LU03d - Webservice planen

Bei der Planung definieren wir die Schnittstelle und die Umsetzung unserer Webservices.

Definition von Webservices

siehe auch *Webservice*

Ein Webservice dient zur Kommunikation zwischen zwei Anwendungen. Dabei sendet der Client einen HTTP-Request mit der eindeutigen URI des Webservices. Der Webservice sendet im Erfolgsfall eine Response mit den Daten.

Planung

Vor der Realisierung eines Webservices steht die Planung.

1. Aufgabe des Services
2. Pfad des Services
3. Bezeichner der Ressource (Python-Modul)
4. Art des HTTP-Requests
5. Parameter im Request
    - Query-Parameter
    - Form-Parameter
6. Response des Services
    - Welche Daten werden geliefert
    - Datenformat der Response
    - HTTP-Statuscodes

Zu jedem Teilschritt finden Sie als Beispiel den Service „Bücherliste“ aus der Applikation „Bookshelf“.

1. Aufgabe des Services

Jeder Webservice soll eine einfache, klar definierte Aufgabe erledigen. Dadurch können die gleichen Services in verschiedenen Formularen und Applikationen eingesetzt werden.

Vermeiden Sie bei der Definition der Aufgabe Wörter wie **und** bzw. **oder**.

Beispiel

Der Webservices liefert eine Collection mit allen Büchern.

2. Pfad des Services

Jedem Webservice wird eine eindeutige URI zugeordnet. Diese URI besteht aus:

- Adresse des Servers:
  - Protokoll: `https`
  - Domain (FQDN): z.B. `localhost` oder `service.ghwalin.ch`
  - Port: z.B. `:5000`
- Name der Applikation: z.B. `bookshelf`
- Pfad des Services: z.B. `library`

Beispiel

Der Service für die Bücherliste soll unter http://localhost:5000/library erreichbar sein.

3. Bezeichner der Ressource

Die Umsetzung eines Webservices erfolgt in einer Klasse in einem Python-Modul. Durch die Wahl eines sprechenden Bezeichners können wir später die Klassen einfach identifizieren.

Beispiel

Ich nenne meine Ressource `library_resource` bzw. die Klasse `LibraryResource`.

4. Art des HTTP-Requests

Es gibt verschiedene Arten von HTTP-Requests. Diese Request-Arten unterscheiden sich aufgrund der Parameterübergabe und der Art der Response.

Vielleicht haben Sie schon von GET und POST gehört. Dies sind zwar die häufigsten Arten, aber längst nicht die einzigen. Einige Request-Arten stelle ich Ihnen hier näher vor. Ein vollständige Liste finden Sie unter https://developer.mozilla.org/de/docs/Web/HTTP/Methods.

Art	**Zweck**	**Parameter**
GET	Daten lesen	URI
POST	Neue Daten speichern	URI und Formulardaten
PUT	Bestehende Daten ersetzen	URI und Formulardaten
DELETE	Bestehende Daten löschen	URI

Beispiel

Der Webservice „Bücherliste“ soll Daten lesen. Daher verwende ich einen GET-Request.

5. Parameter im Request

Manche Requests müssen dem Webservice Parameter bzw. Daten für die Verarbeitung liefern. Dabei unterscheiden wir zwischen verschiedenen Möglichkeiten zur Übermittlung der Parameter.

Query-Parameter

Query-Parameter werden in der URI mitgegeben. Sie bestehen immer aus einem Schlüssel und dem entsprechenden Wert.

Die URI https://moodle.bzz.ch/mod/lesson/view.php?id=8990&pageid=2390 enthält zwei Parameter:

- id=8990
- pageid=2390

Formular-Parameter

Benutzereingaben in einem Formular können theoretisch auch in der URI übertragen werden. Allerdings sieht man dann im Browserverlauf, welche Eingaben gemacht wurden. Das ist spätestens bei Login-Formularen (Passwort!!) sehr schlecht.

In der Regel übertragen wir die Benutzereingaben im Datenteil des Requests.

Beispiel

Mein Webservice „Bücherliste“ erwartet keine Parameter.

6. Response des Services

HTTP-Statuscode

Über den HTTP-Statuscode signalisiert der Service, ob die Verarbeitung erfolgreich war. Einige gebräuchliche Statuscodes sind:

- 200: OK
- 201: Created
- 400: Bad request
- 401: Unauthorized
- 403: Forbidden
- 404: Not found
- 500: Internal Server Error

Daten

Beim Senden der Daten in der Response stehen uns fast alle Möglichkeiten offen. Wir können vom einfachen Text, über ein einzelnes Objekt, bis hin zu Collections im Datenteil der Response senden. Oft bieten Services dem Aufrufer die Daten in unterschiedlichen Formate an, z.B. JSON, XML Wir beschränken uns vorerst auf das JSON-Format, da dieses sehr weit verbreitet ist. Ausserdem kann Flask selbständig einen Dictionary ins JSON-Format umwandeln.

Beispiel

Der HTTP-Statuscode 200 gibt an, dass die Bücher gefunden wurden.

Der Webservice gibt als Response einen Dictionary von Book-Objekten zurück. Diese Liste soll im JSON-Format codiert werden.

M321-LU03