Exercise – A Web Service Customer Orders Retrieval System

Dokumentation

Roland Lehner

MatrNr.: 1310277029

Umsetzung

Die Services wurden mit der Programmiersprache Python realisiert. Grund für diese Entscheidung war, dass Python auf den Betriebssystemen Linux, OSX und Windows läuft. Darüber hinaus ist Python ressourcensparend. Es kann auch auf einem kleinen Server, wie zum Beispiel einem Raspberry Pi eingesetzt werden.

Als Hilfsmittel wurden 3rd Party Libraries verwendet. Eine für die SOAP Implementierung und zwei für die REST Implementierung.

SOAP:

- pysimplesoap: Pysimplesoap ist eine simple SOAP Library, für Client und Server Webservice Interfaces.

REST:

- Flask: Flask ist eine Rest Library, die es vereinfacht einen Rest Server aufzusetzen. Sie macht es einfach die Routes für den REST-Service zu definieren. Die Response wird in Form eines JSON Files geliefert. Anfragen mit POST werden ebenfalls im JSON Format abgesendet.
- Requests: Requests wurde entwickelt um Webrequests einfach und schnell absetzen zu können. Es ersetzt die Python Standard Library urllib2.

SOAP

In der SOAP Implementierung werden folgende Funktionen verwendet:

- AddCustomer
- AddOrder
- DeleteCustomer
- DeleteOrder
- ShowCustomers
- ShowOrders

Es wird ein SOAP Envelope erstellt der an die Url (http://localhost:8008/) gesendet wird.

Pysimplesoap generiert die SOAP XML Envelopes im Hintergrund und schickt sie als Request an den Server bzw. als Response an den Client. Wenn der Server gerade läuft, können die Envelopes über folgende Links generiert werden:

AddCustomer:

Request (http://localhost:8008/AddCustomer?request):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<AddCustomer xmlns="http://example.com/sample.wsdl">
      ▼<customerName>
          <!-- string -->
        </customerName>
      </AddCustomer>
    </soap:Body>
  </soap:Envelope>
```

Response (http://localhost:8008/AddCustomer?response):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<AddCustomerResponse xmlns="http://example.com/sample.wsdl">
      ▼<AddCustomerResult>
          <!-- boolean -->
        </AddCustomerResult>
      </AddCustomerResponse>
    </soap:Body>
  </soap:Envelope>
```

AddOrder:

Request (http://localhost:8008/AddOrder?request):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<AddOrder xmlns="http://example.com/sample.wsdl">
      ▼<customerName>
          <!-- string -->
        </customerName>
      ▼<order>
          <!-- string -->
        </order>
      </AddOrder>
    </soap:Body>
  </soap:Envelope>
```

Response (http://localhost:8008/AddOrder?response):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<AddOrderResponse xmlns="http://example.com/sample.wsdl">
      ▼<AddOrderResult>
          <!-- boolean -->
        </AddOrderResult>
      </AddOrderResponse>
    </soap:Body>
  </soap:Envelope>
```

DeleteCustomer:

Request (http://localhost:8008/DeleteCustomer?request):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<DeleteCustomer xmlns="http://example.com/sample.wsdl">
      ▼<customerName>
          <!-- string -->
        </customerName>
      </DeleteCustomer>
    </soap:Body>
  </soap:Envelope>
```

Response (http://localhost:8008/DeleteCustomer?response):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<DeleteCustomerResponse xmlns="http://example.com/sample.wsdl">
      ▼<DeleteCustomerResult>
          <!-- boolean -->
        </DeleteCustomerResult>
      </DeleteCustomerResponse>
    </soap:Body>
  </soap:Envelope>
```

DeleteOrder:

Request (http://localhost:8008/DeleteOrder?request):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<DeleteOrder xmlns="http://example.com/sample.wsdl">
      ▼<identifier>
          <!-- string -->
        </identifier>
      ▼<customerName>
          <!-- string -->
        </customerName>
      </DeleteOrder>
    </soap:Body>
  </soap:Envelope>
```

Response (http://localhost:8008/DeleteOrder?response):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<DeleteOrderResponse xmlns="http://example.com/sample.wsdl">
      ▼<DeleteOrderResult>
          <!-- boolean -->
        </DeleteOrderResult>
      </DeleteOrderResponse>
    </soap:Body>
  </soap:Envelope>
```

GetCustomers:

Request (http://localhost:8008/GetCustomers?request):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
     <GetCustomers xmlns="http://example.com/sample.wsdl"/>
   </soap:Body>
  </soap:Envelope>
```

Response (http://localhost:8008/GetCustomers?response):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<GetCustomersResponse xmlns="http://example.com/sample.wsdl">
       <GetCustomersResult><type 'list'></GetCustomersResult>
     </GetCustomersResponse>
   </soap:Body>
  </soap:Envelope>
```

GetOrders:

Request (http://localhost:8008/GetOrders?request):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<GetOrders xmlns="http://example.com/sample.wsdl">
      ▼<customerName>
          <!-- string -->
       </customerName>
     </GetOrders>
   </soap:Body>
  </soap:Envelope>
```

Response (http://localhost:8008/GetOrders?response):

```
▼<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  ▼<soap:Body>
    ▼<GetOrdersResponse xmlns="http://example.com/sample.wsdl">
       <GetOrdersResult><type 'list'></GetOrdersResult>
     </GetOrdersResponse>
   </soap:Body>
  </soap:Envelope>
```

REST

Die Routes im REST Service setzen sich wie folgt zusammen:

/orderService/api/v1.0/

Es wird eine Versionsnummer in der URL definiert um ein Update der API durchführen zu können, ohne dass die momentan bestehende API ausfällt.

Folgende URLs wurden für das REST Service definiert:

GET: http://localhost:5000/orderService/api/v1.0/customers

Response:


```
{
  "customers": [
    {
      "name": "Koehler",
      "orders": [
        {
          "orderID": 1,
          "products": [
            {
              "product": "headphones"
            },
            {
              "product": "laptop"
            }
          ]
        },
        {
          "orderID": 2,
          "products": [
            {
              "product": "book"
            },
            {
              "product": "tablet"
            }
          ]
        }
      ]
    }
  ]
}
```

GET: http://localhost:5000/orderService/api/v1.0/<CustomerName>/orders

Response:


```
{
  "orders": [
    {
      "orderID": 1,
      "products": [
        {
          "product": "headphones"
        },
        {
          "product": "laptop"
        }
      ]
    },
    {
      "orderID": 2,
      "products": [
        {
          "product": "book"
        },
        {
          "product": "tablet"
        }
      ]
    }
  ]
}
```


POST: http://localhost:5000/orderService/api/v1.0/customers

Request data : {“name”: “Franz”}

Response:


```
{
  "customer": {
    "name": "Franz",
    "orders": []
  }
}
```

POST: http://localhost:5000/orderService/api/v1.0/customers/<customerName>/order

Request data: {"products": [{"product": "asdf"} , {"product": "laptop"}] }

Response:


```
{
  "order": {
    "orderID": 3,
    "products": [
      {
        "product": "asdf"
      },
      {
        "product": "laptop"
      }
    ]
  }
}
```


Die OrderID wird vom Server automatisch generiert.

DELETE: http://localhost:5000/orderService/api/v1.0/customers/<customerName>/order/<OrderID>

Response:


```
{
   result: true
}
```


DELETE: http://localhost:5000/orderService/api/v1.0/customers/<customerName>

Response:


```
{
   result: true
}
```


Vergleich

Wenn man die beiden Implementierungen vergleicht, fällt auf das SOAP Services einen größeren Overhead an Daten produzieren als REST Services. In SOAP Requests/Responses stehen immer die Envelope Tags und Body Tags mit Informationen. Die Handhabung von REST Services ist auch um einiges leichter, da als Envelope JSON verwendet wird. JSON hat den Vorteil, dass es gut lesbar ist und einfach zu parsen. Bei SOAP ist der Vorteil das WSDL. WSDL definiert die einzelnen Methoden des Services. Damit hat man einen guten Überblick über den Funktionsumfang.

Learnings

Python ist eine tolle Programmiersprache für kleine Projekte. Es bedarf zu Beginn an Einarbeitungszeit aber dann geht es schnell voran.

Die Libraries Flask und Requests sind echt gut und funktionieren einwandfrei. Probleme traten auf mit der Library pysimplesoap. Ich hatte Probleme ein Array von Orders zu erstellen, wo in jedem Order mehrere Produkte eingegliedert waren. Anscheinend hat die Library Probleme mit 2 dimensionalen Arrays. Darüber hinaus hat pysimplesoap Probleme ein WSDL zu generieren. Wenn der Benutzer direkt auf die Server URL navigiert (https://localhost:8008/) sollte das WSDL generiert und angezeigt werden. Leider bricht dieser Vorgang wegen einem string parsing error ab.

Installation

Für die Installation der Libraries wird Python benötigt.

Die Libraries werden wie folgt installiert:

Pysimplesoap:

Für Windows/Linux/Unix gibt es Installationsdateien unter:
https://code.google.com/p/pysimplesoap/downloads/list

Falls der Python Package Installer (pip) installiert ist:

pip install pysimplesoap

Flask:

http://pypi.python.org/packages/source/F/Flask/Flask-0.10.1.tar.gz

Flask wird mit einem Installer geliefert, der auf Python basiert.

Requests:

Requests kann mit dem pip oder von github installiert werden.

Github: https://github.com/kennethreitz/requests.git

Pip: pip install requests