:33:54.308  INFO 30921 --- [restartedMain] com.zaxxer.hikari.HikariDataSource       : wasd4 - Start completed.
```

Passwortpersistenz

Die Passwortsicherheit wird wie beim Session-Handling und dem Logging ebenfalls durch das Spring-Framework gewährleistet, da dieses eine Standardfunktion zum „salzen“ von Passwörtern bereitstellt. Der Benutzer (und somit auch das Passwort) wird in diesem Projekt persistent in einer PostgreSQL-Datenbank abgespeichert.

Der Standardbenutzer, welcher für das WASD-Projekt erstellt werden musste, wurde folgendermassen generiert.

Es wurde ein SQL-Script erstellt, welches entweder manuell durchgeführt werden kann oder dann von Flyway übernommen wird. Anzumerken ist, dass die folgende Postgres-Extension verwendet wird, für das „salzen“ und somit absichern vor Wörterbuch-Angriffen.

```
CREATE EXTENSION IF NOT EXISTS pgcrypto;
```

Angewendet wird das dann beim INSERT Statement vom „lb3“-Benutzer mit dem Passwort „gibbiX12345“ folgendermassen:

```
crypt('gibbiX12345', gen_salt('bf'))
```

Netzwerk-Kommunikation

Das Einbinden von SSL-Zertifikaten wird in der Konfigurationsdatei *application.properties* konfiguriert. Dies ermöglicht uns das Spring-Framework. Die Endpunkte aller Webservices, welche das Framework erstellt, sind somit abgesichert.

In der *application.properties*-Datei sieht das Ganze folgendermassen aus.

```
server.port=8443
security.require-ssl=true
server.ssl.key-store-type=PKCS12
server.ssl.key-store=classpath:keystore.p12
server.ssl.key-store-password=*pw*
server.ssl.key-alias=tomcat
```