Университет ИТМО

ФПИ и КТ

Лабораторная работа №4

По дисциплине Сервис-ориентированная архитектура

Вариант 673

Выполнили: Тучин А.Е.

Никитин Е.А.

Группа: P34111

Преподаватель: Кривоносов Е.Д.

Санкт-Петербург

2025

Задание

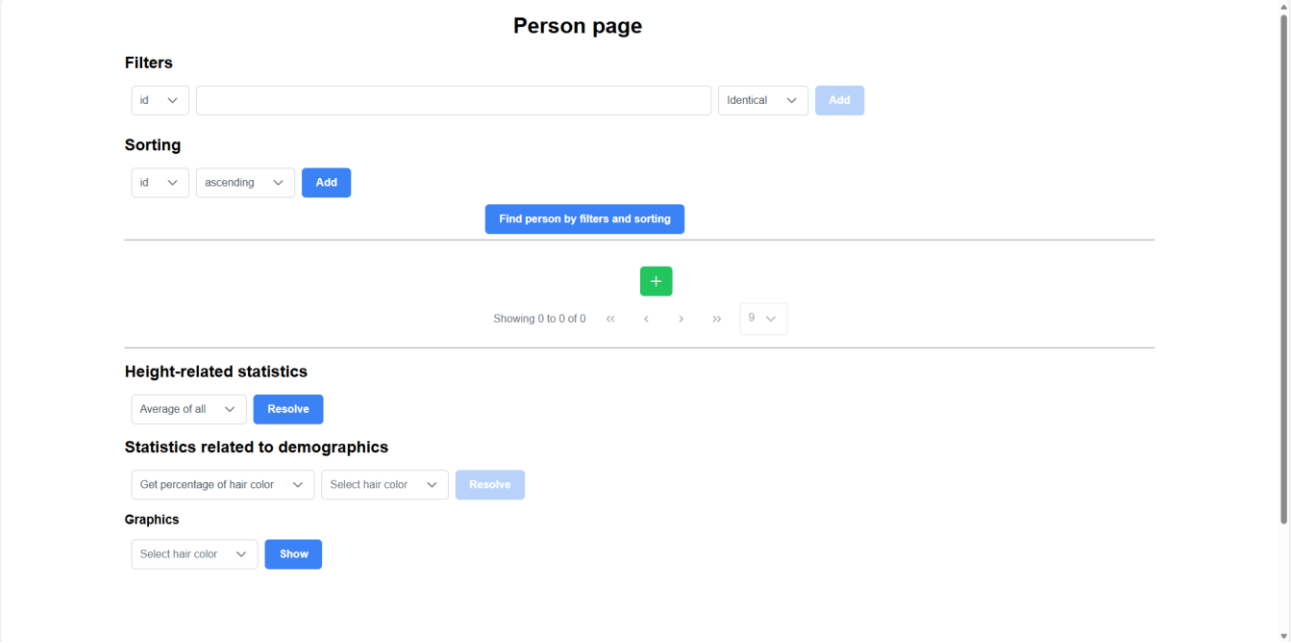
Переработать сервисы из лабораторной работы #3 следующим образом:

- Первый ("вызываемый") сервис переписать в соответствии с требованиями протокола SOAP.
- Развернуть переработанный сервис на сервере приложений по собственному выбору.
- Оставшийся сервис не модифицировать, не менять его API, протокол и используемый сервер приложений.
- Установить и сконфигурировать на сервере Helios программное обеспечение Mule ESB.
- Настроить интеграцию двух сервисов с использованием установленного программного обеспечения.
- Реализовать дополнительную REST-"прослойку", обеспечивающую возможность доступа к переработанному сервису клиентского приложения без необходимости его модификации. Никакой дополнительной логики, помимо вызовов SOAP-сервиса, разработанная REST-прослойка содержать не должна.

Выполнение

Клиентское приложение

[Frontend]



Исходный код

Веб-сервисы и mule: https://github.com/artem00475/soa4

Клиентское приложение: https://github.com/egoryu/SOA2

Настройка и развертывание

Вызываемый сервис

Создаем xsd файл

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
        xmlns:tns="http://se/ifmo/ru/firstservice/person"
        targetNamespace="http://se/ifmo/ru/firstservice/person"
        elementFormDefault="qualified">
```

```
  <xs:element name="getColorRequest"/>

  <xs:element name="getColorResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="data" type="tns:ColorsResponse"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Добавляем Endpoint

```
@Endpoint
@RequiredArgsConstructor
public class PeopleEndpoint {
    private static final String NAMESPACE_URI = "http://se/ifmo/ru/firstservice/person";
    private final PersonService personService;

    @PayloadRoot(namespace = NAMESPACE_URI, localPart = "getColorRequest")
    @ResponsePayload
    public GetColorResponse getColors() {
        GetColorResponse response = new GetColorResponse();
        ColorsResponse colorsResponse = new ColorsResponse();
        List<ColorEnum> data = colorsResponse.getColor();
        data.addAll(Arrays.stream(ColorEnum.values()).toList());
        response.setData(colorsResponse);

        return response;
    }
```

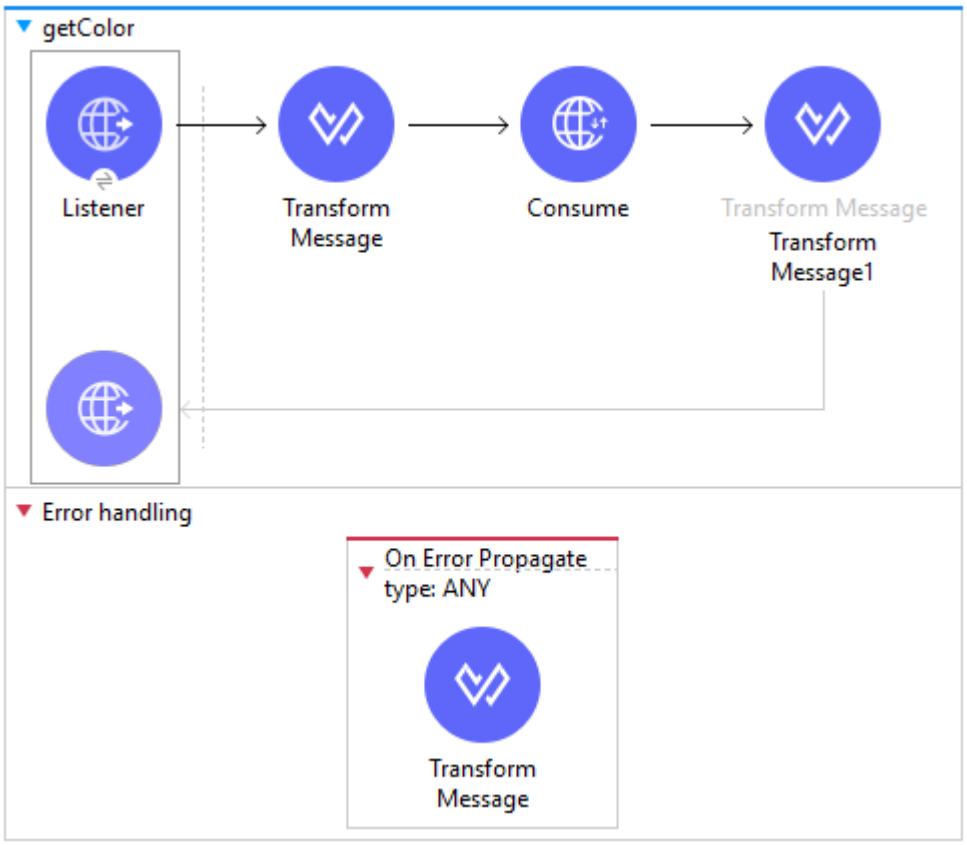
И настраиваем веб сервис

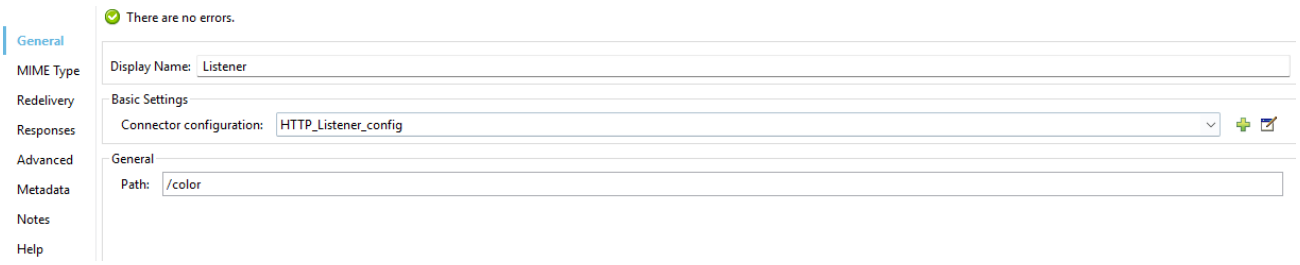
```
@EnableWs
@Configuration
public class WebServiceConfig extends WsConfigurerAdapter {
    @Bean
    public ServletRegistrationBean<MessageDispatcherServlet>
messageDispatcherServlet(ApplicationContext applicationContext) {
```

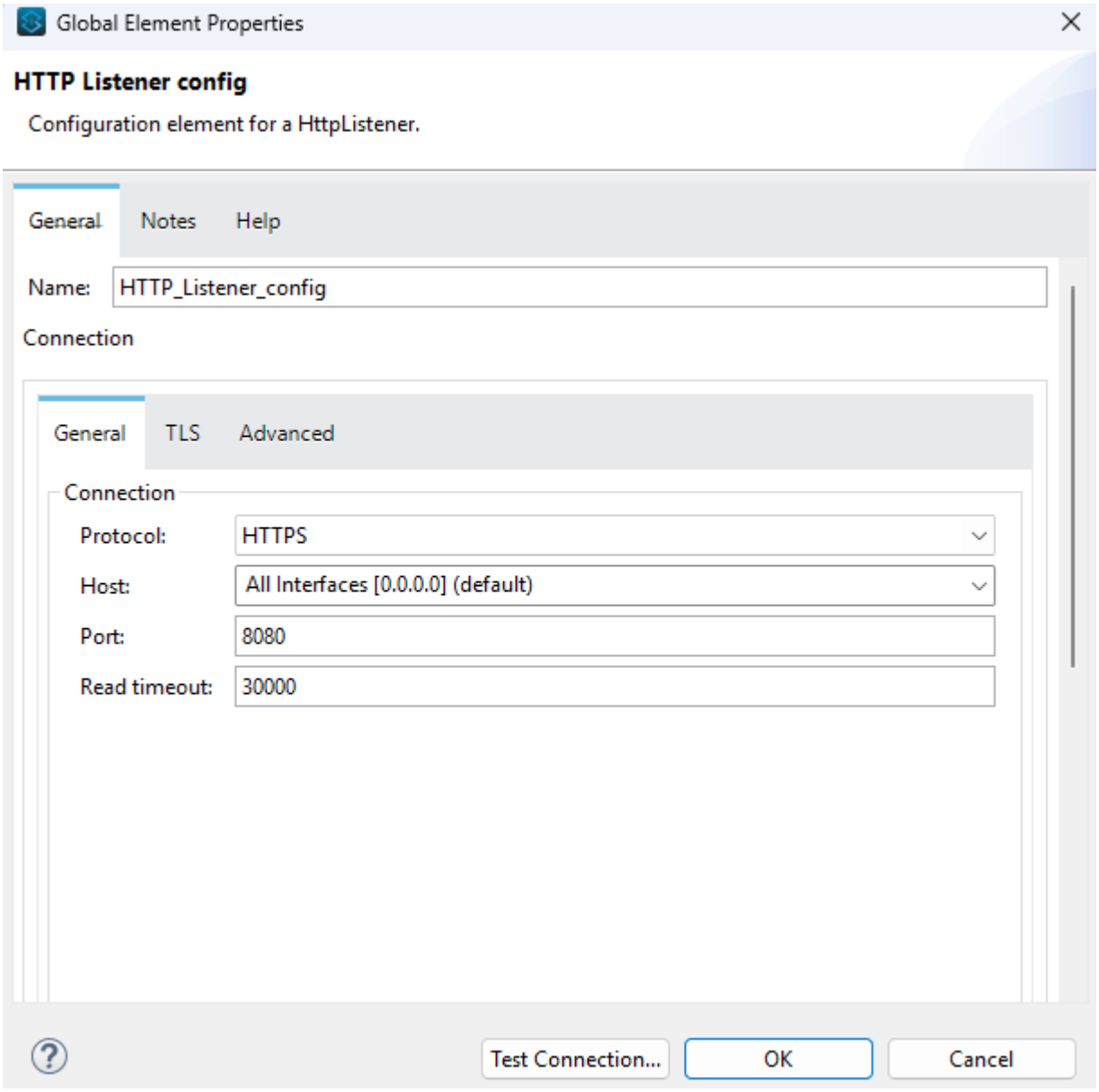
```
    MessageDispatcherServlet servlet = new MessageDispatcherServlet();
    servlet.setApplicationContext(applicationContext);
    servlet.setTransformWsdlLocations(true);
    return new ServletRegistrationBean<>(servlet, "/*");
}

@Bean(name = "person")
public DefaultWsdl11Definition defaultWsdl11Definition(XsdSchema personSchema) {
    DefaultWsdl11Definition wsdl11Definition = new DefaultWsdl11Definition();
    wsdl11Definition.setPortTypeName("PersonPort");
    wsdl11Definition.setLocationUri("/");
    wsdl11Definition.setTargetNamespace("http://se/ifmo/ru/firstservice/person");
    wsdl11Definition.setSchema(personSchema);
    return wsdl11Definition;
}
```

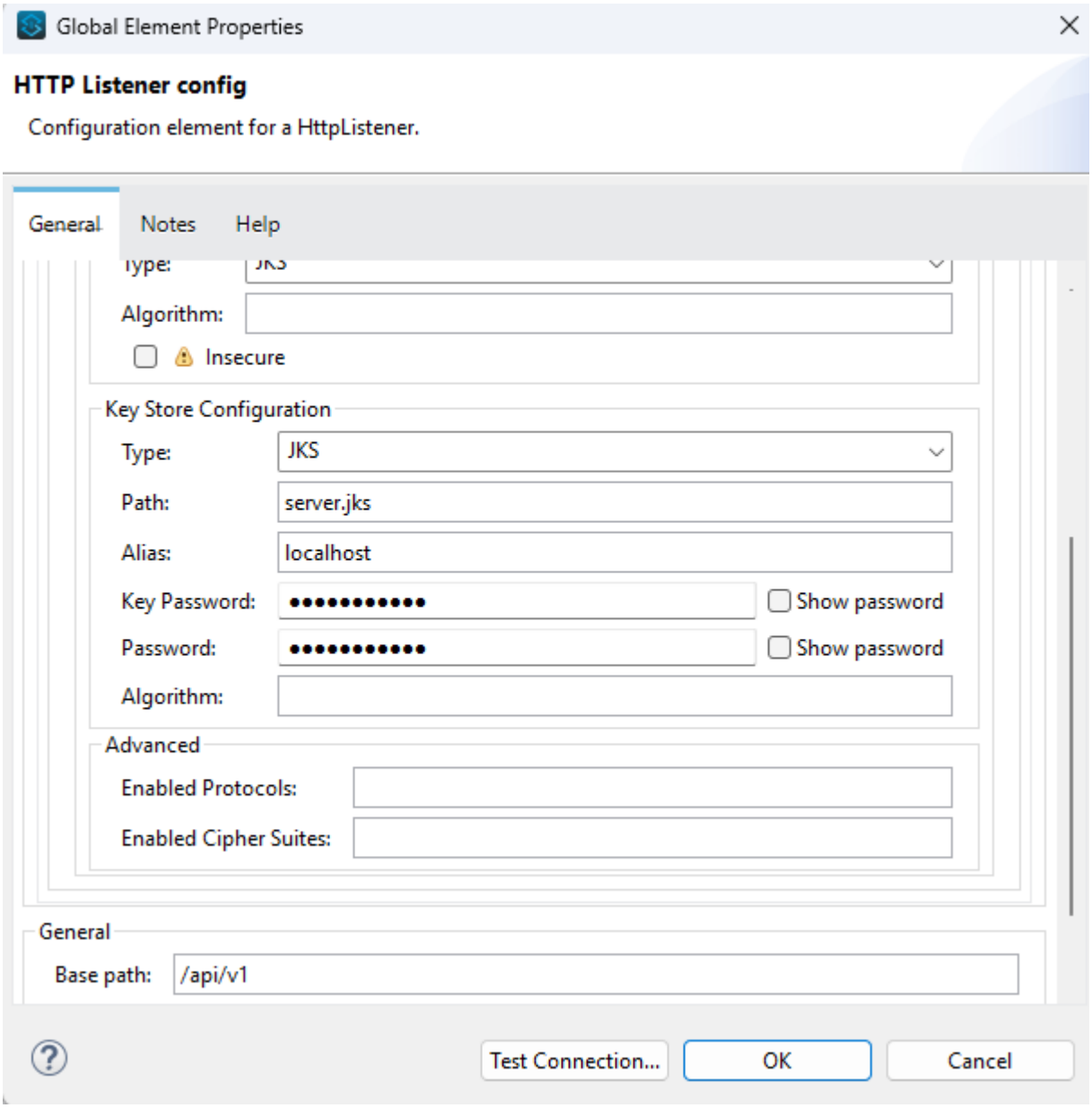
Mule

Создаем flow для каждого запроса, в который настраиваем listener, consumer, error handler и два transform massage




There are no errors.
General
MIME Type
Redelivery
Responses
Advanced
Metadata
Notes
Help
Display Name: Listener
Basic Settings
Connector configuration: HTTP_Listener_config
General
Path: /color

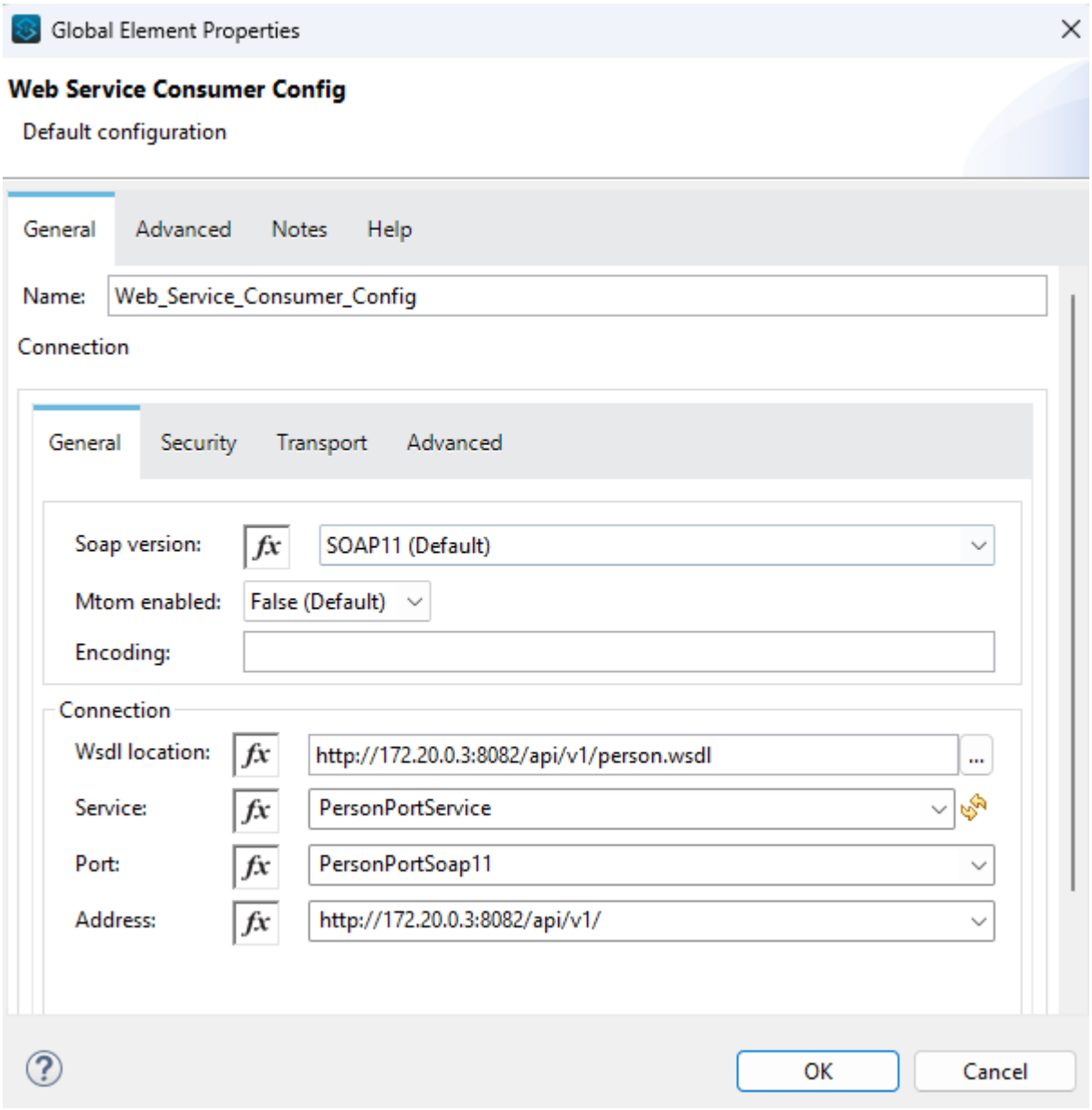

Global Element Properties
HTTP Listener config
Configuration element for a HttpListener.
General
Notes
Help
Name: HTTP_Listener_config
Connection
General
TLS
Advanced
Connection
Protocol: HTTPS
Host: All Interfaces [0.0.0.0] (default)
Port: 8080
Read timeout: 30000
Test Connection...
OK
Cancel


Global Element Properties
HTTP Listener config
Configuration element for a HttpListener.
General
Notes
Help
Algorithm:
Insecure
Key Store Configuration
Type:
JKS
Path:
server.jks
Alias:
localhost
Key Password:
Show password
Password:
Show password
Algorithm:
Advanced
Enabled Protocols:
Enabled Cipher Suites:
General
Base path:
/api/v1
Test Connection...
OK
Cancel


Output
Payload
Preview
%dw 2.0
ns ns0 http://se/ifmo/ru/firstservice/person
output text/xml

{
ns0#getColorRequest: {
},
}


There are no errors.
General
Advanced
Error Mapping
Metadata
Notes
Help
Display Name: Consume
Basic Settings
Connector configuration: Web_Service_Consumer_Config
General
Operation: getColor
Message
Body:
1 payload


Global Element Properties
Web Service Consumer Config
Default configuration
General
Advanced
Notes
Help
Name: Web_Service_Consumer_Config
Connection
General
Security
Transport
Advanced
Soap version: SOAP11 (Default)
Mtom enabled: False (Default)
Encoding:
Connection
Wsdl location: http://172.20.0.3:8082/api/v1/person.wsdl
Service: PersonPortService
Port: PersonPortSoap11
Address: http://172.20.0.3:8082/api/v1/
OK
Cancel

```
%dw 2.0
ns ns0 http://se/ifmo/ru/firstservice/person
output application/json
---
{
    "data": payload.body.ns0#getColorResponse.ns0#data.*ns0#color map(item, index) -> item
}
```

```
%dw 2.0 output application/json
var detail = read(error.exception.cause.detail, "text/xml")
---
{
    "message": detail.detail.message
}
```

Развертывание

```
version: "3"

services:
  postgres:
    image: postgres:latest
    environment:
      POSTGRES_DB: "soa"
      POSTGRES_USER: "postgres"
      POSTGRES_PASSWORD: "postgres"
    ports:
      - "5432:5432"
    healthcheck:
      test: [ "CMD-SHELL", "pg_isready -d soa" ]
      interval: 10s
      timeout: 5s
      retries: 5
    restart: unless-stopped
    networks:
      dev:
        ipv4_address: 172.20.0.2
    deploy:
      resources:
        limits:
          cpus: '1'
          memory: 4G

  first:
```

```
    build: first-service/
    command: java -jar ./first-service-0.0.1-SNAPSHOT.jar
    ports:
      - "8082:8082"
    links:
      - postgres
    networks:
      dev:
        ipv4_address: 172.20.0.3
    depends_on:
      postgres:
        condition: service_healthy

  second:
    build: second-service/
    ports:
      - "8081:8081"
      - "9990:9990"
    networks:
      dev:
        ipv4_address: 172.20.0.8

  mule:
    build: mule/
    ports:
      - "8080:8080"
    networks:
      dev:
        ipv4_address: 172.20.0.9

networks:
  dev:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 172.20.0.0/16
```

Mule

```
FROM openjdk:8

RUN cd /opt \
&& wget https://s3.amazonaws.com/new-mule-artifacts/mule-ee-distribution-standalone-4.4.0.zip
```

```
\
&& unzip *.zip \
&& ln -s /opt/mule-enterprise-standalone-4.4.0 /opt/mule \
&& rm /opt/mule-ee-distribution-standalone-4.4.0.zip

# Define environment variables.
ENV MULE_HOME /opt/mule

# Define mount points.
VOLUME ["/opt/mule/logs", "/opt/mule/apps", "/opt/mule/domains"]

# Define working directory.
WORKDIR /opt/mule

ADD apps/lab4.jar /opt/mule/apps/

CMD ["/opt/mule/bin/mule"]
```

Вывод

В ходе лабораторной работы мы переписали вызываемый сервис под SOAP стандарт и смогли настроить Mule ESB для обращение к SOAP сервису с помощью REST запросов.