



Service Engineering

Übung 2a – Spezifikation und Nutzung von Web-APIs (Services)



Aufgabenstellung



Ziele der Übung

- Spezifikation eines eigenen Serviceangebots Web-API's
- Test eines vorgegebenen Web-API-Clients
- Anpassung des Web-API-Clients auf eigene Zugriffsszenarien
- Recherche und Verwendung von HTTP-Monitoren (API-Gateway)



Aufgabenstellungen - 1

- Erzeugen einer WSDL- oder Swagger-Spezifikation
 - Identifikation der Ziele einer entsprechenden Spezifikation
 - Kriterienbasierte Auswahl eines Werkzeugs zur WSDL/Swagger-Erstellung
 - Analyse verfügbarer kommerzieller Produkte
 - Analyse verfügbarer OpenSource-Produkte
 - Erstellen einer WSDL/Swagger-Spezifikation
(Beispiel mit mind. 4 Operationen/WSDL bzw. 4 Objekte/Swagger)
- Vergleich der Spezifikation mit WSDL und Swagger (OpenAPI)
 - Beschreibung von Vor- und Nachteilen (Kriterienbasiert)
 - Fehlende Elemente zur Servicebeschreibung?

Aufgabenstellungen - 2

- Testen Sie die vorgegebenen Java-Clients (SOAP und REST)
- Entwickeln Sie einen Java-Client der das SOAP- oder JSON-Protokoll direkt nutzt und auf Funktionen selbst gewählter Web-Services zugreift. → ggf. Java-Bibliotheken verwenden!
 - Auswahl von 2 Web Services (mind. je 2 Operationen testen)
 - SOAP/JSON-Schnittstelle des Serviceangebots muss verfügbar sein!!
 - Anpassung des vorgegeben Lösungsmusters oder eigene Lösung
- Identifikation potentieller Problembereiche

Aufgabenstellungen - 3

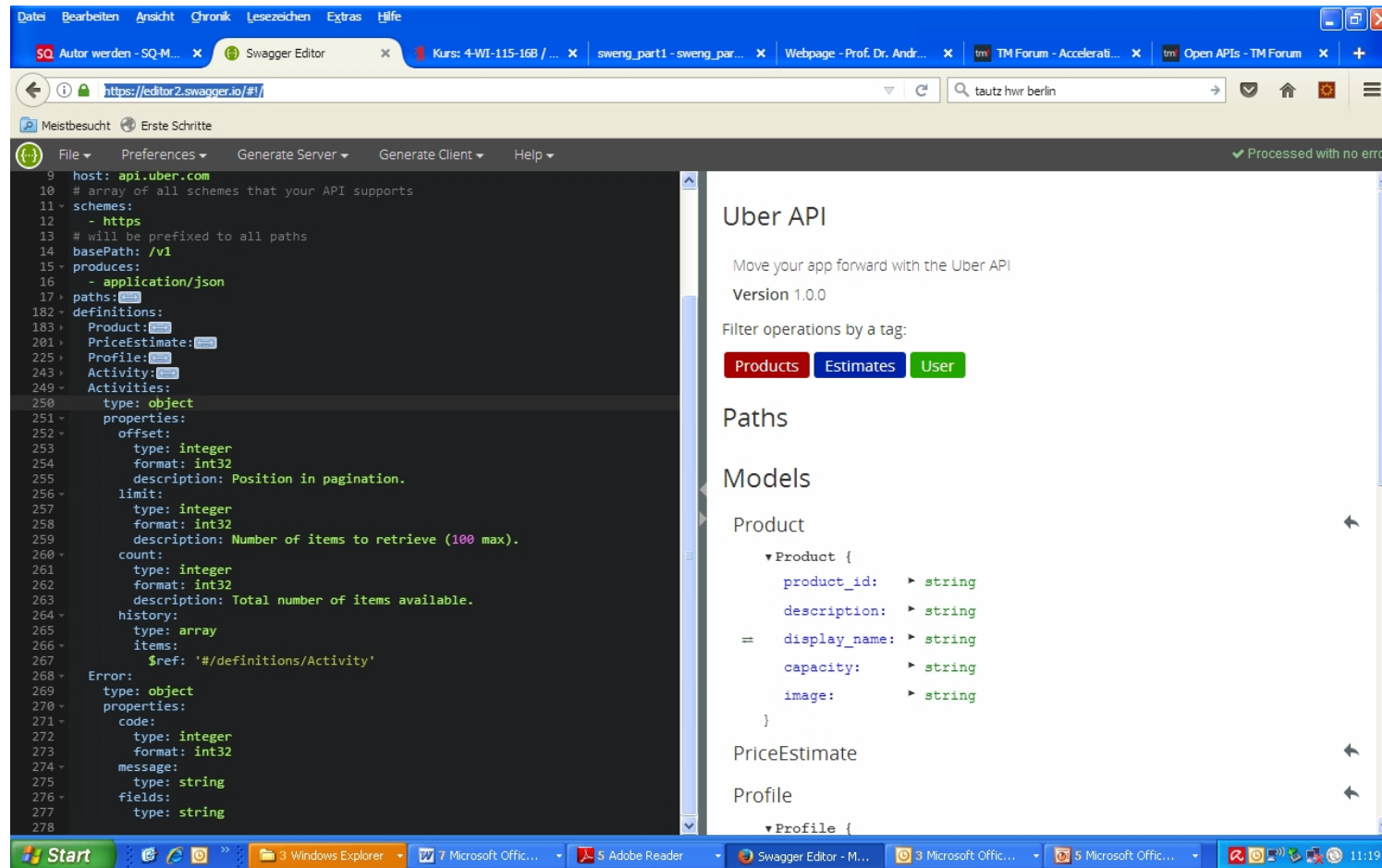
- Recherchieren Sie nach Möglichkeiten zur SOAP/XML und JSON-Überwachung und geben Sie verschiedene Lösungsalternativen an.
- Verwendung des ausgewählten Monitors (z.B. Membrane)
 - Analyse des bereitgestellten Funktionsumfangs
 - Konfiguration für ein eigenes Beispiel
 - Protokollierung der ein- und ausgehenden Nachrichten
 - Bewertung des Einsatzpotentials dieses Werkzeugs



Beispiel Swagger-Editor



Swagger-Editor (Cloud-Lösung)



Quelle: <https://editor2.swagger.io/#/>



Swagger-Editor (hier Uber-API)

```
File Preferences Generate Server Generate Client Help
1 # this is an example of the Uber API
2 # as a demonstration of an API spec in YAML
3 swagger: '2.0'
4 info:
5   title: Uber API
6   description: Move your app forward with the Uber API
7   version: "1.0.0"
8   # the domain of the service
9   host: api.uber.com
10  # array of all schemes that your API supports
11  schemes:
12    - https
13    # will be prefixed to all paths
14  basePath: /v1
15  produces:
16    - application/json
17  paths:
18    /products:
19      get:
20        summary: Product Types
21        description: |
22          The Products endpoint returns information about the *Uber* products
23          offered at a given location. The response includes the display name
24          and other details about each product, and lists the products in the
25          proper display order.
26        parameters:
27          - name: latitude
28            in: query
29            description: Latitude component of location.
30            required: true
31            type: number
32            format: double
33          - name: longitude
34        tags:
35        responses:
36      /estimates/price:
37      /estimates/time:
38      /me:
39      /history:
40  definitions:
41    Product:
42    PriceEstimate:
43    Profile:
```

Quelle: <https://editor2.swagger.io/#/>

Beispiel eines SOAP-Clients



Serviceangebot im Internet

Veröffentlichungen / Bankleitzahlen BLZ Web Service

[Home](#)
[Schulung](#)
[Beratung](#)
[Entwicklung](#)
[Veröffentlichungen](#)
BLZ Service
[CSV2XML](#)
[REST Demo](#)
[Shibboleth](#)
[Shibboleth \(engl.\)](#)
[Web Services](#)
[Bookmarks](#)
[Yahoo Webrank](#)
[WS-Specifications](#)
[Web Services Support](#)
[Open Source](#)
[Compression Filter](#)
[SOAP Compression](#)
[Compression How-To](#)
[SOA vs. EAI & ESB](#)

[Downloads](#)
[Kontakt](#)
[Impressum](#)
[Jobs](#)

(0228) 5552576-0
info@thomas-bayer.com

Copyright (c) 2004 - 2007 Thomas Bayer
Albertus-Magnus-Str. 40, 53177 Bonn, Tel. +49 (228) 5552576-0

Bankleitzahlen Web Service

Dieser Web Service ermittelt zu einer Bankleitzahl den Namen des Kreditinstitutes, die Postleitzahl und den Ort. Falls eine Bankleitzahl nicht gefunden werden kann, wird eine Fehlermeldung zurückgegeben.

Quelle der Daten ist die [Deutsche Bundesbank](#).

WSDL: <http://www.thomas-bayer.com/axis2/services/BLZService?wsdl>

Nutzung: Die Nutzung ist kostenlos und auf eigenes Risiko. Für die Verfügbarkeit des Services sowie die Richtigkeit der Daten wird keine Haftung übernommen.

Quelle des Web Service: Bayer, T.: <http://www.thomas-bayer.com/soap/blz-web-service.htm>

Beispiel eines SOAP-Clients - 1

```
import java.io.*; import java.net.*;

public class TestClient {

    private static final int BUFFER = 50;
    public static void main(String[] argv) throws Exception{
        String request =
            "<?xml version='1.0' encoding='UTF-8'?>" +
            "<soapenv:Envelope " +
            "xmlns:soapenv='http://schemas.xmlsoap.org/soap/envelope/' " +
            "xmlns:xsd='http://www.w3.org/2001/XMLSchema' " +
            "xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'> " +
            "<soapenv:Body> " +
            "<tns:getBank xmlns:tns='http://thomas-bayer.com/blz/'>"
            + "<tns:blz>10090000</tns:blz></tns:getBank>"
            + "</soapenv:Body>" +
            "</soapenv:Envelope>";
```

Beispiel eines SOAP-Clients - 2

```
// Hier muss die URL des WS-Angebots eingetragen werden
URL url = new URL("http://www.thomas-
    bayer.com/axis2/services/BLZService?wsdl");
// Verbindungobjekt erstellen und konfigurieren
URLConnection con = (URLConnection) url.openConnection();
con.setDoOutput(true);
con.setUseCaches(false);
con.setRequestProperty("Accept", "text/xml");
con.setRequestProperty("Connection", "keep-alive");
con.setRequestProperty("Content-Type", "text/xml");
con.setRequestProperty(
    "Content-length",
    Integer.toString(request.length()));
con.setRequestProperty("SOAPAction", "\"\"");
```

Beispiel eines SOAP-Clients - 3

```
// Vorbereitung und Ausgabe der XML-Datei zum Web Service
OutputStream out = con.getOutputStream();
out.write(request.getBytes()); out.flush();
// Verarbeitung und Ausgabe der Antwort XML
StringBuffer response = new StringBuffer(BUFFER);
InputStreamReader input = new InputStreamReader(con.getInputStream(),
    "UTF-8");
char buff[] = new char[BUFFER];
int n;
while (( n = input.read(buff, 0, BUFFER - 1)) > 0) {
    response.append(buff, 0, n);
}
out.close(); input.close();
System.out.println( response.toString() );
}
```



Ergebnisausgabe

```
<?xml version='1.0' encoding='UTF-8'?>

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

  <soapenv:Body>

    <ns1:getBankResponse xmlns:ns1="http://thomas-bayer.com/blz/">

      <ns1:details>

        <ns1:bezeichnung>Berliner Volksbank</ns1:bezeichnung>

        <ns1:bic>BEVODEBBXXX</ns1:bic><ns1:ort>Berlin</ns1:ort>

        <ns1:plz>10892</ns1:plz></ns1:details></ns1:getBankResponse>

      </soapenv:Body>

    </soapenv:Envelope>
```



Beispiel eines REST-Clients

Beispiel eines REST-Clients

```
import java.io.*;
import java.net.*;

public class RestClient {

    public static void main(String[] args) throws IOException {

        //Adresse der Web-API konfigurieren - JSON
        String urlT1 = "http://api.geonames.org/citiesJSON?north=44.1&south=-9.9&east=-22.4&west=55.2&lang=de&username=demo";
        //Adresse der Web-API konfigurieren - XML
        String urlT2 = "http://api.geonames.org/cities?north=44.1&south=-9.9&east=-22.4&west=55.2&username=demo";
        URL url = new URL(urlT2);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");
        conn.setRequestProperty("accept", "application/json");
        conn.setRequestProperty("accept", "text/xml");

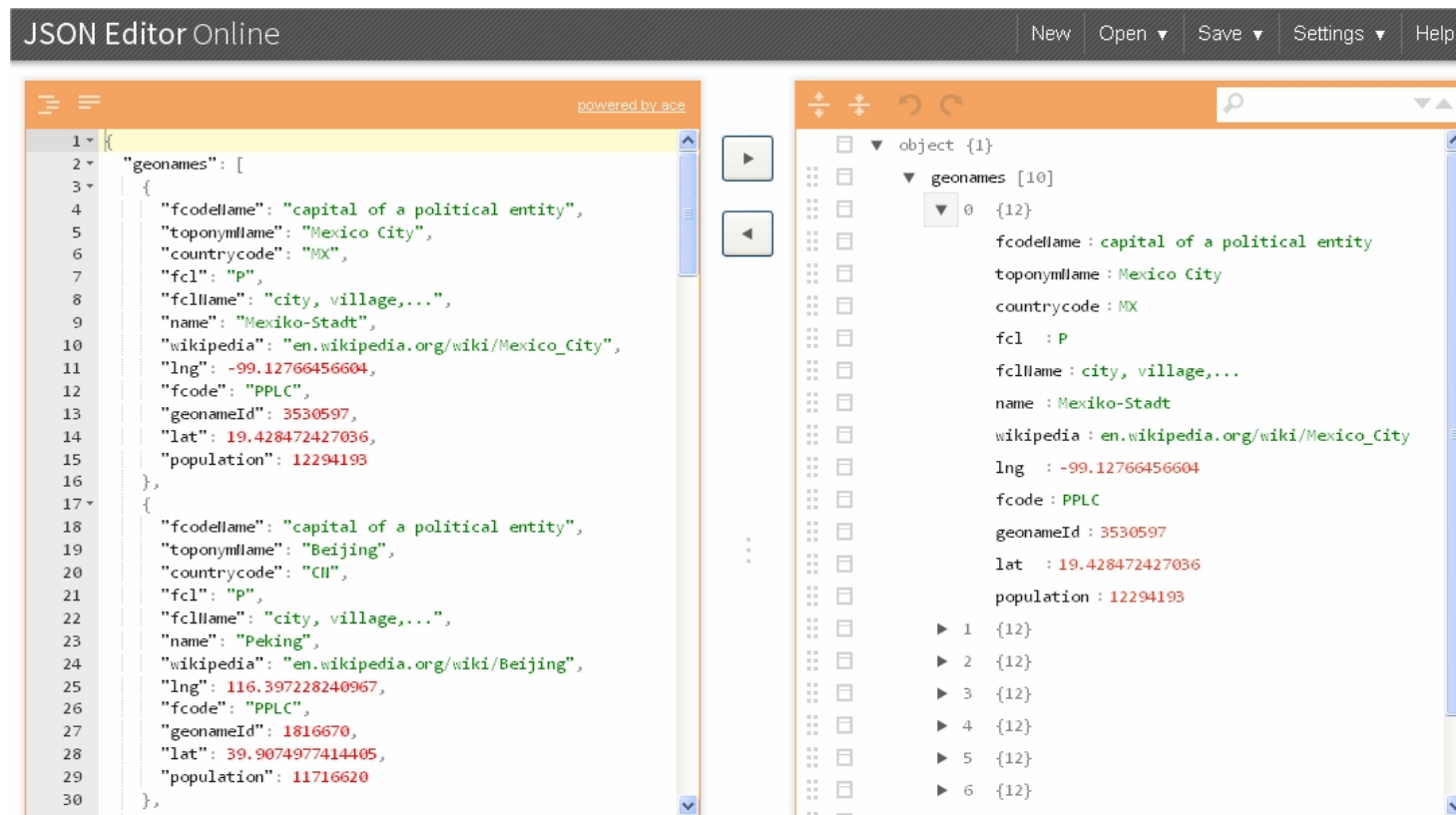
        if (conn.getResponseCode() != 200) {
            throw new RuntimeException("HTTP-Fehler : "
                + conn.getResponseCode());
        }

        BufferedReader br = new BufferedReader(new InputStreamReader(
            (conn.getInputStream())));

        String ausgabe;
        System.out.println("Ausgabe des Servers ...\n");
        while ((ausgabe = br.readLine()) != null) {
            System.out.println(ausgabe);
        }

        conn.disconnect();
    }
}
```

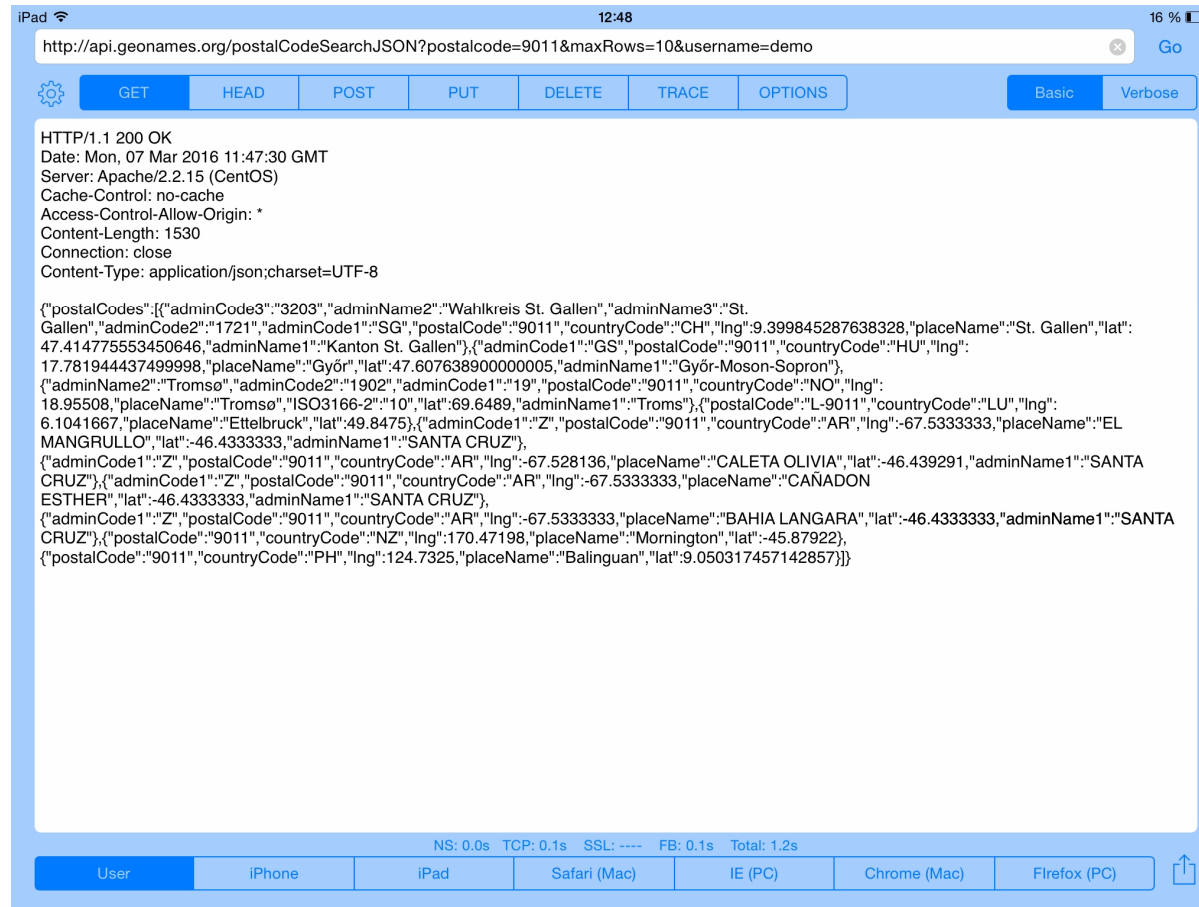
JSON Editor Online



Quelle: <http://www.jsoneditoronline.org/> (letzte Verwendung: März 2016)



iCurl – HTTP-Operationen



Quelle: <https://itunes.apple.com/de/app/icurlhttp/id611943891?mt=8>

XML Editor Online

Code Beautify

News | My Ip | Search | Recent Links | Sample | Save | More ▾ | Sign in | (?)

XML VIEWER

Save & Share

XML Input

```
1 <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2 <geonames>
3 <geoname>
4 <toponymname>Mexico City</toponymname>
5 <name>Mexico City</name>
6 <lat>19.42847</lat>
7 <lng>-99.12766</lng>
8 <geonameId>3530597</geonameId>
9 <countryCode>MX</countryCode>
10 <countryName>Mexico</countryName>
11 <fcl>P</fcl>
12 <fcode>PPLC</fcode>
13 </geoname>
14 <geoname>
15 <toponymname>Beijing</toponymname>
16 <name>Beijing</name>
17 <lat>39.9075</lat>
18 <lng>116.39723</lng>
19 <geonameId>1816670</geonameId>
20 <countryCode>CN</countryCode>
21 <countryName>China</countryName>
22 <fcl>P</fcl>
23 <fcode>PPLC</fcode>
24 </geoname>
25 <geoname>
26 <toponymname>Manila</toponymname>
27 <name>Manila</name>
28 <lat>14.6042</lat>
29 <lng>120.9822</lng>
30 <geonameId>1701668</geonameId>
31 <countryCode>PH</countryCode>
32 <countryName>Philippines</countryName>
```

Load Url

Browse

Tree View

Beautify / Format

Minify

Checkout
NeoSense 1.2
Detect and Diagnose Perf
Problems. Discover what's
New Now!

XML To JSON

Result : XML Tree View

```
geonames ..
├── geoname ..
│   ├── toponymname Mexico City
│   ├── name Mexico City
│   ├── lat 19.42847
│   ├── lng -99.12766
│   ├── geonameid 3530597
│   ├── countryCode MX
│   ├── countrynome Mexico
│   ├── fcl P
│   └── fcode PPLC
├── geoname ..
│   └── toponymname Beijing
```

Quelle: <http://codebeautify.org/xmlviewer> (letzte Verwendung: März 2016)



Einsatz Mock-Service (REST/XML)

The screenshot displays the Mockable.io Admin console interface. At the top, it shows the user is signed in as 'Future mockable.io User em6cgu7ejd'. A warning banner indicates the temporary account will expire in 24 hours. The main area shows the configuration for a REST mock service named 'GET /HWR' on the path 'HWR'. The service is currently 'Started' and is configured to return a '200 OK' response. A table below the configuration shows the mock is running, with three requests logged, each returning a '200 OK' response. On the right side, there is a section titled 'Mock-Service einrichten' with an arrow pointing to the configuration area. On the left side, there is a section titled 'Requests überwachen' with an arrow pointing to the monitoring table.

Mock-Service einrichten

Requests überwachen

Status	Name	Path	Verb	Consumes	Produces	Logger	Set Delay
Started	GET /HWR	HWR	GET	*/*	*/*	Actions	Toggle (2s)

Mock Name	Path	Verb	Response	Time
GET /HWR	http://demo0744095.mockable.io/HWR	GET	200 OK	a few seconds ago
GET /HWR	http://demo0744095.mockable.io/HWR	GET	200 OK	a few seconds ago
GET /HWR	http://demo0744095.mockable.io/HWR	GET	200 OK	a few seconds ago

Quelle: <https://www.mockable.io> (letzte Verwendung: März 2016)



HTTP-Monitoring

HTTP-Monitoring

localhost:9000/admin/

ServiceProxies Proxies Transport System Statistics Calls Clients About

ServiceProxies

Show 25 entries Search:

Order	Name	Listen Port	Virtual Host	Method	Path	Target Host	Target Port	Count	Actions
1	<u>** */restnames/:80</u>	80	*	*	/restnames/	www.thomas-bayer.com	80	0	
2	<u>BLZService</u>	80	*	*	/Q/axis2/services/BLZService\E.*	www.thomas-bayer.com	80	0	
3	<u>** *.*:9000</u>	9000	*	*	.*		0	16	

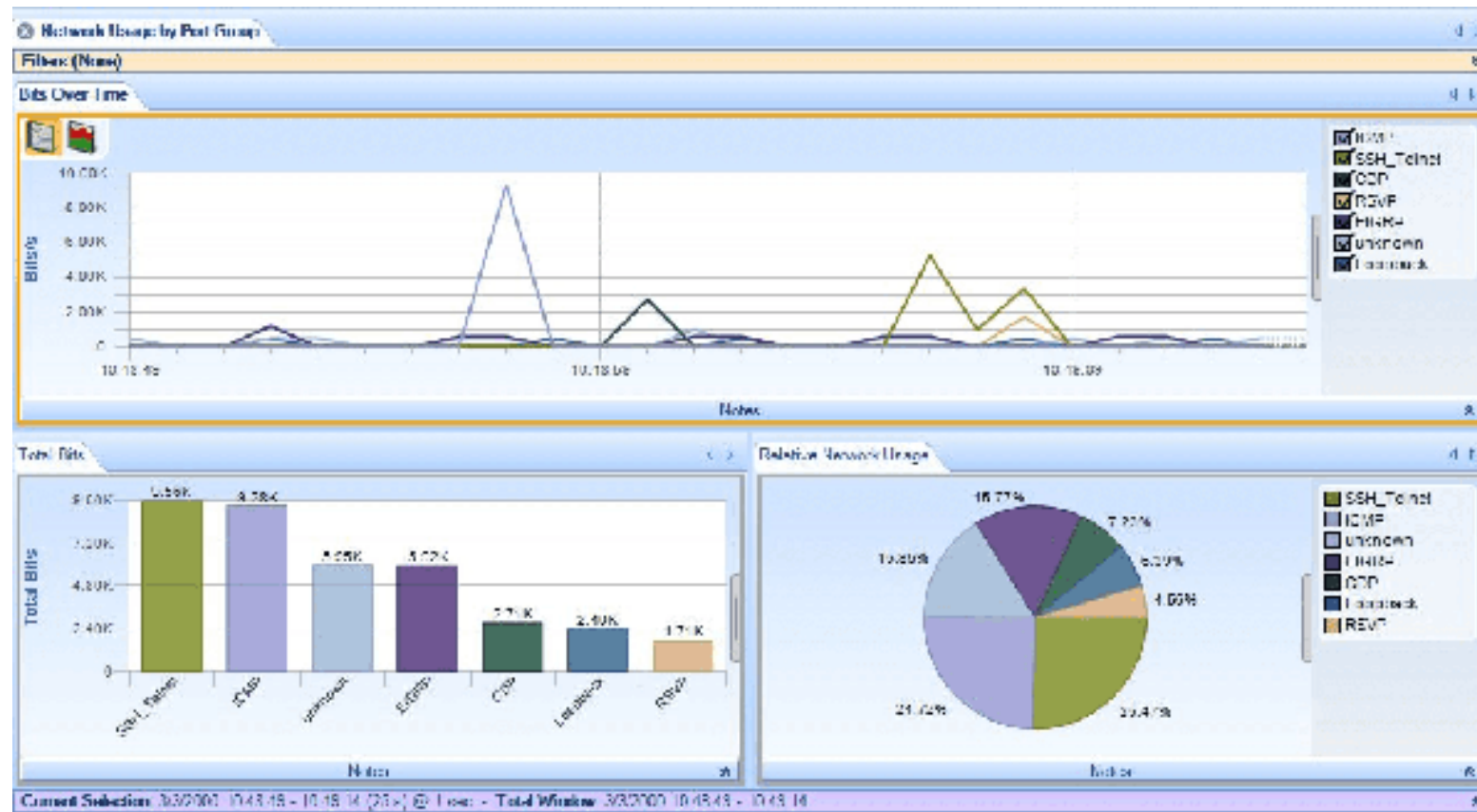
Showing 1 to 3 of 3 entries

Name Listen Port Method Target Host Target Port

Copyright ©2009-2012 predic8 GmbH. All Rights Reserved. See <http://membrane-soa.org/esb/> for documentation and updates.

Quelle: <https://www.membrane-soa.org/>

Netzwerkmonitoring



Quelle: www.wireshark.org

Organisation der Übung



Organisation der Übung

Bitte berücksichtigen Sie die folgende Vorgehensweise:

- Die Übung wird in max. 4er Gruppen durchgeführt!
- Vorgeschlagene Methode zur Aufgabenlösung
 - Abstimmung einer geeigneten Vorgehensweise (20 min)
 - Ausführen der Aufgabenstellungen (150 min)
 - Erstellung eines Protokolls (30 min)
- Gesamtzeit für die Durchführung: ca. 200 min
- Bereitstellung eines entsprechenden Protokolls je Gruppe
- Abgabe des Protokolls am Ende des Semester



Aufbau des Protokolls

Verwenden Sie bitte das folgende Muster für das Protokoll:

- Allgemeine Angaben
 - Versuch, Beteiligte Studenten, Datum
 - Rahmenbedingungen (Software, ...)
 - Methodisches Vorgehen zu Bearbeitung
- Aufgaben des Laborversuchs
 - Aufgabenstellung
 - Textliche Ausführungen zu den Lösungen
 - Verwendung von Grafiken und Screenshots
- Zusammenfassung (Bewertung der erreichten Ergebnisse)
- Genutzte Quellen (z.B. Literatur, Internet, ...)