TKN Telecommunication Networks Group

Praktikum
Rechnernetze und Verteilte Systeme

Block 5

— RESTful APIs —

Termin: 12.-14.12.2016 & 3.1.-6.1.2017

1 Theoretische Vorbereitungsaufgaben

Die folgenden Aufgaben sollen Ihnen helfen, sich auf den Vorbereitungstest vorzubereiten. Klären Sie bitte mögliche Fragen oder Unklarheiten unbedingt vor den ISIS-Testaten!

Aufgabe 1:
Beantworten Sie im Kontext vom Hypertext Transfer Protocol (HTTP) folgende Fragen:

a) Welche HTTP Methoden gibt es und was machen diese?

b) Welche HTTP Methoden sind idempotent?

c) Was sind persistente und nicht-persistente Verbindungen?

d) HTTP ist ein Zustandsloses Protokoll. Was bedeutet das? Mit welcher Technik können Server z.B. trotzdem den Warenkorb einer Nutzers speichern?

Aufgabe 2:
Beantworten Sie im Kontext vom Caching im WWW folgende Fragen:

a) Warum wird Caching genutzt?

b) Welche Nachteile hat Caching?

c) Welche verschiedenen Arten von Caching werden im WWW eingesetzt?

d) Was bedeutet es für das Caching, wenn Methoden idempotent sind?

e) Mit welchen HTTP-Headern kann das Caching gesteuert werden? Was sind dabei die Unterschiede?

Aufgabe 3:
Beantworten Sie im Kontext vom RESTful Webservices folgende Fragen:

a) Was ist REST? Was genau bedeutet “Representational State Transfer”?

b) Was sind die Haupt-Prinzipien von REST, die in der Vorlesung angesprochen werden?

c) Welche Eigenschaften verspricht man sich von der Nutzung von REST?

d) Was kann bei REST (k)eine Ressource sein? Was ist dabei der Unterschied zu klassischen Webservices aufbauend auf WSDL/SOAP?

e) Welche Methoden können auf Ressourcen ausgeführt werden?

f) Wann ist eine Ressource cachable?

2 Praktische Aufgabe

Wir wollen diesen Block nutzen, um einen etwas abgewandelten Modus zu testen. Die praktischen Aufgabe sind weiterhin in Kleingruppen von i. d. R. 3 Personen zu lösen. Es wird nur noch eine Aufgabe geben, die bis zum Ende des Blocks bearbeitet werden muss. Die Rücksprache kann weiterhin im zweiten Termin durchgeführt werden, als Alternative kann aber auch bei unfertigen Lösungen der Tutor mit Fragen konsultiert werden. Damit ist also eine Art Sprechstunde beim Tutor gegeben. Reichen Sie also bitte den Quelltext bzw. Lösungen bis Sonntag nach dem dem zweiten Termin des Blocks bis 23:55 Uhr per ISIS ein, das ist in diesem Fall der 8.1.2017. Es besteht weiterhin in beiden Terminen offiziell Anwesenheitspflicht.

Aufgabe 4:
Sie haben in vorherigen Blöcken bereits einige Male mit HTTP gearbeitet und ebenfalls einige Male entfernte Dienste bereitgestellt und aufgerufen. Ziel dieser Aufgabe ist nun, einen RESTful Webservice auf Basis von HTTP anzubieten.

Als Beispielanwendung soll wieder unsere Filmdatenbank bzw. Hash-Tabelle dienen. Diese soll über ein RESTful Interface angesprochen werden. Sie kennen aus der Vorlesung schon die HTTP-Methoden:

Resource	**GET**	**PUT**	**POST**	**DELETE**
Collection	Liste	(Ersetze Collection)	Erzeuge Element	Lösche Collection
Element	Repräsentation	Ersetzen	-	Löschen

Es ist Ihre Aufgabe sich selbst zu überlegen, welche Ressourcen sie benutzen wollen und unter welcher URL diese zu finden sind. Sie sollen die Methoden so implementieren, wie sie in der Tabelle oben gegeben sind. Ein neuer Datensatz wird also durch ein POST auf die Collection-URL erstellt. Sollte der Datensatz schon vorhanden sein, soll ein Fehler zurückgegeben werden. Ebenso beantworten Sie einen POST auf ein Element mit einem Fehler. Sie brauchen den PUT-Befehl nur für einzelne Elemente implementieren und nicht für die gesamte Collection, auch wenn das nach den RESTful-Prinzipien sinnvoll wäre.

Ihr HTTP-Server soll dabei korrekt mit einem entsprechenden Status-Code antworten. Status-Codes stehen im Header und geben an, ob die Aktion auf Server-Seite erfolgreich ausgeführt wurde. RFC7231[1] definiert grob, wie wann zu antworten ist. In der Regel sollen erfolgreiche Operationen mit `200 (OK)` bestätigt werden. Fehler können `404 (Not Found)` zurückgeben. Bei neu angelegten Ressourcen sollte der Server `201 (Created)` zurückgeben und den Location-Header auf die entsprechende Adresse, an der sich die neue Ressource befindet, setzen.

Als Rückgabe-Format sollten Sie den MIME-Type `application/json` verwenden, auch wenn der Client etwas anderes anfragt (das ist im HTTP-Standard ausdrücklich erlaubt). Für Key und die einzelnen Attribute aus vorherigen Aufgaben können Strings angenommen werden. Wie genau Sie Filme in JSON darstellen, bleibt Ihnen überlassen. In Collection-URLs sollten Hyperlinks verwendet werden[2]. Es bietet sich an zum Parsen eine geeignete Library zu verwenden. Wir würden cJSON empfehlen [3].

[1] `http://tools.ietf.org/html/rfc7231#section-4.3`
[2] `https://spring.io/understanding/HATEOAS`
[3] `https://github.com/DaveGamble/cJSON`

Damit Sie nicht den kompletten Server selbst programmieren müssen und Sie in Kontakt mit externen Bibliotheken kommen, verwenden wir in diesem Block zusätzlich die Onion http server library[4]. Wir haben für diese Aufgabe ein Beispiel in C vorgegeben, welches im Wesentlichen einen Beispielserver aus der oben angegebenen Bibliothek enthält. Wir haben jedoch die Bibliothek schon für Sie kompiliert und eine sinnvolle Ordnerstruktur angelegt. Die Bibliothek wird als 32 und 64bit Version für Linux bereitgestellt und ist dazu gedacht, statisch gelinkt zu werden. Das Beispiel kann mit dem folgen Aufruf kompilieren werden:

```
gcc -o block4 -I inlcude -L lib/32 -L lib/64 block4.c -pthread -lonion_static
-lrt
```

Im include-Pfad befindet sich dabei die Header-Dateien der Bibliothek, in der die Funktionsrümpfe und Makros deklariert sind. Hier sollten Sie suchen, wenn Sie z.B. nicht wissen, wie ein korrekter Aufruf auszusehen hat oder welches Makro für welchen Status-Code steht. In den beiden Unterverzeichnissen in lib finden sich die kompilierten Bibliotheken. Der Linker sucht sich die passende Bibliothek, je nachdem, ob für 32 oder 64bit kompiliert wird. Der folgende Hinweis kann dabei ignoriert werden. Er deutet nur darauf hin, dass eine der beiden Bibliotheken nicht anwendbar war:

```
/usr/bin/ld:  skipping incompatible lib/32/libonion_static.a when searching for
-lonion_static
```

Die Vorgabe sollte so schon kompilierbar sein und eine einfache Demo-Seite zeigen.

Die Vorgabe akzeptiert nur die GET-Methode. Sie werden im Laufe der Aufgabe auch die POST, PUT und DELETE Methoden implementieren müssen. Ein guter Weg diese Methode zu testen ist das Kommandozeilentool “curl”. Mit ihm können verschiedene Typen von Requests abgesetzt werden. Zusätzlich können die HTTP Header gesetzt und angezeigt werden. Einige Beispiele des Aufrufs von curl auf einen Server auf localhost auf Port 8080:

GET:

```
curl -v http://localhost:8080/example
```

PUT:

```
curl -v -H "Content-Type:  application/json" -X PUT -d
'{"username":"xyz","password":"xyz"}' http://localhost:8080/example
```

POST:

```
curl -v -H "Content-Type:  application/json" -X POST -d
'{"username":"xyz","password":"xyz"}' http://localhost:8080/example
```

DELETE:

```
curl -v -X DELETE http://localhost:8080/example
```

Wir haben die Bibliothek so angepasst, dass die Arbeit mit ihr etwas einfacher ist. In der Bibliothek wird eigentlich bei jedem PUT eine temporäre Datei angelegt, deren Inhalt dann wieder ausgelesen werden muss. Das hat viele Studenten aus vorherigen Semestern verwirrt. Wir haben deshalb libonion dehingehend angepasst, dass man den Inhalt der PUT-Methode in der gleichen Weise abrufen kann wie bei der POST-Methode. Konsultieren Sie für Details das Zusatzmaterial.

Implementieren Sie selbst, wie Methoden auf einzelne Ressourcen durch Ihren Server verarbeitet werden. Die benötigte Hashtable können sie direkt in ihren Server integrieren. Dadurch ist es auch möglich, alle Ressourcen aufzulisten.

3 Vertiefungsaufgaben

Diese Aufgaben sind zu Ihrer eigenen Vertiefung in Hinblick auf die Klausurvorbereitung gedacht:

Aufgabe 5:
Nennen Sie die wesentlichen Unterschiede zwischen HTML, XML und XHTML!

[4] `https://github.com/davidmoreno/onion`

Aufgabe 6:
Laden Sie die WSDL-Datei des TKN-Sensornetzwerk-Gateways von der ISIS-Seite herunter (inzwischen wird der Service zu Gunsten von REST nicht mehr angeboten).

a) Interpretieren Sie knapp die allgemeine Bedeutung der folgenden Elemente[5]:
   - `<types>`
   - `<message>`
   - `<interface>/<portType>`[6]
   - `<binding>`
   - `<service>`

b) Welche Adresse hat der in der WSDL-Datei beschriebene Web-Service?

c) In welcher Programmiersprache ist der Web-Service implementiert? In welcher Programmiersprache ist der Web-Service-Client zu implementieren?

Aufgabe 7:
UDDI war nie so weit verbreiteten, wie die Erfinder gehofft hatten. IBM, Microsoft und SAP kündigten bereits im Januar 2006 an, ihre öffentlichen UDDI-Knoten zu schließen. Kann ein Web-Service auch ohne "Universal Description, Discovery and Integration" (UDDI)-Eintrag angeboten werden? Ergibt das ggf. Sinn?

[5] Hilfreich ist dabei die z.B. die WSDL-Spezifikation: `http://www.w3.org/TR/wsdl`
[6] Beide Bezeichner werden genutzt; im konkreten Beispiel `<portType>`