SOAP-сервисы в Java Proxy
Теория и практика

Федечкин Сергей, SAP
6 сентября, 2019

Agenda

Теория

- Основные паттерны SOAP
- Аннотации Java
- Конфигурация SOAP-сервисов

Практика

- Вызов oneway-сервиса
- Обработка ошибок

Итоги

Зачем?! Ведь SOAP:

- тяжелый
- избыточный
- устаревший
- немодный

Да! Но:

- это базовый протокол SOA в целом и SAP Netweaver on-premise в частности
- накоплен большой объем знаний и опыта
- все еще популярен в Java EE - разработке
- имеет подмножество стандартизированных протоколов WS-*

Немного теории

Основные паттерны SOAP

Теория

Message exchange pattern (MEP) определяется в WSDL сервиса в PortType в виде операции:

```
<wsdl:portType name="nmtoken">
    <wsdl:operation name="nmtoken" .... />
</wsdl:portType>
```

В WSDL 1.1 определено 4 типа операции:

- **One-way** – endpoint только принимает сообщение и не отправляет ответ
- **Request-response** – endpoint принимает запрос и отправляет ответ
- Solicit-response – endpoint отправляет запрос и ждет ответ
- Notification – endpoint отправляет сообщение и не ждет ответ

Основные паттерны SOAP

Теория

- One-way = асинхронный сервис в PI:

```
<wsdl:operation name="nmtoken">
        <wsdl:input name="nmtoken"? message="qname"/>
 </wsdl:operation>
```

- Request-response = синхронный сервис в PI:

```
<wsdl:operation name="nmtoken" parameterOrder="nmtokens">
        <wsdl:input name="nmtoken"? message="qname"/>
        <wsdl:output name="nmtoken"? message="qname"/>
        <wsdl:fault name="nmtoken" message="qname"/>*
</wsdl:operation>
```

Аннотации Java

Теория

JWS - Web Services Metadata Annotations (JSR-181)

JAX-WS - Java API for XML Based Web Services (JSR-224)

- SAP Netweaver 7.5 – реализация JAX-WS 2.0 (help.sap.com)
- SAP JVM 8 – JAX-WS 2.2.9 (wsimport -version, NW 7.5 SP14)
- Последняя версия JAX-WS RI – 2.3.1 (javaee.github.io/metro-jax-ws/)

Аннотации Java

Теория

Создание сервиса (Provider):

- **@WebService** в классе (SIB) и интерфейсе (SEI, опционально)
- **@WebServiceProvider** в классе SIB с имплементацией интерфейса *javax.xml.ws.Provider*
- **@WebMethod** в методе для мэппинга операции
- **@WebResult** в методе для мэппинга ответа
- **@Oneway** в методе для определения операции без ответа

Вызов сервиса (Consumer):

- **@WebService** в интерфейсе (SEI)
- **@WebServiceClient** в сгенерированном клиенте сервиса
- **@ WebServiceRef** в классе для определения ссылки на сгенерированный клиент

Server proxy (Provider)

Теория

- Service Endpoint Interface

```
@javax.jws.WebService(name = "SI_Request_In", targetNamespace = "http://pimon2019.ru")
@javax.jws.soap.SOAPBinding(parameterStyle = javax.jws.soap.SOAPBinding.ParameterStyle.BARE, style =
javax.jws.soap.SOAPBinding.Style.DOCUMENT, use = javax.jws.soap.SOAPBinding.Use.LITERAL)
public interface SIRequestIn {

  @javax.jws.WebMethod(operationName = "SI_Request_In", action = "http://sap.com/xi/WebService/soap1.1")
  @javax.jws.Oneway
  public void send(@javax.jws.WebParam(name = "MT_Request", targetNamespace = "http://pimon2019.ru", partName =
"MT_Request") ru.pimon2019.MT_Request MT_Request);

}
```

- Service Implementation Bean

```
@WebService(portName = "SI_Request_In_Port", serviceName = "SI_Request_In_Service", endpointInterface =
"ru.pimon2019.SIRequestIn", targetNamespace = "http://pimon2019.ru", wsdlLocation = "META-
INF/wsdl/ru/pimon2019/SI_Request_In/SI_Request_In.wsdl")
@Stateless
public class SIRequestInImplBean {
    public  void send(ru.pimon2019.MT_Request MT_Request){...}
}
```

Client proxy (Consumer)

Теория

- Service Endpoint Interface

```
@javax.jws.WebService(name = "SI_Request_Out", targetNamespace = "http://pimon2019.ru")
@javax.jws.soap.SOAPBinding(parameterStyle = javax.jws.soap.SOAPBinding.ParameterStyle.BARE, style =
javax.jws.soap.SOAPBinding.Style.DOCUMENT, use = javax.jws.soap.SOAPBinding.Use.LITERAL)
public interface SIRequestOut {

  @javax.jws.WebMethod(operationName = "SI_Request_Out", action = "http://sap.com/xi/WebService/soap1.1")
  @javax.jws.Oneway
  public void send(@javax.jws.WebParam(name = "MT_Request", targetNamespace = "http://pimon2019.ru", partName =
"MT_Document") ru.pimon2019.MT_Request MT_Request);

}
```

- Service Implementation Client

```
@javax.xml.ws.WebServiceClient(name = "SI_Request_Out_Service", targetNamespace = "http://pimon2019.ru",
wsdlLocation = "META-INF/wsdl/ru/pimon2019/SI_Request_Out/SI_Request_Out.wsdl")
public class SIRequestOutService extends javax.xml.ws.Service {

  private final static java.net.URL SIRequestOUTSERVICE_WSDL_LOCATION = null;

  public SIRequestOutService() throws java.net.MalformedURLException {
    super(SIRequestOUTSERVICE_WSDL_LOCATION, new javax.xml.namespace.QName("http://pimon2019.ru",
"SI_Request_Out_Service"));
  }
  public SIRequestOutService(java.net.URL wsdlLocation, javax.xml.namespace.QName serviceName) {
    super(wsdlLocation, serviceName);
  }

  @javax.xml.ws.WebEndpoint(name = "SI_Request_Out_Port")
  public ru.pimon2019.SIRequestOut getSI_Request_Out_Port() {
    javax.xml.namespace.QName portName = new javax.xml.namespace.QName("http://pimon2019.ru","SI_Request_Out_Port");
    return (ru.pimon2019.SIRequestOut) super.getPort(portName,ru.pimon2019.SIRequestOut.class);
  }
}
```

Client proxy (Consumer)

Теория

- Service reference client call

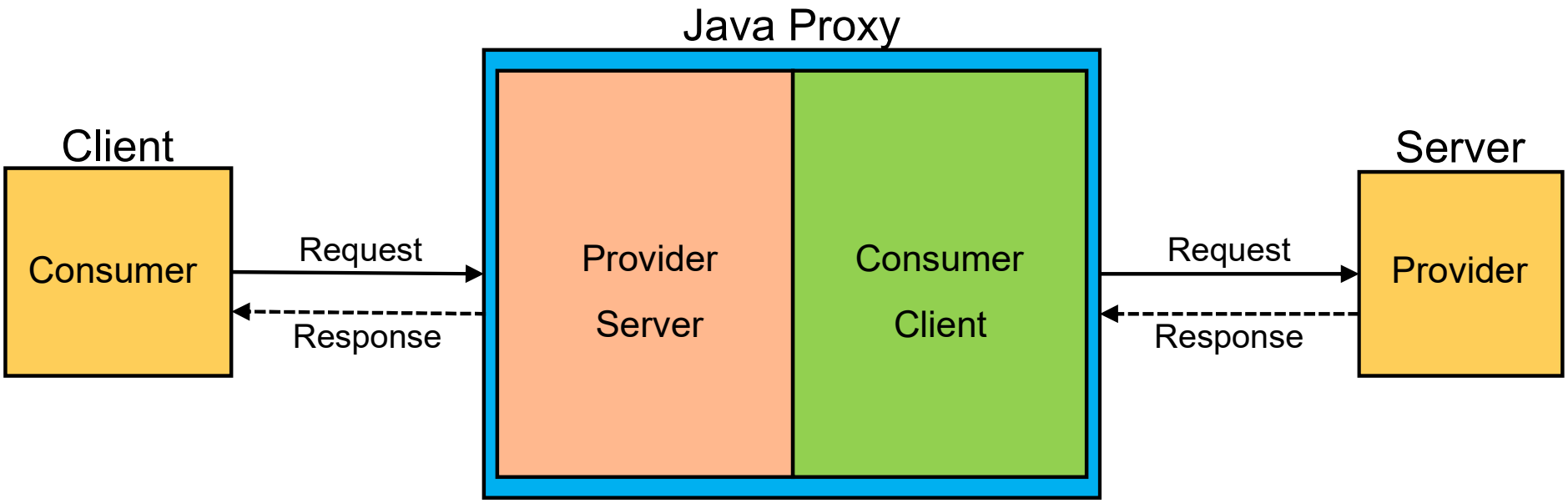
```
public class SIRequestInImplBean {

    @WebServiceRef (name="SIRequestOutService")
    SIRequestOutService siRequestOutService;

    public  void send(...){
        SIRequestOut requestOutPort = siRequestOutService.getPort(SIRequestOut.class);
        requestOutPort.send(MT_Request);
    }
}
```

Схема взаимодействия и ролей

Теория



Конфигурация SOAP-сервисов – Design Time

Теория

- Вызов сервиса - программная конфигурация через BindingProvider

Главный недостаток – отсутствие централизованной, зависимой от ландшафта настройки

```
javax.xml.ws.BindingProvider bp = (javax.xml.ws.BindingProvider)port;

Map<String,Object> context = bp.getRequestContext();

context.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY," http://pimon2019.ru");

context.put(BindingProvider.USERNAME_PROPERTY,"userName" );

context.put(BindingProvider.PASSWORD_PROPERTY,"pass");
```

- Публикация сервиса - аннотация @TransportBindingRT

Конфигурация SOAP-сервисов - Runtime

Теория

- Конфигурация в NWA – SOA
- Service Reference - добавление consumed-сервисов в Service Group в ejb-проекте
  - XI – локальные вызовы AEX
  - WS – вызовы сервисов внешних систем, в том числе внешних AEX по SOAP
- Technical Configuration: System Connection
  - Provider Systems – настройки подключения к SAP и non-SAP системам, предоставляющих сервисы. Для внешнего AEX – поиск сервисов по Service Registry, WSIL (dir/inspection.wsil), WSDL
  - Communication Profiles – профили настроек provided-сервисов
- Application communication – настройка сервисов приложения
  - Связка Service Group – Provider Systems
  - Связка Provided-сервисы - Communication Profiles
- Single Service Administration – ведение настроек сервисов (wsdl, ports, logs)

Чудеса практики

Вызов oneway-сервиса

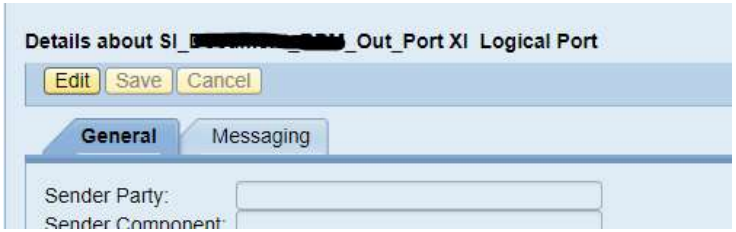
Вызов oneway-сервиса

Практика

- Создаем порт из сгенерированного метода:

```
SIRequestOut requestOutPort = siRequestOutService.getSI_Request_Out_Port();
requestOutPort.send(MT_Request);
```

- Default trace: HTTP call to IS unsupported in AEX case
- Single service administration: SI_Request_Out_Port – **XI** Logical Port



Error	Scheduled	Successful	Terminated with error	Sender Partner	Sender Component	Receiver Partner	Receiver Component	Interface	Interface Namespace
					JAVA				
0	0	0	2		INTEGRATION_ENGINE_JAVA_[redacted]			SI_[redacted]_Out	[redacted]

Вызов oneway-сервиса

Практика

- Создаем порт по умолчанию:

```
SIRequestOut requestOutPort = siRequestOutService.getPort(SIRequestOut.class);
requestOutPort.send(MT_Request);
```

- Default trace: Server error

- Single service administration: default logical port – **WS**

Вызов oneway-сервиса

Практика

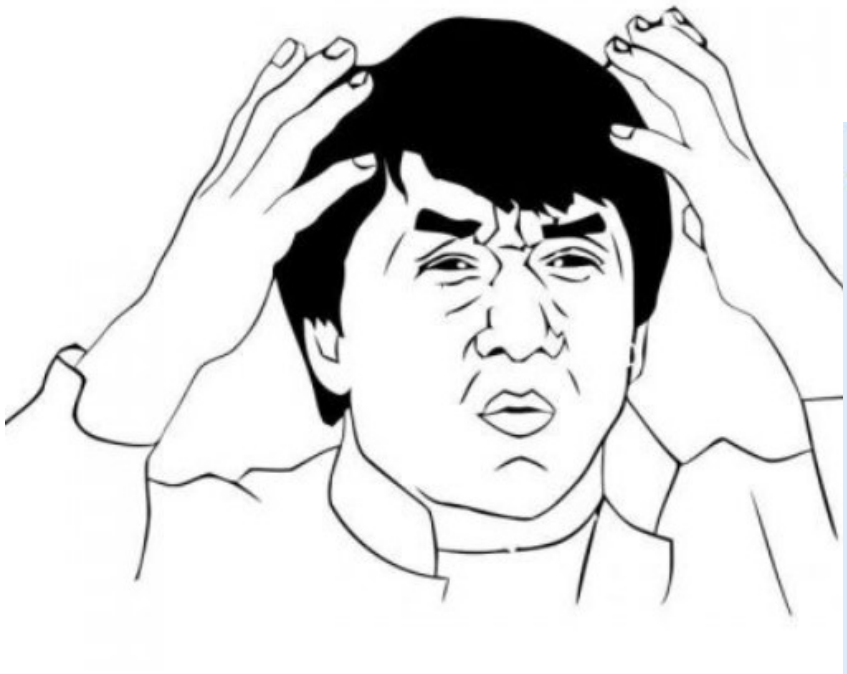
- PIMON

ıl	Terminated with error	Sender Partner	Sender Component	Receiver Partner	Re
		WS-RM			
1	0	WS-RM	{[redacted]}SI_[redacted]_Out_Serv	WS-RM	{ht
0	1	WS-RM	{[redacted]}SI_[redacted]_Out_Serv	WS-RM	{ht

Вызов oneway-сервиса

Практика

WS-RM?!



General | Security | **Messaging** | Web Service Addressing | Transport Settings

Metering of Service Calls

Data Transfer Scope: (•) Minimal Data Transfer

( ) Basic Data Transfer

( ) Enhanced Data Transfer

Transfer Protocol: (•) Transfer Using HTTP Header

( ) Transfer Using SOAP Header

Reliable Messaging

RM Protocol: WS-RM 2005/02

Exponential Backoff: ☑

Retransmission Interval in Milliseconds: 180000

Inactivity Timeout in Milliseconds: 0

lInImplBean_f4a0dc32-2cf5-4ed7-9b75-e191a2f569af_WS is automatically generated and thus cannot be edited

Search:

Вызов oneway-сервиса

Практика

- Причина ошибки Server error, полученной из внешнего AEX – вместо payload отправляется запрос на создание ws-rm sequence

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <SOAP-ENV:Header>
        <Action xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:mustUnderstand="1">http://schemas.xmlsoap.org/ws/2005/02/rm/CreateSequence</Action>
        <To xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing" xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/" SOAP-
ENV:mustUnderstand="1">http://pimon2019.ru:50000/XISOAPAdapter/MessageServlet?
senderParty=&senderService=BC_Sender&receiverParty=&receiverService=&interface=SI_Reuqest_Out&interfaceNamespace=htt
p%3A%2F%2Fpimon2019.ru</To>
        <MessageID xmlns="http://schemas.xmlsoap.org/ws/2004/08/addressing">urn:uuid:9df6d8f4-cd99-11e9-86df-
00000039fac6</MessageID>
    </SOAP-ENV:Header>
    <SOAP-ENV:Body>
        <pns:CreateSequence xmlns:pns='http://schemas.xmlsoap.org/ws/2005/02/rm'>
            <yq1:AcksTo xmlns:yq1='http://schemas.xmlsoap.org/ws/2005/02/rm'
xmlns:pns='http://schemas.xmlsoap.org/ws/2004/08/addressing'>
                <pns:Address>http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</pns:Address>
            </yq1:AcksTo>
            <pns:Expires>P0Y0M3DT0H0M0.000S</pns:Expires>
        </pns:CreateSequence>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Вызов oneway-сервиса

Практика

- Как отключить WS-RM?
  - Порт настроить нельзя
  - Аннотация @RelMessagingNW05DTOperation(enableWSRM=false) на поведение не влияет
  - Если убрать аннотацию @Oneway, то будет ошибка, что WS-RM может быть только для @Oneway

Чудеса практики
Обработка ошибок

Обработка ошибок

Практика

- Попытка отловить исключение через try-catch:

```
try{

  SIRequestOut requestOutPort = siRequestOutService.getPort(SIRequestOut.class);
  requestOutPort.send(MT_Request);

}catch(SOAPFaultException e){
    SimpleLogger.trace(Severity.ERROR, LOC, "Something goes wrong:{0}", new Object[] {e.getMessage()});
}
```

- Вызов метода будет всегда успешным

Обработка ошибок

Практика

- Попробуем через BindingProvider и свойства MessageContext получить HTTP-код ответа:

```
SIRequestOut requestOutPort = siRequestOutService.getPort(SIRequestOut.class);
BindingProvider binding = (BindingProvider)requestOutPort;

try{

  requestOutPort.send(MT_Request);

}catch(SOAPFaultException e){
    SimpleLogger.trace(Severity.ERROR, LOC, "Something goes wrong:{0}", new Object[] {e.getMessage()});
}finally{
    int responseCode = (Integer)binding.getResponseContext().get(MessageContext.HTTP_RESPONSE_CODE);
    SimpleLogger.trace(Severity.ERROR, LOC, "HTTP code:{0}", new Object[] {responseCode});
}
```

- Получение любого свойства вернет NullPointerException

Решение

Практика

Handler chain

Практика

- Цепочка обработчиков (handler chain) – JWS (JSR-181)
- Способы подключения:
  - Аннотация @HandlerChain + конфигурационный xml-файл
  - Программно через BindingProvider

```
SIRequestOut port = siRequestOutService.getPort(SIRequestOut.class);
BindingProvider binding = (BindingProvider)port;
List<Handler> handlerChain = binding.getBinding().getHandlerChain();
handlerChain.add(new CustomSOAPHandler());
binding.getBinding().setHandlerChain(handlerChain);
```

Handler chain

Практика

- Обработчики бывают двух типов:
  - **SOAPHandler** – обработчик SOAP с доступом к SOAPMessageContext и протокол-специфичными параметрам
  - **LogicalHandler** – протокол-независимый обработчик с доступом к LogicalMessageContext

Handler chain

Практика

- Методы SOAPHandler:

```
public class SampleProtocolHandler implements
  javax.xml.ws.handler.soap.SOAPHandler<SOAPMessageContext> {

    public Set<QName> getHeaders() {
      //Получение коллекции заголовков
      return null;
    }
    public boolean handleFault(SOAPMessageContext messagecontext) {
      //обработка SOAP Fault
        return true;
    }
    public boolean handleMessage(SOAPMessageContext messagecontext) {
      //обработка SOAPMessage обоих направлений

      Boolean outbound = (Boolean) messagecontext.get(MessageContext.MESSAGE_OUTBOUND_PROPERTY);
        if (outbound) {
            ...
        }
        return true;
    }
    public void close(MessageContext messagecontext) {
        //Вызывается после завершения передачи сообщения или ошибки
    }
}
```

Handler chain

Практика

- Определение Scope для свойства MessageContext в SOAPHandler:

```
public boolean handleFault(SOAPMessageContext messagecontext) {
    messageContext.put("MyCustomProperty", "houston we have a problem");
    messageContext.setScope("MyCustomProperty", MessageContext.Scope.APPLICATION);
    return true;
}
```

- Получение свойства в клиентском коде через BindingProvider:

```
Map<String, Object> responseContext = binding.getResponseContext();
String text = (String)responseContext.get("MyCustomProperty");
```

Handler chain

Практика

- Почему это работает?
  - есть предположение, что подключение handler chain сообщает runtime-у, что мы самостоятельно берем на себя обработку MEP и сами решаем, что делать с ответами и ошибками, даже если их может не быть.
  - это тема дальнейшего исследования

Сценарии применения

Практика

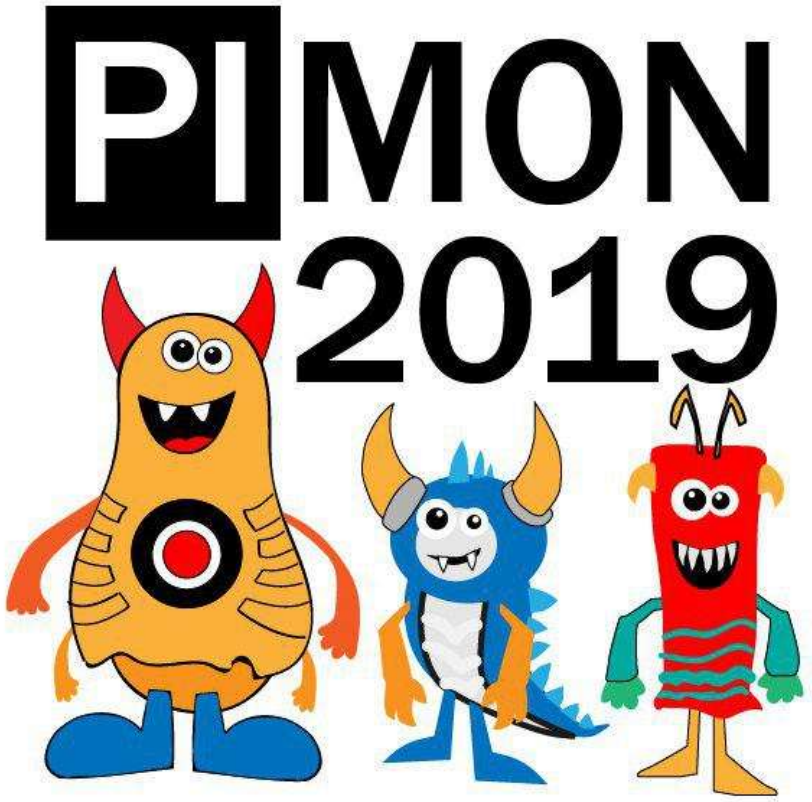
- Вызовы внешних SOAP-сервисов из Java Proxy с расширенной обработкой (логирование, безопасность, модификация)
- Связка BPM + JavaProxy для pre/post обработки сообщений + внешний AEX
- Организация мостов с внешним AEX с расширенной обработкой
  - комбинация паттернов в *consumer-provider* может быть любой (RR-RR, RR-Oneway, Oneway-RR, Oneway-Oneway), но с ограничениями на расширенную обработку со стороны provider

Выводы

Практика

- Netweaver Java Proxy:
  - стандартные инструменты конфигурации сервисов приложения – в целом это удобно
  - oneway-сервисы – это возможно
  - основные паттерны SOAP-сервисов – есть расширенная обработка
- SAP Netweaver Java AS:
  - это весело и интересно
  - разработку вести порой сложно, но можно
  - информации и документации нет или она давно устарела, нужно расширять поиски
  - можно найти инструменты для решения нестандартных задач

Спасибо!



Partner logo

THE BEST RUN SAP