

TECHNOLOGIE INTERNETOWE

INSTALACJA USŁUG SIECIOWYCH Z WYKORZYSTANIEM WSDD, DEFINIOWANIE PROCEDUR OBSŁUGI, PRZETWARZANIE WIADOMOŚCI W SYSTEMIE AXIS, SERIALIZACJA

LABORATORIUM 6

1. WSTĘP i KONFIGURACJA

Tematem niniejszego laboratorium jest konfiguracja zaawansowanych elementów serwera AXIS oraz instalacja i uruchamianie usług sieciowych z wykorzystaniem deskryptorów WSDD. Przed przystąpieniem do niniejszego laboratorium, należy zapoznać się z następującą literaturą:

1. Instrukcje do laboratorium 4 oraz 5.
2. User's Guide systemu AXIS. <http://ws.apache.org/axis/java/user-guide.html>
3. Architecture Guide systemu AXIS. <http://ws.apache.org/axis/java/architecture-guide.html>
4. Reference Guide systemu AXIS. <http://ws.apache.org/axis/java/reference.html>
5. Przykłady dostarczone z systemem AXIS. Katalog `samples/userguide/example*`.

Kompilacja i uruchomienie przykładów wyglądają następująco:

1. Kompilacja klienta, usługi sieciowej oraz pliku procedur(y) obsługi:
`javac ...*.java`
np.:
`javac samples/userguide/example4/*.java`
2. Umieszczenie kodu na serwerze:
`cp ../axis-1_1beta/samples/userguide/example4/*.class webapps/axis/WEB-INF/classes/samples/userguide/example4/`
3. Uruchomienie serwera Tomcat z zainstalowanym AXISem: `startup.sh` w katalogu `bin` katalogu głównego serwera Tomcat. W katalogu, w którym uruchamiany jest serwer, znajdą się ewentualne pliki zapisywane przez procedurę obsługi.
4. Zainstalowanie procedury obsługi wraz z usługą:
`java org.apache.axis.client.AdminClient samples/userguide/example4/deploy.wsdd`
5. Uruchomienie klienta:
`java samples/userguide/example4/Client`

2. PRZETWARZANIE WIADOMOŚCI, PROCEDURY OBSŁUGI

AXIS jest narzędziem, które służy przetwarzaniu wiadomości. Ścieżka przetwarzania wiadomości na serwerze wygląda następująco:

1. `TransportListener` nasłuchuje na przyjsie wiadomości, następnie tworzy obiekt `MessageContext`, który jest przekazywany do przetwarzania serwerowi AXIS. Nasłuchiwanie może odbywać się na porcie 80, ale może również polegać na

sprawdzeniu pojawienia się pliku o określonej nazwie w katalogu i przekazaniu jego zawartości do przetwarzania przez serwer.

2. W ramach AXIS Engine, wiadomość może być przetwarzana przez łańcuchy procedur obsługi, zarówno przed wywołaniem właściwej metody usługi sieciowej jak i po.

Etapy, na których procedury obsługi (handlers) mogą być wywołane są następujące:

- a. Warstwa transport – dla specyficznego protokołu
- b. Global – dla wszystkich wiadomości
- c. Service – dla konkretnej usługi sieciowej.
- d. Wywołanie właściwej metody usługi sieciowej
- e. Service
- f. Global
- g. Warstwa transport

Odpowiednia konfiguracja usługi sieciowej wraz z procedurami obsługi, które mogą być zainstalowane na poszczególnych etapach, opisana jest w:

1. dokumentacji AXIS. <http://ws.apache.org/axis/java/reference.html>
2. przykładzie w instrukcji użytkownika AXIS. <http://ws.apache.org/axis/java/user-guide.html#CustomDeploymentIntroducingWSDD>

Instalacja usługi sieciowej z dodatkowymi procedurami obsługi, wywoływanymi przed lub po wywołaniu właściwej metody usługi sieciowej, polega na zdefiniowaniu odpowiedniego deskryptora WSDD (Web Service Deployment Descriptor), który pozwala na zdefiniowanie:

1. usługi sieciowej – element service wraz z:
 - a. nazwą,
 - b. klasą zawierającą implementację: `<parameter name="className" value="..."/>`
 - c. listą metod, które można wywołać w usłudze sieciowej `<parameter name="allowedMethods" value="*" />` - wszystkie metody
 - d. procedurami obsługi (handlers) lub łańcuchami (ciągi procedur obsługi), które mogą być wywołane dla danej usługi sieciowej przed lub po wywołaniu właściwej metody usługi sieciowej:
 - i. element requestFlow – przed wywołaniem usługi
 - ii. element responseFlow – po wywołaniu usługi

Elementy te, zdefiniowane wewnątrz elementu service tyczą się danej usługi sieciowej, mogą być również zdefiniowane globalnie (element GlobalConfiguration, <http://ws.apache.org/axis/java/reference.html>), dotyczą wówczas wszystkich wywołań

możliwe jest również użycie elementów requestFlow oraz responseFlow dla warstwy transport – element transport (<http://ws.apache.org/axis/java/reference.html>).

Elementy requestFlow oraz responseFlow mogą zawierać łańcuchy oraz procedury obsługi, wywołane zostaną zgodnie z kolejnością, w której zostały wyspecyfikowane. Procedury obsługi lub łańcuchy przywoływane są po nazwie, zdefiniowane mogą być poza elementem service. Procedury obsługi mogą czytać argumenty w postaci pary nazwa, wartość, wyspecyfikowane w deskrypcji WSDD. Przykład:

```
<handler name="acceptreject" type="java:AcceptReject">
  <parameter name="deadline" value="..." />
</handler>
```

W powyższym przykładzie, który może być fragmentem usługi sieciowej, która przyjmuje artykuły na konferencję, ww. procedura obsługi może być pierwszym etapem na drodze do sprawdzenia poprawności publikacji. Mianowicie, procedura obsługi może sprawdzać czy artykuł, który zostaje złożony w wywołaniu usługi sieciowej, został złożony przed czy po wyznaczonym terminie. Termin ten zdefiniowany jest w parametrze o nazwie deadline, który kod procedury obsługi może odczytać. Fragment kodu odpowiedniej procedury obsługi mógłby wyglądać następująco:

```
public class ConferenceDeadlineHandler extends BasicHandler {
    public void invoke(MessageContext msgContext) throws AxisFault
    {
        try {

            String filename = (String)getOption("deadline");
            ...
            if (deadline<currentTime) reject();

        } catch (Exception e) {
            throw AxisFault.makeFault(e);
        }
    }
}
```

Dane o żądaniu, w szczególności informacje czy procedura obsługi znajduje się za właściwą metodą usługi sieciowej czy po, dostęp do właściwych danych żądania itp. można uzyskać poprzez obiekt MessageContext (<http://ws.apache.org/axis/java/apiDocs/org/apache/axis/MessageContext.html>).

Przykładowo, poniższy kod sprawdza czy procedura znajduje się przed czy po wywołaniu właściwej metody usługi sieciowej:

```
if (msgContext.getPastPivot())
    writer.println("yes");
```

Wiele informacji uzyskać można poprzez wywołanie metody getProperty obiektu MessageContext. Możliwe do wykorzystania nazwy atrybutów znaleźć można np. w

<http://ws.apache.org/axis/java/apiDocs/org/apache/axis/transport/http/HTTPConstants.html>.

Należy pamiętać o dołączeniu właściwej linijki import do kodu procedury obsługi.

Referencję do obiektu MessageContext można pobrać z argumentu metody invoke procedury obsługi bądź też wywołując statyczną metodę MessageContext.getCurrentContext().

W ten sposób, z wykorzystaniem procedur obsługi, można zaimplementować przetwarzanie zgłoszenia publikacji:

1. Procedura obsługi – sprawdzanie terminu zgłoszenia
2. Procedura obsługi – sprawdzanie długości zgłoszenia itp.
3. Właściwa metoda usługi sieciowej – zapis publikacji do bazy danych
4. procedura obsługi – wysłanie emaila do zgłaszającego publikację, iż została przyjęta do recenzji.

Implementacja taka pozwala na uniknięcie błędów poprzez implementację mniejszych modułów, polepsza konfigurowalność aplikacji poprzez definiowanie parametrów dla procedur obsługi (serwis może być wykorzystany na różnych konferencjach z różnymi parametrami jak data zgłoszenia, maksymalny rozmiar itp.) oraz możliwość wykorzystania pojedynczych procedur w różnych usługach sieciowych (np. procedura sprawdzająca termin zgłoszenia).

Procedura obsługi może również wygenerować wyjątek. Dla przykładu, dzielenie przez 0, a więc kod procedury obsługi:

```
int a=2/0;

writer.close();
} catch (Exception e) {
    throw AxisFault.makeFault(e);
}
```

pokaże po stronie klienta:

```
java samples/userguide/example4/Client - Mapping Exception to AxisFault
AxisFault
faultCode: {http://xml.apache.org/axis/}Server.userException
faultString: java.lang.ArithmeticException: / by zero
faultActor: null
faultDetail:
  stackTrace: java.lang.ArithmeticException: / by zero
  at samples.userguide.example4.LogHandler.invoke(LogHandler.java:105)
```

Artykuł dostępny pod adresem <http://www-128.ibm.com/developerworks/xml/library/x-tipsoap.html> pokazuje w jaki sposób można użyć nagłówek SOAP w celu implementacji priorytetów wywołań usług sieciowych i ewentualnego opóźnienia usług o niższym priorytecie. Klient, przy wywołaniu usługi sieciowej określa jej priorytet, jest on następnie odczytywany przez procedurę obsługi, która wprowadza dodatkowe opóźnienie w zależności od zadanego priorytetu.

3. STYLE PRZETWARZANIA WIADOMOŚCI W AXIS

AXIS pozwala na przetwarzanie wiadomości w różnych „stylach”, z których najpopularniejszym jest RPC, polegający na wykorzystaniu usług sieciowych, protokołów SOAP np. z http jako implementacji zdalnegowołania procedur. Z tego punktu widzenia, usługi sieciowe są w dużej mierze równoważne technologiom CORBA itp., z wyjątkiem użycia innych protokołów i szczegółów implementacyjnych.

Z drugiej strony, oprócz stylów dokument i wrapped (<http://ws.apache.org/axis/java/user-guide.html#ServiceStylesRPCDocumentWrappedAndMessage>), AXIS umożliwia przetwarzanie wiadomości na poziomie protokołu SOAP, metoda usługi sieciowej może mieć więc dostęp do poszczególnych elementów wiadomości w formacie XML. Metody obsługujące po stronie serwera mogą mieć następujące sygnatury (<http://ws.apache.org/axis/java/user-guide.html#ServiceStylesRPCDocumentWrappedAndMessage>):

```
public Element [] method(Element [] bodies);
public SOAPBodyElement [] method (SOAPBodyElement [] bodies);
public Document method(Document body);
public void method(SOAPEnvelope req, SOAPEnvelope resp);
```

AXIS można więc wykorzystać jako serwer przetwarzający dokumenty XML.

4. SERIALIZACJA

AXIS umożliwia automatyczną serializację i deserializację, a więc i przezroczyste dla programisty przesyłanie typów prostych, ale również klas, które zgodnie z wymaganiami dla JavaBeans zawierają metody set i get. Dla takich klas, wystarczy w deskrypcji WSDO w wyspecyfikować nazwę klasy:

```
<beanMapping qname="ns:local" xmlns:ns="someNamespace"
  languageSpecificType="java:nazwaklasy"/>
```

Dla klas, które nie są zgodne z wymaganiami dla JavaBeans, należy dostarczyć kod serializujący i deserializujący kod danej klasy oraz zapisać powiązanie w elemencie deskryptora WSDO w sposób następujący:

```
<typeMapping qname="ns:local" xmlns:ns="someNamespace"
  languageSpecificType="java:nazwaklasy"
```

```
serializer="serializator"  
deserializer="deserializator"  
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
```

5. ASYNCHRONICZNE WOŁANIE METOD W AXIS2

W ramach nowych funkcji systemu AXIS, należy zwrócić uwagę na asynchroniczne wołania metod na serwerze, które dostępne są w systemie AXIS2 (<http://ws.apache.org/axis2/userguide.html>). Wystarczy wówczas zarejestrować funkcję callback, która wywołana zostanie asynchronicznie (umożliwiając klientowi kontynuowanie po wywołaniu metody usługi sieciowej) przy zakończeniu obsługi metody.