Университет ИТМО

Факультет программной инженерии и компьютерной техники

Сервис-ориентированная архитектура

Лабораторная работа №2

Вариант 3007.6

Группа: P34122

Студент: Данков Антон Игоревич

Преподаватель: Усков Иван Владимирович

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 /booking и реализовывать следующие операции:

- /sell/vip/ticket-id/person-id : скопировать указанный билет, создав такой же, но с категорией "VIP" и с удвоенной ценой
- /event/{event-id}/cancel : отменить указанное событие, удалив все билеты на него

Оба веб-сервиса и клиентское приложение должны быть развёрнуты на сервере helios.

Код

https://github.com/Antondstd/Service-oriented-architecture/tree/master/Lab2

Создание сертификатов

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

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

Настройка серверов

Payara Micro

java -Djavax.net.ssl.keyStore="..\Lab2\Security certificates\soastore" -Djavax.net.ssl.keyStorePassword="soasoa" -Djavax.net.ssl.trustStore="..\Lab2\Security certificates\payaratruststore.jks" -Djavax.net.ssl.trustStorePassword="soasoa" -jar payara-micro.jar --deploy D:/University/7_semester/SOA/wildfly2/standalone/deployments/ROOT.war --contextroot / --sslPort 8181 --autoBindSsl --sslCert payara

Для запрета http в web.xml

```
<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>
```

WildFly

Создание Security Real в standalone.xml

```
<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" keystore-
password="soasoa" />
             </authentication>
          </security-realm>

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

Для запрета http удаляется http-listener

Также чтобы второй сервер через RestTemplate мог отправлять запросы, надо в него добавить SslContext в котором указать путь к TrustStore и пароль к нему.

```
fun restTemplate(builder: RestTemplateBuilder): RestTemplate {
      val sslContext = SSLContextBuilder.create()
         .loadTrustMaterial(File(dirToSertificates + customTrustStore),
            customTrustStorePassword.toCharArray())
         .build()
      val httpClient = HttpClients.custom()
         .setSSLContext(sslContext)
         .setSSLHostnameVerifier(NoopHostnameVerifier.INSTANCE)
         .build()
      val customRequestFactory = HttpComponentsClientHttpRequestFactory()
      customRequestFactory.httpClient = httpClient
      return builder.requestFactory { customRequestFactory }.errorHandler(RestTemplateResponseErrorHandler())
         .messageConverters(
            StringHttpMessageConverter(
               StandardCharsets.UTF_8
```

```
            )
        ).build()
    }
```

Вывод:

В ходе выполнения данной лабораторной работы настроил два веб сервиса и взаимодействие между ними. Для двунаправленного защищенного соединения необходимо, чтобы у каждого вебсервиса были настроены keystore и truststore. В Truststore должны находится сертификаты другого веб-сервиса, чтобы можно было подтвердить валидность отправителя и используя публичный ключ, отправить сообщение другому веб-сервису. Также чтобы пересылать через RestTemplate запросы, нужно добавить SslContext, в который можно загрузить файл с сертификатами, иначе будет писать ошибку, что не было найдено подходящих сертификатов, хоть TrustStore настроен на сервере.