

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерных технологий

Сервисно-ориентированная архитектура

Лабораторная работа № 4

Вариант 1006

Выполнил  
студент

Демичев Даниил Дмитриевич

Группа № Р34112

Преподаватель: Усков Иван Владимирович

г. Санкт-Петербург

2022

## Задание

- 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.

## Исходный код

<https://github.com/xxPFFxx/SOA-lab3>

## Конфигурация интерфейса и реализации WebService на вызываемом сервисе

```
@WebService
public interface HumanBeingWebService {
    @WebMethod
    HumanBeingDTOList getHumanBeings(@WebParam(name = "perPage") String perPage,
    @WebParam(name = "curPage") String curPage, @WebParam(name = "sortBy") String
    sortBy, @WebParam(name = "filterBy") String filterBy) throws MyBadRequestException,
    MyNotFoundException;
    @WebMethod
    HumanBeingDTO getHumanBeing(@WebParam(name = "id") String id) throws
    MyBadRequestException, MyNotFoundException;
    @WebMethod
    void createHumanBeing(@WebParam(name = "humanBeing") String humanBeing)
    throws MyBadRequestException, MyNotFoundException;
    @WebMethod
    void updateHumanBeing(@WebParam(name = "id") String id, @WebParam(name =
    "humanBeing") String humanBeing) throws MyBadRequestException,
    MyNotFoundException;
    @WebMethod
    void deleteHumanBeing(@WebParam(name = "id") String id) throws
    MyBadRequestException, MyNotFoundException;
```

```
}
```

```
@WebService(endpointInterface = "com.soa.controllers.HumanBeingWebService")  
public class HumanBeingWebServiceImpl implements HumanBeingWebService {
```

```
    private final HumanBeingServiceInterface humanBeingServiceInterface;  
    private final HumanBeingMapper humanBeingMapper;
```

```
    public HumanBeingWebServiceImpl(){  
        humanBeingMapper = new HumanBeingMapper();  
        humanBeingServiceInterface = RemoteBeanUtil.lookupRemoteStatelessBean();  
    }
```

```
    @WebMethod
```

```
    public HumanBeingDTOList getHumanBeings(@WebParam(name = "perPage") String  
perPage, @WebParam(name = "curPage") String curPage, @WebParam(name = "sortBy")  
String sortBy, @WebParam(name = "filterBy") String filterBy) throws  
MyBadRequestException, MyNotFoundException {
```

```
        try{  
            PagedHumanBeingList pagedHumanBeingList =  
humanBeingServiceInterface.getHumanBeings(perPage, curPage, sortBy, filterBy);  
            return new  
HumanBeingDTOList((humanBeingMapper.mapHumanBeingListToHumanBeingDTOList(pa  
gedHumanBeingList.getHumanBeingList()), pagedHumanBeingList.getCount());  
        }catch (EJBException e){  
            ServiceFault fault = new ServiceFault();  
            fault.setFaultString(Throwables.getRootCause(e).getMessage());  
            System.out.println(Throwables.getRootCause(e).getClass());  
            System.out.println(Throwables.getRootCause(e).getMessage());  
            System.out.println(e.getClass());  
            System.out.println(e.getMessage());  
            if (Throwables.getRootCause(e) instanceof NotFoundException){  
                System.out.println("InNotFound");  
                fault.setFaultCode("404");  
                throw new MyNotFoundException(fault);  
            }  
            else {  
                System.out.println("InBadRequest");  
                fault.setFaultCode("400");  
                throw new MyBadRequestException(fault);  
            }  
        }catch (Exception e){  
            ServiceFault fault = new ServiceFault();  
            fault.setFaultString("Error in the request");  
            fault.setFaultCode("400");  
            throw new MyBadRequestException(fault);  
        }
```

```
}  
}
```

```
@WebMethod  
public HumanBeingDTO getHumanBeing(@WebParam(name = "id") String id) throws  
MyBadRequestException, MyNotFoundException{  
    System.out.println(id);  
    Long long_id = FieldValidationUtil.getLongFieldValue(id);  
    try {  
        HumanBeing humanBeing = humanBeingServiceInterface.getHumanBeing(long_id);  
        return humanBeingMapper.mapHumanBeingToHumanBeingDTO(humanBeing);  
    }catch (EJBException e){  
        ServiceFault fault = new ServiceFault();  
        fault.setFaultString(Throwables.getRootCause(e).getMessage());  
        System.out.println(Throwables.getRootCause(e).getClass());  
        System.out.println(Throwables.getRootCause(e).getMessage());  
        if (Throwables.getRootCause(e) instanceof NotFoundException){  
            fault.setFaultCode("404");  
            throw new MyNotFoundException(fault);  
        }  
        else {  
            fault.setFaultCode("400");  
            throw new MyBadRequestException(fault);  
        }  
    }catch (Exception e){  
        ServiceFault fault = new ServiceFault();  
        fault.setFaultString("Error in the request");  
        fault.setFaultCode("400");  
        throw new MyBadRequestException(fault);  
    }  
}
```

```
}  
@WebMethod  
public void createHumanBeing(@WebParam(name = "humanBeing") String humanBeing)  
throws MyBadRequestException, MyNotFoundException{  
    try {  
        humanBeingServiceInterface.saveHumanBeing(humanBeing);  
    }catch (EJBException e){  
        ServiceFault fault = new ServiceFault();  
        fault.setFaultString(Throwables.getRootCause(e).getMessage());  
        System.out.println(Throwables.getRootCause(e).getClass());  
        System.out.println(Throwables.getRootCause(e).getMessage());  
        if (Throwables.getRootCause(e) instanceof NotFoundException){  
            fault.setFaultCode("404");  
            throw new MyNotFoundException(fault);  
        }  
    }  
}
```

```

        else {
            fault.setFaultCode("400");
            throw new MyBadRequestException(fault);
        }
    }catch (Exception e){
        ServiceFault fault = new ServiceFault();
        fault.setFaultString("Error in the request");
        fault.setFaultCode("400");
        throw new MyBadRequestException(fault);
    }
}

```

```

@WebMethod
public void updateHumanBeing(@WebParam(name = "id") String id, @WebParam(name
= "humanBeing") String humanBeing) throws MyBadRequestException,
MyNotFoundException{
    Long long_id = FieldValidationUtil.getLongFieldValue(id);
    try {
        humanBeingServiceInterface.updateHumanBeing(humanBeing, long_id);
    }catch (EJBException e){
        ServiceFault fault = new ServiceFault();
        fault.setFaultString(Throwables.getRootCause(e).getMessage());
        System.out.println(Throwables.getRootCause(e).getClass());
        System.out.println(Throwables.getRootCause(e).getMessage());
        if (Throwables.getRootCause(e) instanceof NotFoundException){
            fault.setFaultCode("404");
            throw new MyNotFoundException(fault);
        }
        else {
            fault.setFaultCode("400");
            throw new MyBadRequestException(fault);
        }
    }catch (Exception e){
        ServiceFault fault = new ServiceFault();
        fault.setFaultString("Error in the request");
        fault.setFaultCode("400");
        throw new MyBadRequestException(fault);
    }
}

```

```

@WebMethod
public void deleteHumanBeing(@WebParam(name = "id") String id) throws
MyBadRequestException, MyNotFoundException {
    Long long_id = FieldValidationUtil.getLongFieldValue(id);
    try {
        humanBeingServiceInterface.deleteHumanBeing(long_id);
    }catch (EJBException e){
        ServiceFault fault = new ServiceFault();

```

```

        fault.setFaultString(Throwables.getRootCause(e).getMessage());
        System.out.println(Throwables.getRootCause(e).getClass());
        System.out.println(Throwables.getRootCause(e).getMessage());
        if (Throwables.getRootCause(e) instanceof NotFoundException){
            fault.setFaultCode("404");
            throw new MyNotFoundException(fault);
        }
        else {
            fault.setFaultCode("400");
            throw new MyBadRequestException(fault);
        }
    }catch (Exception e){
        ServiceFault fault = new ServiceFault();
        fault.setFaultString("Error in the request");
        fault.setFaultCode("400");
        throw new MyBadRequestException(fault);
    }
}
}
}

```

## WSDL, созданный на основе WebService

```

<definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy"
xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy"
xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://controllers.soa.com/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://controllers.soa.com/"
name="HumanBeingWebServiceImplService">
<types>
<xsd:schema>
<xsd:import namespace="http://controllers.soa.com/"
schemaLocation="http://localhost:8080/humanbeingservice?xsd=1"/>
</xsd:schema>
</types>
<message name="deleteHumanBeing">
<part name="parameters" element="tns:deleteHumanBeing"/>
</message>
<message name="deleteHumanBeingResponse">
<part name="parameters" element="tns:deleteHumanBeingResponse"/>
</message>
<message name="MyBadRequestException">
<part name="fault" element="tns:MyBadRequestException"/>
</message>

```

```

<message name="MyNotFoundException">
<part name="fault" element="tns:MyNotFoundException"/>
</message>
<message name="createHumanBeing">
<part name="parameters" element="tns:createHumanBeing"/>
</message>
<message name="createHumanBeingResponse">
<part name="parameters" element="tns:createHumanBeingResponse"/>
</message>
<message name="updateHumanBeing">
<part name="parameters" element="tns:updateHumanBeing"/>
</message>
<message name="updateHumanBeingResponse">
<part name="parameters" element="tns:updateHumanBeingResponse"/>
</message>
<message name="getHumanBeings">
<part name="parameters" element="tns:getHumanBeings"/>
</message>
<message name="getHumanBeingsResponse">
<part name="parameters" element="tns:getHumanBeingsResponse"/>
</message>
<message name="getHumanBeing">
<part name="parameters" element="tns:getHumanBeing"/>
</message>
<message name="getHumanBeingResponse">
<part name="parameters" element="tns:getHumanBeingResponse"/>
</message>
<portType name="HumanBeingWebService">
<operation name="deleteHumanBeing">
<input
wsam:Action="http://controllers.soa.com/HumanBeingWebService/deleteHumanBeingReque
st" message="tns:deleteHumanBeing"/>
<output
wsam:Action="http://controllers.soa.com/HumanBeingWebService/deleteHumanBeingRespo
nse" message="tns:deleteHumanBeingResponse"/>
<fault message="tns:MyBadRequestException" name="MyBadRequestException"
wsam:Action="http://controllers.soa.com/HumanBeingWebService/deleteHumanBeing/Fault/
MyBadRequestException"/>
<fault message="tns:MyNotFoundException" name="MyNotFoundException"
wsam:Action="http://controllers.soa.com/HumanBeingWebService/deleteHumanBeing/Fault/
MyNotFoundException"/>
</operation>
<operation name="createHumanBeing">
<input
wsam:Action="http://controllers.soa.com/HumanBeingWebService/createHumanBeingReque
st" message="tns:createHumanBeing"/>

```

```

<output
wsam:Action="http://controllers.soa.com/HumanBeingWebService/createHumanBeingRespo
nse" message="tns:createHumanBeingResponse"/>
<fault message="tns:MyBadRequestException" name="MyBadRequestException"
wsam:Action="http://controllers.soa.com/HumanBeingWebService/createHumanBeing/Fault/
MyBadRequestException"/>
<fault message="tns:MyNotFoundException" name="MyNotFoundException"
wsam:Action="http://controllers.soa.com/HumanBeingWebService/createHumanBeing/Fault/
MyNotFoundException"/>
</operation>
<operation name="updateHumanBeing">
<input
wsam:Action="http://controllers.soa.com/HumanBeingWebService/updateHumanBeingRequ
est" message="tns:updateHumanBeing"/>
<output
wsam:Action="http://controllers.soa.com/HumanBeingWebService/updateHumanBeingResp
onse" message="tns:updateHumanBeingResponse"/>
<fault message="tns:MyBadRequestException" name="MyBadRequestException"
wsam:Action="http://controllers.soa.com/HumanBeingWebService/updateHumanBeing/Fault/
MyBadRequestException"/>
<fault message="tns:MyNotFoundException" name="MyNotFoundException"
wsam:Action="http://controllers.soa.com/HumanBeingWebService/updateHumanBeing/Fault/
MyNotFoundException"/>
</operation>
<operation name="getHumanBeings">
<input
wsam:Action="http://controllers.soa.com/HumanBeingWebService/getHumanBeingsRequest
" message="tns:getHumanBeings"/>
<output
wsam:Action="http://controllers.soa.com/HumanBeingWebService/getHumanBeingsRespon
se" message="tns:getHumanBeingsResponse"/>
<fault message="tns:MyBadRequestException" name="MyBadRequestException"
wsam:Action="http://controllers.soa.com/HumanBeingWebService/getHumanBeings/Fault/M
yBadRequestException"/>
<fault message="tns:MyNotFoundException" name="MyNotFoundException"
wsam:Action="http://controllers.soa.com/HumanBeingWebService/getHumanBeings/Fault/M
yNotFoundException"/>
</operation>
<operation name="getHumanBeing">
<input
wsam:Action="http://controllers.soa.com/HumanBeingWebService/getHumanBeingRequest"
message="tns:getHumanBeing"/>
<output
wsam:Action="http://controllers.soa.com/HumanBeingWebService/getHumanBeingRespons
e" message="tns:getHumanBeingResponse"/>
<fault message="tns:MyBadRequestException" name="MyBadRequestException"
wsam:Action="http://controllers.soa.com/HumanBeingWebService/getHumanBeing/Fault/M
yBadRequestException"/>

```



```
<fault message="tns:MyNotFoundException" name="MyNotFoundException"
wsam:Action="http://controllers.soa.com/HumanBeingWebService/getHumanBeing/Fault/My
NotFoundException"/>
</operation>
</portType>
<binding name="HumanBeingWebServiceImplPortBinding"
type="tns:HumanBeingWebService">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<operation name="deleteHumanBeing">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
<fault name="MyBadRequestException">
<soap:fault name="MyBadRequestException" use="literal"/>
</fault>
<fault name="MyNotFoundException">
<soap:fault name="MyNotFoundException" use="literal"/>
</fault>
</operation>
<operation name="createHumanBeing">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
<fault name="MyBadRequestException">
<soap:fault name="MyBadRequestException" use="literal"/>
</fault>
<fault name="MyNotFoundException">
<soap:fault name="MyNotFoundException" use="literal"/>
</fault>
</operation>
<operation name="updateHumanBeing">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
<fault name="MyBadRequestException">
<soap:fault name="MyBadRequestException" use="literal"/>
```

```

</fault>
<fault name="MyNotFoundException">
<soap:fault name="MyNotFoundException" use="literal"/>
</fault>
</operation>
<operation name="getHumanBeings">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
<fault name="MyBadRequestException">
<soap:fault name="MyBadRequestException" use="literal"/>
</fault>
<fault name="MyNotFoundException">
<soap:fault name="MyNotFoundException" use="literal"/>
</fault>
</operation>
<operation name="getHumanBeing">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
<fault name="MyBadRequestException">
<soap:fault name="MyBadRequestException" use="literal"/>
</fault>
<fault name="MyNotFoundException">
<soap:fault name="MyNotFoundException" use="literal"/>
</fault>
</operation>
</binding>
<service name="HumanBeingWebServiceImplService">
<port name="HumanBeingWebServiceImplPort"
binding="tns:HumanBeingWebServiceImplPortBinding">
<soap:address location="http://localhost:8080/humanbeingservice"/>
</port>
</service>
</definitions>

```

## Обработка ошибок в JAX-WS

```
public class MyBadRequestException extends Exception implements Serializable {
```

```

private static final long serialVersionUID = 10L;
private ServiceFault fault;
public MyBadRequestException(String message) {
    super(message);
}
public MyBadRequestException(Throwable throwable) {
    super(throwable);
}
public MyBadRequestException(String message, Throwable throwable) {
    super(message, throwable);
}
public MyBadRequestException() {
}
public MyBadRequestException(ServiceFault fault) {
    super(fault.getFaultString());
    this.fault = fault;
}
public MyBadRequestException(String message, ServiceFault faultInfo) {
    super(message);
    this.fault = faultInfo;
}
public MyBadRequestException(String message, ServiceFault faultInfo, Throwable cause)
{
    super(message, cause);
    this.fault = faultInfo;
}
public ServiceFault getFaultInfo() {
    return fault;
}
public MyBadRequestException(String code, String message) {
    super(message);
    this.fault = new ServiceFault();
    this.fault.setFaultString(message);
    this.fault.setFaultCode(code);
}
}

```

```

ServiceFault fault = new ServiceFault();
    fault.setFaultString("Error in the request");
    fault.setFaultCode("400");
    throw new MyBadRequestException(fault);

```

## Настройка MULE в AnyPoint Studio



## Вывод

В ходе выполнения этой лабораторной работы я переписал вызываемый сервис в соответствии с протоколом SOAP, установил и сконфигурировал программное обеспечение Mule ESB, и в программе AnyPoint Studio настроил REST-прослойку для вызова SOAP-сервиса.