Санкт-Петербургский национальный исследовательский университет информационных технологий, механики и оптики

Сервис-ориентированная архитектура

Лабораторная работа №2

Вариант 3002.1

Группа: P33122

Студент:

Беляков Д.С

Преподаватель:

Усков И. В.

Санкт-Петербург

2021

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 JAX-RS 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 /bars и реализовывать следующие операции:

- `/labwork/{labwork-id}/difficulty/increase/{steps-count}` : повысить сложность заданной лабораторной работы на указанное число "шагов"
- `/discipline/{discipline-id}/labwork/{labwork-id}/remove` : удалить лабораторную работу из программы дисциплины

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

Код: https://github.com/kevinche75/soa_lab2_itmo_autumn_2021/

Создание сертификатов

keytool -genkey -alias wildfly -keyalg RSA -keystore wildflystore -validity 999 -keysize 2048

keytool -export -alias wildfly -keyalg RSA -keystore wildflystore -file wildflytrust.crt

keytool -import -alias wildfly -keyalg RSA -keystore payaratruststore.jks -file wildflytrust.crt

keytool -genkey -alias payara -keyalg RSA -keystore payarastore -validity 999 -keysize 2048

keytool -export -alias payara -keyalg RSA -keystore payarastore -file payaratrust.crt

keytool -import -alias payara -keyalg RSA -keystore wildflytruststore.jks -file payaratrust.crt

Настройка серверов

Payara Micro

java -Djavax.net.ssl.keyStore="/Users/kevinche75/servers/certificates/payarastore" \

-Djavax.net.ssl.keyStorePassword="soasoa" \

-Djavax.net.ssl.trustStore="/Users/kevinche75/servers/certificates/payaratruststore.jks" \

-Djavax.net.ssl.trustStorePassword="soasoa" \

-jar payara-micro-5.2021.8.jar \

--deploy /Users/kevinche75/IdeaProjects/soa_lab2_itmo_autumn_2021/back2/target/back2-1.0-SNAPSHOT:/lab2 \

--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

```
<security-realms>
        <security-realm name="SoaRealm">
            <server-identities>
                <ssl>
                    <keystore path="../../../certificates/wildflystore" relative-to="jboss.server.config.dir" keystore-password="soasoa"/>
                </ssl>
            </server-identities>
            <authentication>
                <truststore path="../../../certificates/wildflytruststore.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

Чтобы второй сервис мог общаться с первым, добавляется SSLContext в JAX-RS Client

```
private Client createClientBuilderSSL() {
   SSLContext sslContext = SslConfigurator.newInstance()
         .keyPassword("soasoa")
         .trustStorePassword("soasoa")
         .createSSLContext();
   HostnameVerifier hostnameVerifier = (hostname, sslSession) -> {
      System.out.println(" hostname = " + hostname);
      if (hostname.equals("localhost")) {
         return true;
      }
      return false;
   };
   return ClientBuilder.newBuilder()
         .sslContext(sslContext)
         .hostnameVerifier(hostnameVerifier)
         .build();
}
```

Вывод:

В ходе выполнения данной работы я сделал второй сервис, дополнил его сущностью, которая существует только там, реализовал методы для работы со новой сущностью, настроил базу данных для корректной работы со связанной сущностью на первом сервисе, а также настроил двунаправленное защищенное соединение между двумя сервисами.