Theory Exercise 4

Tarek Auel, Tiange Hu, Benedikt Christoph Naumann, Markus Schanz

Task 1: WSDL elements

The below WSDL definition was auto-generated by java. It may requires some fine tuning, as some attributes are optional.

```
<?xml version="1.0" encoding="UTF-8"?>
<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://tk.informatik.tu-darmstadt.de"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/"
    targetNamespace="http://tk.informatik.tu-darmstadt.de"
    name="LocationServiceImplService">

    <types>
        <xsd:schema>
            <xsd:complexType name="Location">
                <xsd:sequence>
                    <xsd:element name="x" type="xsd:int"/>
                    <xsd:element name="y" type="xsd:int"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:schema>
    </types>

    <message name="getLocation">
        <part name="id" type="xsd:int"></part>
    </message>

    <message name="getLocationResponse">
        <part name="return" type="tns:location"></part>
    </message>

    <portType name="LocationService">
        <operation name="getLocation">
            <input
                wsam:Action="http://tk.informatik.tu-darmstadt.de/LocationService/getLocationRequest"
                message="tns:getLocation">
            </input>

            <output
                wsam:Action="http://tk.informatik.tu-darmstadt.de/LocationService/getLocationResponse"
                message="tns:getLocationResponse">
            </output>
        </operation>
    </portType>

    <binding name="LocationServicePortBinding" type="tns:LocationService">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="rpc"></soap:binding>
        <operation name="getLocation">
            <soap:operation soapAction=""></soap:operation>
            <input>
                <soap:body use="literal" namespace="http://tk.informatik.tu-darmstadt.de"></soap:body>
            </input>

            <output>
                <soap:body use="literal" namespace="http://tk.informatik.tu-darmstadt.de"></soap:body>
            </output>
```

```
        </operation>
    </binding>

    <service name="LocationServiceImplService">
        <port name="LocationServicePort" binding="tns:LocationServicePortBinding">
            <soap:address location="http://localhost:8090/"></soap:address>
        </port>
    </service>
</definitions>
```

Task 2: Servlet Container

	REST Web Service	SOAP Web Service
Protokoll	RESTful Webservices bauen prinzipbedingt auf dem HTTP Protokoll auf.	SOAP wird ebenfalls hauptsächlich über HTTP verwendet, baut allerdings auf XML auf und ist damit nicht an ein Protokoll gebunden. Neben HTTP können z.B. TCP, SMTP oder JMS benutzt werden.
Typing	Keine statisch Typisierung. Parameter werden in der URL übergeben, jedoch ohne festen Typ.	SOAP unterstüzt sowohl primitive Typen, wie z.B. Zahlen und Strings, als auch komplexe (=zusammengesetzte) Typen. Üblicherweise nutzen Dienste die Web Services Description Language (WSDL) um dem Client alle notwendigen Informationen über vorhandene komplexen Typen sowie vorhandene Endpunkte zukommen zu lassen. Diese Information kann dann zum Beispiel für eine automatische Codegenerierung genutzt werden.
Data Format	Bei Anfragen dient meist die URL als einziges Datenformat, Antworten werden üblicherweise als JSON oder XML gegeben, wobei der Nutzer über den HTTP Accept Header ggf. selber bestimmen kann welches Datenformat er als Antwort erwartet. Jedoch können auch die üblichen HTTP Mechanismen genutzt werden um komplexere Daten zu übertragen oder zu empfangen (z.B. beliebige binärdaten über HTTP POST).	Anfragen werden in SOAP als XML Dokumente repräsentiert. Das Format wird durch SOAP vorgegeben. Dies erlaubt es auch komplexe Daten in der Anfrage zu verschicken, bzw. zurückzugeben.
Addressing	Die Adressierung folgt durch die Nutzung von Domains (DNS Protokoll). Klassischerweise werden die verschiedenen Endpunkte durch zusammengesetzte URL's angesprochen ("Resourcen"). Eine Resource ist dabei überlicherweise eine Auflistung eines Typs (z.B. Liste von Nutzern) oder eine konkrete Instanz des Typs (z.B. ein bestimmter Nutzer)	Im Falle von SOAP over HTTP dient auch hier das DNS Protokoll. Durch XML Namespacing werden einzelne Methoden, komplexe Typen usw. identifieziert. Die konkret erreichbaren Endpunkte sind durch die WSDL Datei beschrieben, die von dem Dienst bereitgestellt wird.
State	Kein State.	I.d.R. auch kein State.