НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерных технологий

Сервисно-ориентированная архитектура

Лабораторная работа № 2

Вариант 1006.69

Выполнил
студент

Демичев Даниил Дмитриевич

Группа № P34112

Преподаватель: Усков Иван Владимирович

г. Санкт-Петербург

2021

Задание

Доработать веб-сервис и клиентское приложение из лабораторной работы #1 следующим образом:

- Отрефакторить сервис из лабораторной работы #1, переписав его на фреймворке JAX-RS с сохранением функциональности и API.
- **Набор функций, реализуемых сервисом, изменяться не должен!**
- Развернуть переработанный сервис на сервере приложений Payara.
- Разработать новый сервис, вызывающий API существующего.
- Новый сервис должен быть разработан на базе Spring MVC REST и развёрнут на сервере приложений WildFly.
- Разработать клиентское приложение, позволяющее протестировать API нового сервиса.
- Доступ к обоим сервисам должен быть реализован с по протоколу https с самоподписанным сертификатом сервера. Доступ к сервисам посредством http без шифрования должен быть запрещён.

Новый сервис должен располагаться на URL /heroes и реализовывать следующие операции:

- `/team/{team-id}/remove-without-toothpick` : удалить из команды всех героев без зубочисток
- `/team/{team-id}/make-depressive` : поменять всем героям команды настроение на максимально печальное

Исходный код

https://github.com/xxPFFxx/SOA-lab2

Настройка сертификатов

Основной сервис

keytool -genkey -alias payara -keyalg RSA -keystore soastore -validity 999 -keysize 2048
keytool -export -alias payara -keyalg RSA -keystore soastore -file payaratrust.crt
keytool -import -alias payara -keyalg RSA -keystore payaratospringtruststore.jks -file payaratrust.crt

Новый сервис

keytool -genkey -alias soaspring -keyalg RSA -keystore soaspringstore -validity 999 -keysize 2048

keytool -export -alias soaspring -keyalg RSA -keystore soaspringstore -file soaspringtrust.crt
keytool -import -alias soaspring -keyalg RSA -keystore payaratruststore.jks -file soaspringtrust.crt

Конфигурация серверов

Payara Micro

Чтобы запретить обмен данными через http в JAVA EE можно добавить в web.xml Security Constraint:

```
<security-constraint>
    <web-resource-collection>
        <web-resource-name>My Secure Stuff</web-resource-name>
        <url-pattern>/*</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>CONFIDENTIAL</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

Запуск сервера производится следующей командой, в которой указываются пути и пароли к keyStore и trustStore:

```
java -Djavax.net.ssl.keyStore=".\SecuritySertificates\soastore" -
Djavax.net.ssl.keyStorePassword="soasoa" -
Djavax.net.ssl.trustStore=".\SecuritySertificates\payaratruststore.jks" -
Djavax.net.ssl.trustStorePassword="soasoa" -jar payara-micro.jar --deploy
C:/Users/Daniil/IdeaProjects/SOA-lab2/MainService/target/MainService --contextroot / --
sslPort 51510 --autoBindSsl --sslCert payara
```

WildFly

Создание Security Realm с указанием keyStore и trustStore, указание его в https-листенере:

```
        <security-realm name="SoaRealm">
                <server-identities>
                        <ssl>
                                <keystore path="soaspringstore" relative-
to="jboss.server.config.dir" keystore-password="soasoa"/>
                        </ssl>
                </server-identities>
                <authentication>
                        <truststore path="payaratospringtruststore.jks" relative-
to="jboss.server.config.dir" keystorepassword="soasoa" />
                </authentication>
        </security-realm>
```

<https-listener name="default" socket-binding="https" security-realm="SoaRealm" enable-http2="true"/>

В WildFly для запрета передачи по http нужно удалить http-listener из standalone.xml

Для того, чтобы второй сервис мог обращаться к первому через RestTemplate, необходимо настроить его, указав SSLContext и trustStore:

```
   public RestTemplate restTemplate() throws CertificateException,
NoSuchAlgorithmException, KeyStoreException, IOException, KeyManagementException {
      System.out.println(dirToSertificates + " " + customTrustStore + " " +
customTrustStorePassword);
      SSLContext sslContext = SSLContextBuilder.create()
            .loadTrustMaterial(new File(dirToSertificates + customTrustStore),
                  customTrustStorePassword.toCharArray())
            .build();
      CloseableHttpClient httpClient = HttpClients.custom()
            .setSSLContext(sslContext)
            .setSSLHostnameVerifier(NoopHostnameVerifier.INSTANCE)
            .build();
      HttpComponentsClientHttpRequestFactory customRequestFactory = new
HttpComponentsClientHttpRequestFactory();
      customRequestFactory.setHttpClient(httpClient);
      return new RestTemplate(customRequestFactory);
```

Вывод

В ходе выполнения этой лабораторной работы я разработал два сервиса и настроил их защищенное взаимодействие с помощью ssl и https. Для этого у каждого сервиса должен храниться trustStore с доверенными сертификатами и keyStore с собственными приватными ключами.