Sistemas Distribuídos

2018/2019

Aula 3 - WebServices SOAP ¶

Web Services SOAP - resumo

Um web service é um sistema especificamente desenhado para suportar a **interoperação** e a **interação máquina-para-máquina** sobre uma rede.

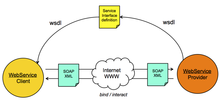
A interface de um WebService SOAP é descrita num formato que é processável por uma máquina: **WSDL**.

Através do documento WSDL de um WebService é possível implementar clientes compatíveis com esse serviço, na maioria das linguagems correntes.

A interação entre o cliente e o servidor, processa-se tipicamente usando HTTP ou HTTPS (eventualmente SMTP).

As invocações são suportadas por mensagens do protocolo SOAP (descrito em XML). Consequentemente, os dados transferidos nas invocações (parâmetros e resultado das operações) são também convertidos para XML.

Arquitetura



Desenvolvimento de WebServices em Java

É possível obter o documento WSDL a partir de uma instância do WebService já em execução.

Por essa razão, em Java, torna-se mais simples implementar o serviço de raiz na própria linguagem Java, sem antes passar pela criação do ficheiro WSDL.

Criação do Servidor

- Começa pela criação da interface e a sua implementação na linguagem Java;
  - Utilizam-se **anotações** para indicar: os métodos disponibilizados remotamente; a classe que implementa o serviço propriamente dito; as excepções que poderão ser lançadas.
- Instancia-se:
  1. Criando um **endpoint SOAP** com a ligação à classe que implementa o serviço.
  2. **Publica-se** o endpoint SOAP num servidor HTTP ou HTTPS.

Cliente

- **Com base no URL** de uma instância do serviço em execução:
  1. O cliente obtém um objeto ***"proxy"*** que implementa a interface do serviço localmente;
  2. As operações invocadas localmente pelo cliente traduzem-se em invocações no servidor.
- Havendo múltiplas instâncias do serviço, o cliente pode aceder a cada uma delas através do URL respetivo, criando um "proxy" para cada uma delas.

Outros aspetos práticos

A passagem de parâmetros e o retorno dos métodos é feita **por cópia dos valores** (não há passagem de referências)

O tipo dos parâmetros e do retorno poderá ter sofrer adaptações (em particular se cliente aceder um serviço SOAP implementado noutra linguagem).

- Por exemplo: `String[] -> List<String>`

Exemplo - MediaStorage

```
@WebService(serviceName=SoapMedia.NAME, targetNamespace=SoapMedia.NAMESPACE, endpointInterface=SoapMedia.INTERFACE)
public interface SoapMedia {

    static final String NAME = "media";
    static final String NAMESPACE = "http://sd2019";
    static final String INTERFACE = "microgram.api.soap.SoapMedia";

    @WebMethod
    String upload( byte[] bytes) throws MicrogramException;

    @WebMethod
    byte[] download(String id) throws MicrogramException;
}
```

```
@WebFault
class MicrogramException extends Exception {

    private static final long serialVersionUID = 1L;

    public MicrogramException() {
        super("");
    }

    public MicrogramException(String errorMessage ) {
        super(errorMessage);
    }
}
```

Observação: A utilização de SOAP para implementar a funcionalidade de transferência e disponibilização de dados binários, não é a escolha mais apropriada.
É possível utilizar um WebService SOAP para transferir bytes em bruto, mas a codificação XML, envolvida na transferência, torna o processo ineficiente.

Adicionalmente, ao contrário dos WebServices REST, um WebService SOAP não pode ser acedido por um *browser* ou por um simples cliente HTTP. Ou seja, os WebServices SOAP são para ser consumidos por clientes SOAP.

A versão SOAP do serviço MediaStorage serve apenas de pretexto para ilustrar o desenvolvimento em Java deste tipo de WebServices.

MediaStorage - Servidor

Implementação do serviço

```
...
@WebService(serviceName=SoapMedia.NAME, targetNamespace=SoapMedia.NAMESPACE, endpointInterface=SoapMedia.INTERFACE)
public class MediaWebService implements SoapMedia {
    ...
    @Override
    public String upload(byte[] bytes) throws MicrogramException {
        try {
            String id = Hash.of(bytes);
            File filename = new File(ROOT_DIR + id + MEDIA_EXTENSION);
            if( filename.exists() )
                throw new MicrogramException("Conflict...");
            Files.write(filename.toPath(), bytes);
            return id;
        } catch( IOException x  ) {
            x.printStackTrace();
            throw new MicrogramException("Internal Error..." + x.getMessage());
        }
    }
    ...
}
```

Nota: A classe que implementa o serviço precisa de repetir a anotação @WebService As demais anotações @WebMethod, etc. podem ser omitidas

Instanciação do serviço

```
public static final int PORT = 7777;
public static String SOAP_BASE_PATH = "/soap/media";

// Create an HTTP server, accepting requests at PORT (from all local interfaces)
HttpServer server = HttpServer.create(new InetSocketAddress("0.0.0.0", PORT), 0);

// Create the SOAP Endpoint
Endpoint soapEndpoint = Endpoint.create(new MediaWebService());

// Publish the SOAP webservice, under the "http://<ip>:<port>/soap"
soapEndpoint.publish( server.createContext(SOAP_BASE_PATH) );

// Provide an executor to create threads as needed...
server.setExecutor( Executors.newCachedThreadPool() );

// Start Serving Requests: both SOAP Requests
server.start();
```

MediaStorage - Cliente (Upload)

```
String serverUrl = "http://localhost:7777/soap/media"

QName QNAME = new QName(SoapMedia.NAMESPACE, SoapMedia.NAME);
Service service = Service.create( new URL(serverUrl + WSDL), QNAME);
SoapMedia media = service.getPort( microgram.api.soap.SoapMedia.class );

try {
    byte[] bytes = Files.readAllBytes( EARTH.toPath() );
    String uri = media.upload( bytes );
    Log.info("Upload completed: " + uri );
} catch( MicrogramException x ) {
    Log.info("Upload failed, reason: "  + x.getMessage());
}
```

Exercício

O projeto P3-Soap/SD2019-Labs-P3, incluído no repositório, contém o exemplo incompleto do serviço MediaStorage na versão SOAP.

Analise o código fornecido. Para já, ignore os pacotes `rest` e `shared` fornecidos no projeto.

1. Complete a versão SOAP do serviço MediaStorage, implementando a operação `download`;
2. Teste o serviço SOAP MediaStorage;
   - Use o código do cliente fornecido para fazer `upload` de uma imagem;
   - Desenvolva um cliente para testar a operação de *download*;

Compare a implementação das operações `upload` e `download` na versão SOAP, desta semana, e na versão REST feita na aula anterior.

É provável que muito código esteja duplicado. O próximo exercício irá resolver este problema.

Exercício Suplementar

Quando um mesmo serviço é oferecido nas versões REST e SOAP pode ser possível partilhar código entre elas e evitar implementar a lógica do serviço em duplicado.

No projeto fornecido poderá encontrar a versão REST do serviço MediaStorage, implementada à custa da classe JavaMedia.

Nesta versão REST, a leitura e escrita dos dados de e para ficheiro foi movida para a classe JavaMedia.

O código específico da versão REST limita-se, agora, a usar o resultado da operação correspondente na classe JavaMedia, para gerar a resposta ou a traduzir o erro retornado na excepção `WebApplicationException` apropriada.

- Implemente a versão SOAP do serviço MediaStorage tirando partido da classe `JavaMedia`.