

Университет «ИТМО»
Факультет Программной Инженерии и Компьютерной Техники
Дисциплина: Сервис-ориентированная архитектура

Лабораторная работа №2

Выполнили: Тучин Артём Евгеньевич,
Никитин Егор Алексеевич

Вариант: 885

Группа: Р34111

Преподаватель: Кривоносов Егор Дмитриевич

г. Санкт-Петербург, 2024г.

Задание

Вариант - 885

На основе разработанной в рамках лабораторной работы #1 спецификации реализовать два веб-сервиса и использующее их API клиентское приложение.

Требования к реализации и развёртыванию сервисов:

- Первый ("вызываемый") веб-сервис должен быть реализован на фреймворке Spring MVC REST и развёрнут в окружении под управлением сервера приложений Jetty.
- Второй веб-сервис должен быть реализован на фреймворке JAX-RS, развёрнут в окружении под управлением сервера приложений WildFly и вызывать REST API первого сервиса.
- Для обоих сервисов необходимо реализовать все функции, задокументированные в API, в строгом соответствии со спецификацией!
- 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.

Требования к клиентскому приложению:

- Клиентское приложение может быть написано на любом веб-фреймворке, который можно запустить на сервере helios.
- Приложение должно обеспечить полный набор возможностей, предоставляемых API обоих сервисов -- включая сортировку, фильтрацию и страничный вывод элементов коллекции.
- Приложение должно преобразовывать передаваемые сервисами данные в человеко-читаемый вид -- параграф текста, таблицу и т.д.
- Клиентское приложение должно информировать пользователя об ошибках, возникающих на стороне сервисов, в частности, о том, что сервису были отправлены невалидные данные.

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

Клиентское приложение

<https://se.ifmo.ru/~s335133/SOA2>

Person page

Filters

id ▾

Identical ▾

Add

Sorting

id ▾

ascending ▾

Add

Find person by filters and sorting

+

Showing 0 to 0 of 0 << < > >> 9 ▾

Height-related statistics

Average of all ▾

Resolve

Statistics related to demographics

Get percentage of hair color ▾

Select hair color ▾

Resolve

Graphics

Select hair color ▾

Show

Исходный код

Веб-сервисы - <https://github.com/artem00475/soa2>

Клиентское приложение - <https://github.com/egoryu/SOA2>

Настройка и развертывание

Настройка сертификатов серверов

Для генерации самоподписанных сертификатов используем утилиту keytool. Сначала создаем пары ключей RSA для наших сервисов и помещаем их в хранилища:

```
keytool -genkeypair -alias localhost -keyalg RSA -keysize 2048 -validity 365 -keystore server.jks -dname "cn=ServerAdministrator,o=Acme,c=GB" -keypass certificate -storepass certificate
```

```
keytool -genkeypair -alias localhost -keyalg RSA -keysize 2048 -validity 365 -keystore application.keystore -dname "cn=ServerAdministrator,o=Acme,c=GB" -keypass password -storepass password
```

Далее, чтобы второй сервис при обращении к первому смог подтвердить сертификат, необходимо экспортировать сертификат первого сервиса и поместить его в хранилище доверенных сертификатов:

```
keytool -exportcert -keystore server.jks -alias localhost -keypass certificate -storepass certificate -file server.crt
```

```
keytool -importcert -keystore application.truststore -storepass password -alias localhost -trustcacerts -file server.crt -noprompt
```

В итоге мы создали два keystore и truststore. Для конфигурации первого сервиса кладем server.jks в директорию src/main/resources/keystore и задаем следующие параметры в application.properties:

```
server.ssl.key-store-type=JKS
server.ssl.key-store=classpath:keystore/server.jks
server.ssl.key-store-password=certificate
server.ssl.key-alias=localhost
server.ssl.enabled=true
```

Для настройки ssl на втором сервисе, который разворачивается на WildFly, кладем application.keystore и application.truststore в директорию wildfly-33.0.2.Final\standalone\configuration, там же в standalone.xml редактируем параметры:

```
<?xml version="1.0" encoding="UTF-8"?>
<tls>
  <key-stores>
    <key-store name="applicationKS">
      <credential-reference clear-text="password"/>
      <implementation type="JKS"/>
      <file path="application.keystore" relative-to="jboss.server.config.dir"/>
    </key-store>
  </key-stores>
  <key-managers>
    <key-manager name="applicationKM" key-store="applicationKS" generate-self-signed-certificate-host="localhost">
      <credential-reference clear-text="password"/>
    </key-manager>
  </key-managers>
</tls>
```

Далее в wildfly-33.0.2.Final\bin\standalone.sh добавляем параметры запуска:

```
while true; do
  if [ "x$LAUNCH_JBOSS_IN_BACKGROUND" = "x" ]; then
    # Execute the JVM in the foreground
    eval \"\$JAVA\" -D\"[Standalone]\" \"\$JAVA_OPTS\" \
      \"-Dorg.jboss.boot.log.file=\"$JBOSS_LOG_DIR/server.log\" \
      \"-Djavax.net.ssl.trustStore=\"$JBOSS_CONFIG_DIR/application.truststore\" \
      \"-Djavax.net.ssl.trustStorePassword=password\" \
      \"-Dlogging.configuration=file:\"$JBOSS_CONFIG_DIR/logging.properties\" \
      -jar \"\"$JBOSS_HOME\"/jboss-modules.jar\" \
      \"$MODULE_OPTS\" \
      -mp \"\"${JBOSS_MODULEPATH}\"\" \
      org.jboss.as.standalone \
      -Djboss.home.dir=\"\"$JBOSS_HOME\"\" \
      -Djboss.server.base.dir=\"\"$JBOSS_BASE_DIR\"\" \
      \"$SERVER_OPTS\"
    JBOSS_STATUS=$?
  fi
done
```

Развертывание

Для первого сервиса в src/main/resources/application.properties задаем номер порта и параметры подключения к бд, командой mvn package собираем JAR-архив, перемещаем его на helios и запускаем командой java -jar ...

Для второго сервиса необходима скачать и распаковать архив с WildFly на helios, в 33.0.2.Final\standalone\configuration\standalone.xml поменять хост с 127.0.0.1 на 0.0.0.0 и настроить используемые порты. Далее командой mvn package собираем WAR-архив и переносим его на helios в wildfly-33.0.2.Final\standalone\deployments. Для запуска выполняем скрипт wildfly-33.0.2.Final\bin\standalone.sh.

Для клиентского приложения нужно командой `ng build` собрать проект, далее перенести содержимое директории `dist` в `public_html` на `helios`.

Вывод

Выполнив данную работу, мы по разработанной ранее спецификации реализовали два веб-сервиса и использующее их API клиентское приложение, развернули сервисы на `helios`. Так же познакомились с протоколом `https`, настроили работу сервисов на этот протокол, создав самоподписанные сертификаты.