TKN Telecommunication Networks Group

Praktikum
Rechnernetze und Verteilte Systeme

Block 5

— REST —

Termin: 8.-10.12.2014 & 15.-17.12.2014

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) 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 Aufgaben

Die praktischen Aufgaben sind in Kleingruppen von i. d. R. 3 Personen zu lösen. Die Ergebnisse des ersten Termins führen Sie im zweiten Termin dem Tutor vor. Reichen Sie bitte des weiteren den Quelltext bzw. Lösungen bis Sonntag vor dem dem zweiten Termin 23:55 Uhr per ISIS ein.

Im zweiten Termin werden vertiefende praktische Aufgaben gestellt, während der Tutor Lösungen des ersten Termins abnimmt. Reichen Sie bitte den Quelltext bzw. Lösungen dieser Aufgaben bis Sonntag vor dem nächsten Termin 23:55 Uhr per ISIS ein.

Es besteht in beiden Terminen grundsätzlich Anwesenheitspflicht.

2.1 Im ersten Termin zu lösen

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 Hash-Tabelle dienen. Diese soll über ein RESTful Interface angesprochen werden. Dazu lassen Sie den Hash-Server bzw. die Peers der DHT unverändert und passen Ihren Client so an, dass er Anfragen von HTTP-Clients entgegennehmen, weiterleiten und beantworten kann.

Sie kennen aus der Vorlesung schon die HTTP-Methoden. Für diese Aufgabe sollten die Methoden GET, POST und DELETE ausreichen:

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 mit welchen Methoden diese aufgerufen werden sollen. Als Rückgabe-Format sollten Sie den MIME-Type "text/-plain" verwenden, auch wenn der Client etwas anderes anfragt (das ist im HTTP-Standard ausdrücklich erlaubt) und eine geeignete Darstellung für die Zahlen für Key und Value verwenden. Es ist keine besondere Kodierung vorgegeben, sie sollte aber Menschen-lesbar sein.

Ihr HTTP-Server soll dabei auch 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.

Damit Sie nicht den kompletten Server selbst programmieren müssen und Sie in Kontakt mit externen Bibliotheken kommen, verwenden wir in diesem Block die GNU libmicrohttpd[2]. Für diese gibt es auch ein gutes Tutorial[3] mit relevanten Beispielen.

Wir haben für diese Aufgabe ein Grundgerüst für den Server in C vorgegeben. Die Bibliothek wird als 32 und 64bit Version für Linux bereitgestellt und ist dazu gedacht, statisch gelinkt zu werden. Das

[1] `http://tools.ietf.org/html/rfc7231#section-4.3`
[2] `http://www.gnu.org/software/libmicrohttpd/`
[3] `http://www.gnu.org/software/libmicrohttpd/tutorial.pdf`

Grundgerüst kann mit dem folgen Aufruf kompilieren werden:
`gcc -o block5 -Iinclude/ -Llib/32bit -Llib/64bit *.c -lmicrohttpd -lpthread`
Im include-Pfad befindet sich dabei die Header-Datei der libmicrohttpd, 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/32bit/libmicrohttpd.a when searching for -lmicrohttpd`
Die Vorgabe sollte so schon kompilierbar sein und demonstrieren, wie ein einfacher POST implementiert werden könnte.

Die Vorgabe akzeptiert nur die POST-Methode. 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 angezeigt werden. Einige Beispiele des Aufrufs von curl auf einen Server auf localhost auf Port 8080:
GET:
`curl -v http://localhost:8080/example`
POST:
`curl -v --data "name=Tarzan" http://localhost:8080/example`
DELETE:
`curl -v -X DELETE http://localhost:8080/example`

Die Hauptaufgaben sind hier nochmal zusammengefasst:

- Überlegen Sie zunächst auf Papier, welche Ressourcen sie benötigen, ob und wie es Sinn macht diese hierarchisch zu organisieren und welche HTTP Methoden jeweils unterstützt werden sollen bzw. was diese bewirken. Überlegen Sie sich auch, in welchem Format die Daten repräsentiert werden.
- Testen Sie die gegebene Vorgabe und versuchen Sie sie grob zu verstehen.
- Implementieren Sie selbst, wie Methoden auf einzelne Ressourcen durch Ihren Server verarbeitet werden und wie diese an den Hash-Table Server weitergeleitet werden.

2.2 Im zweiten Termin zu lösen

Aufgabe 5:

Passen sie einen Ihrer HTTP-Clients von vorherigen Aufgaben dahingehen an, dass er den vorher bereitgestellten RESTful Webservice aufrufen kann. Dazu soll diesmal keine externe Bibliothek genutzt werden. Als Test verwenden Sie wieder die Funktionalität Ihres alten Clients, also schreiben Sie 25 Werte, lesen Sie diese wieder aus, löschen Sie die Werte und lesen sie wieder (vergeblich).

3 Vertiefungsaufgaben

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

Aufgabe 6:
Nennen Sie die wesentlichen Unterschiede zwischen HTML, XML und XHTML!

Aufgabe 7:
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[4]:
   - `<types>`
   - `<message>`
   - `<interface>/<portType>`[5]
   - `<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 8:
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?

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