HOMEWORK REVIEW

RESTFUL APIS

GOOGLE CONTACTS API VERSION 3.0

RETRIEVING ALL CONTACTS

- Request

```
GET /m8/feeds/contacts/default/full
```

- Response

```
HTTP/1.1 200 OK
Content-Type: application/atom+xml; charset=UTF-8; type=feed
...

<feed>
  <id>userEmail</id>
  <updated>2008-12-10T10:04:15.446Z</updated>
  <entry>
    <id>
      http://www.google.com/m8/feeds/contacts/userEmail/base/contactId
    </id>
  </entry>
  <!-- Other entries ... -->
</feed>
```

RETRIEVING CONTACTS USING QUERY PARAMETERS

```
GET
https://www.google.com/m8/feeds/contacts/{userEmail}/full?
updated-min=2007-03-16T00:00:00
```

RETRIEVING A SINGLE CONTACT

- Request

```
GET /m8/feeds/contacts/default/full/{contactId}
```

- Response

```
HTTP/1.1 200 OK
Content-Type: application/atom+xml; charset=UTF-8; type=feed
...

<entry>
  <id>
    http://www.google.com/m8/feeds/contacts/{userEmail}/base/{contactId}
  </id>
</entry>
```

CREATING CONTACTS

- Request

```
POST /m8/feeds/contacts/default/full
...
<atom:entry>
  <atom:category scheme='http://schemas.google.com/g/2005#kind'
    term='http://schemas.google.com/contact/2008#contact'/>
  <gd:name>
     <gd:givenName>Elizabeth</gd:givenName>
     <gd:familyName>Bennet</gd:familyName>
     <gd:fullName>Elizabeth Bennet</gd:fullName>
  </gd:name>
  <atom:content type='text'>Notes</atom:content>
  <gd:email rel='http://schemas.google.com/g/2005#work'
    primary='true'
    address='liz@gmail.com' displayName='E. Bennet'/>
</atom:entry>
```

- Response

```
HTTP/1.1 201 Created
Content-Type: application/atom+xml; charset=UTF-8; type=feed
...
<atom:entry>
  <id>http://www.google.com/m8/feeds/contacts/userEmail</bar>/base/{contactId}</id>
  <updated>2008-12-10T04:45:03.331Z</updated>
  <title>Elizabeth Bennet</title>
  <gd:name>
     <gd:givenName>Elizabeth</gd:givenName>
     <gd:familyName>Bennet</gd:familyName>
     <gd:fullName>Elizabeth Bennet</gd:fullName>
  </gd:name>
</atom:entry>
```

DELETING CONTACTS

- Request

```
DELETE /m8/feeds/contacts/default/full/contactId
If-match: Etag
```

- Response

```
HTTP/1.1 200 OK
```

REPRESENTATIONAL STATE TRANSFER (REST)

- is a software architecture style
- consisting of guidelines and best practices for creating scalable web services.
- "Architectural Styles and the Design of Network-based Software Architectures"

ROY THOMAS FIELDING

- principal authors of the HTTP specification
- co-founder of the Apache HTTP Server project
- the chair of the Apache Software Foundation for its first three years

在为*HTTP/1.1*和*URI*的新标准设计扩展时，我认识到需要建立一个关于万维网应该如何运转的模型。这个关于整个*Web*应用中的交互的理想化的模型被称作表述性状态转移*(REST)*架构风格，成为了现代*Web*架构的基础。

这个名称“表述性状态转移”是有意唤起人们对于一个良好设计的*Web*应用如何运转的印象：一个由网页组成的网络（一个虚拟的状态机），用户通过选择链接（状态迁移）在应用中前进，导致下一个页面（展现应用的下一个状态）被转移给用户，并且呈现给他们，以便他们来使用。

REST架构约束

- Client-Server
- Stateless
- Cache
- Uniform Interface
- Layered System
- Code-On-Demand (optional)

REST架构过程视图

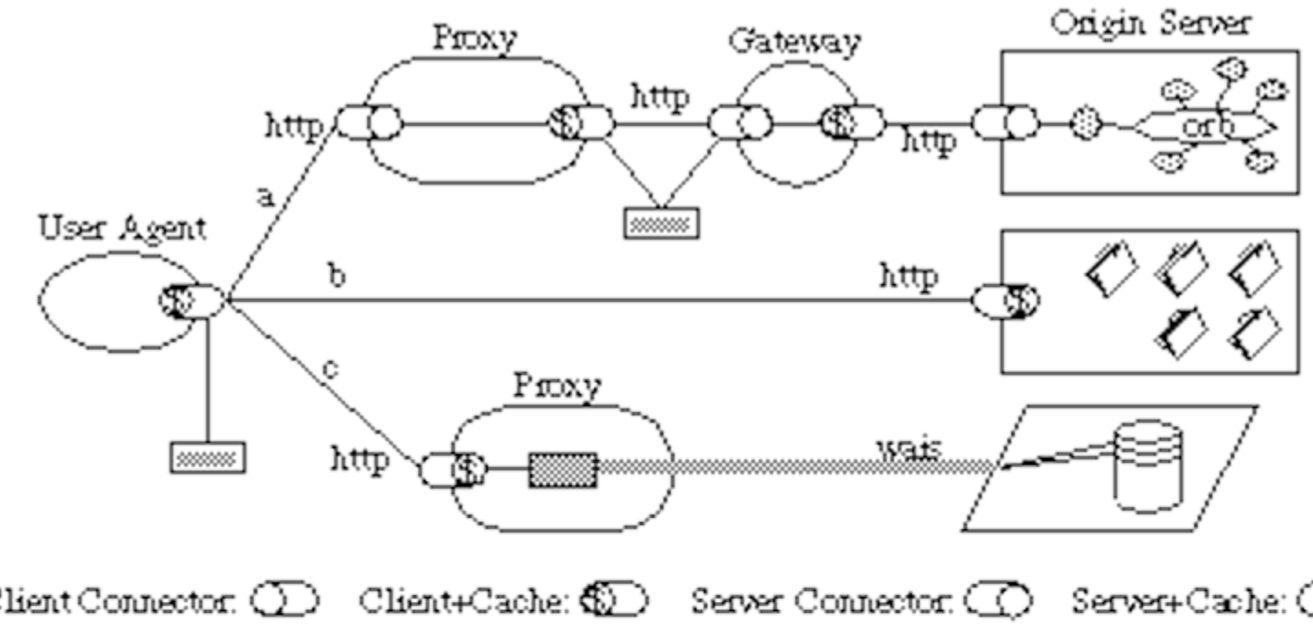


Figure 5-10. Process View of a REST-based Architecture

A user agent is portrayed in the midst of three parallel interactions: a, b, and c. The interactions were not satisfied by the user agent's client connector cache, so each request has been routed to the resource origin according to the properties of each resource identifier and the configuration of the client connector. Request (a) has been sent to a local proxy, which in turn accesses a caching gateway found by DNS lookup, which forwards the request on to be satisfied by an origin server whose internal resources are defined by an encapsulated object request broker architecture. Request (b) is sent directly to an origin server, which is able to satisfy the request from its own cache. Request (c) is sent to a proxy that is capable of directly accessing WAIS, an information service that is separate from the Web architecture, and translating the WAIS response into a format recognized by the generic connector interface. Each component is only aware of the interaction with their own client or server connectors; the overall process topology is an artifact of our view.

按照Fielding的描述，REST的统一接口由4个部分组成：

- 资源的标识
- 通过表述对资源执行的操作
- 自描述的消息
- 以及超媒体作为应用状态引擎（现在通常缩写为HATEOAS）

以HTTP为例

- 资源的标识就是资源的URI
- 资源的表述是资源在特定时刻状态的描述，可以通过在客户-服务器之间传递资源的表述，对资源执行某种操作
- 自描述的消息由一些标准的HTTP方法、可定制的HTTP头信息、可定制的HTTP响应代码组成
- 超媒体就是HTML，可以使用HTML作为引擎，驱动应用状态的迁移。

资源

- REST对于信息的核心抽象是资源。
- 任何能够被命名的信息都能够作为一个资源：
  - 一份文档或
  - 一张图片
  - 一个与时间相关的服务(比如洛杉矶今日的天气)
  - 一个其他资源的集合
  - 一个非虚拟的对象(比如一个人)等等。

- 一篇学术论文的创作者首选的版本
- X会议学报中发表的论文
- 最新版本
- 版本号1.2.7
- 包含有Orange功能实现的修订版本

资源标识符

- REST使用一个资源标识符来标识特定资源
- REST被用来为URI标准定义术语“资源”

REST MISMATCHES IN URI

- 所有的URI中包括标识当前用户的信息

表述

- REST组件通过以下方式在一个资源上执行动作：使用一个表述来捕获资源的当前的或预期的状态，并在组件之间传递该表述。
- 表述的其他常用但不够精确的名称包括：文档、文件、HTTP消息实体、实例或变量。
- 表述由数据、描述数据的元数据、以及（有时候存在的）描述元数据的元数据组成（通常用来验证消息的完整性）。

REST APPLIED TO HTTP

HTTP请求的语义通过请求方法的名称来表示

RESPONSE STATUS CODES

- 100-199表示消息中包含一个临时的信息响应
- 200-299表示请求成功
- 300-399表示请求需要被重定向到另一个资源
- 400 - 499表示客户端发生了一个不应该重复的错误
- 500-599表示服务器端遇到了一个错误，但是客户端稍后可以得到一个更好的响应（或者通过某个其他服务器）

自描述的消息

- 主机
- 分层的编码
- 语义独立性
- 传输独立性
- 尺寸限制
- 缓存控制
- 内容协商

内容协商

REST组件通过转移一种表述来进行通信，REST组件可以基于接收者的能力或者其期待的内容，以及资源的性质来动态地选择不同的表述

REST MISMATCHES IN HTTP

- 区分非权威的响应
- Cookie
- 强制性扩展
- 混合元数据
- MIME语法

HTTP并不是RPC

连接器接口与过程调用在参数和结果的传递方式上有着重要的区别：

- 其传入参数由请求的控制数据、一个表示请求的目标的资源标识符、以及一个可选的表述组成。
- 其传出参数由响应的控制数据、可选的资源元数据、以及一个可选的表述组成。
- 请求是使用具有标准语义的通用的接口定向到资源的，这些语义能够被中间组件和提供服务的来源机器进行解释。结果是使得一个应用支持分层的转换和间接层，并且独立于消息的来源，

HTTP并不是一种传输协议

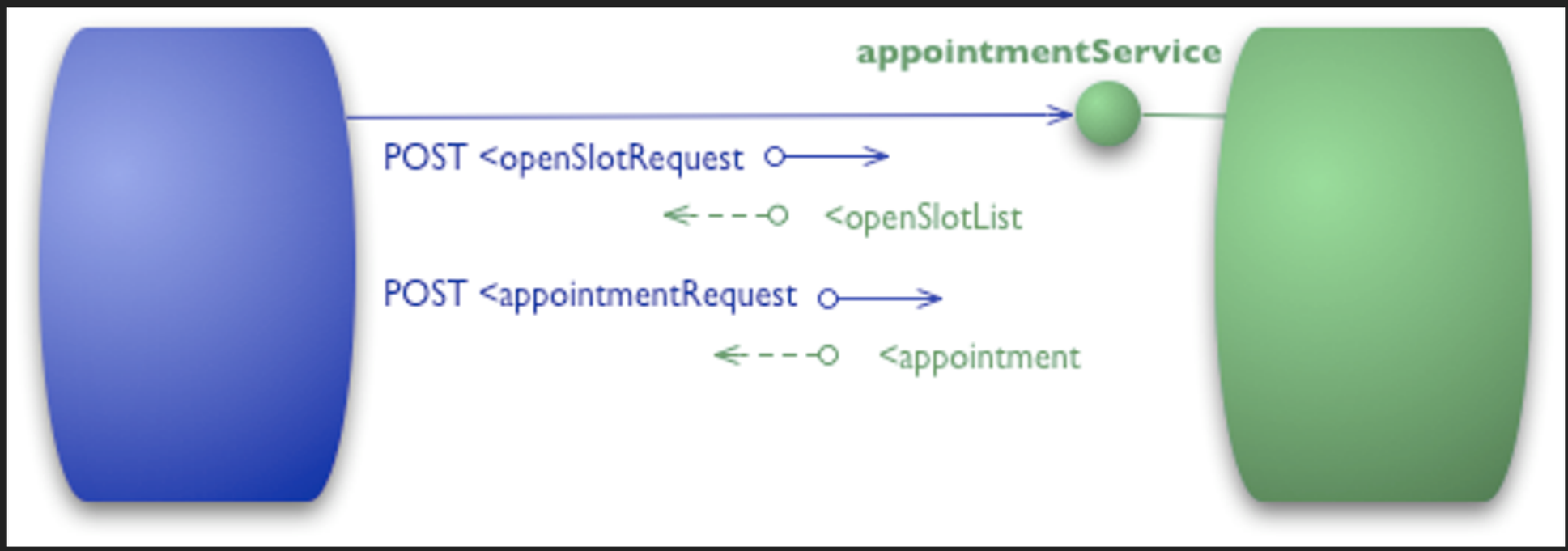
在HTTP协议中，消息通过在那些资源的表述上的转移和操作，来对资源执一些动作，从而反映出Web架构的语义。

RESTFUL APIS

- 遵守REST架构约束的Web service APIs被称作RESTful APIs

RICHARDSON MATURITY MODEL

LEVEL 0 - RPC


appointmentService
POST <openSlotRequest
<openSlotList
POST <appointmentRequest
<appointment

```
POST /appointmentService HTTP/1.1
[various other headers]

<openSlotRequest date = "2010-01-04" doctor = "mjones"/>
```

```
HTTP/1.1 200 OK
[various headers]

<openSlotList>
  <slot start = "1400" end = "1450">
    <doctor id = "mjones"/>
  </slot>
  <slot start = "1600" end = "1650">
    <doctor id = "mjones"/>
  </slot>
</openSlotList>
```

```
POST /appointmentService HTTP/1.1
[various other headers]

<appointmentRequest>
  <slot doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
</appointmentRequest>
```

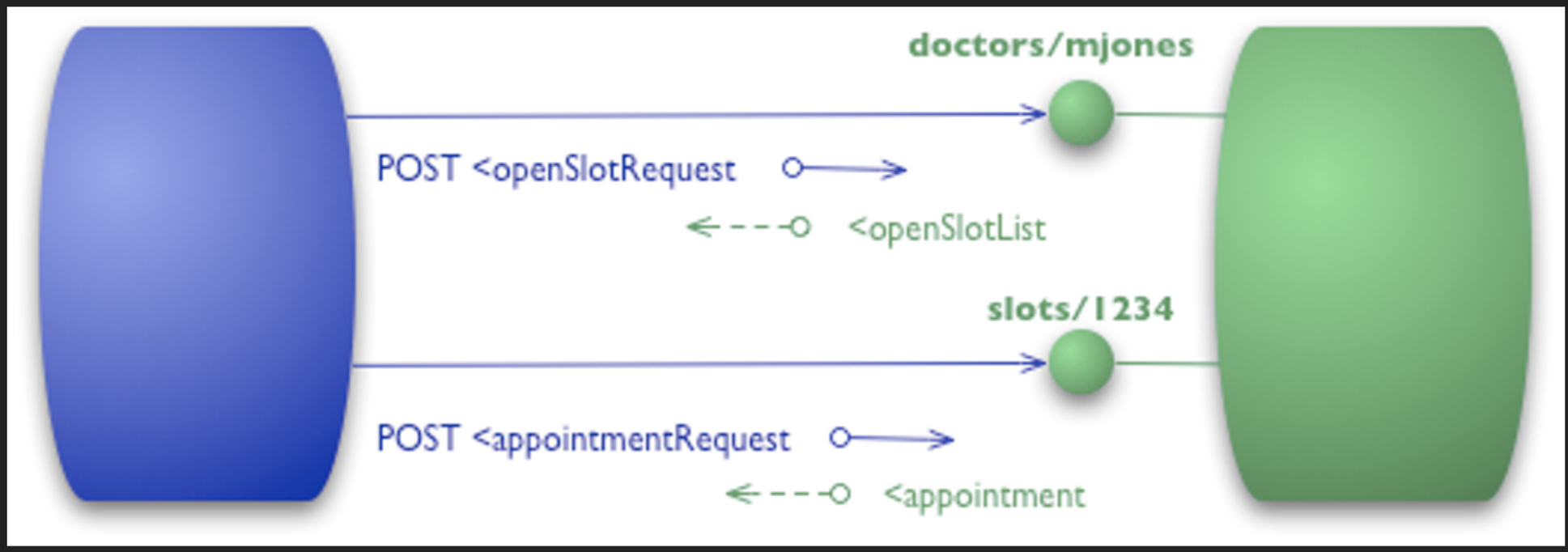
```
HTTP/1.1 200 OK
[various headers]

<appointment>
  <slot doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
</appointment>
```

```
HTTP/1.1 200 OK
[various headers]

<appointmentRequestFailure>
  <slot doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
  <reason>Slot not available</reason>
</appointmentRequestFailure>
```

LEVEL 1 - RESOURCES



```
POST /doctors/mjones HTTP/1.1
[various other headers]

<openSlotRequest date = "2010-01-04"/>
```

```
HTTP/1.1 200 OK
[various other headers]

<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
</openSlotList>
```

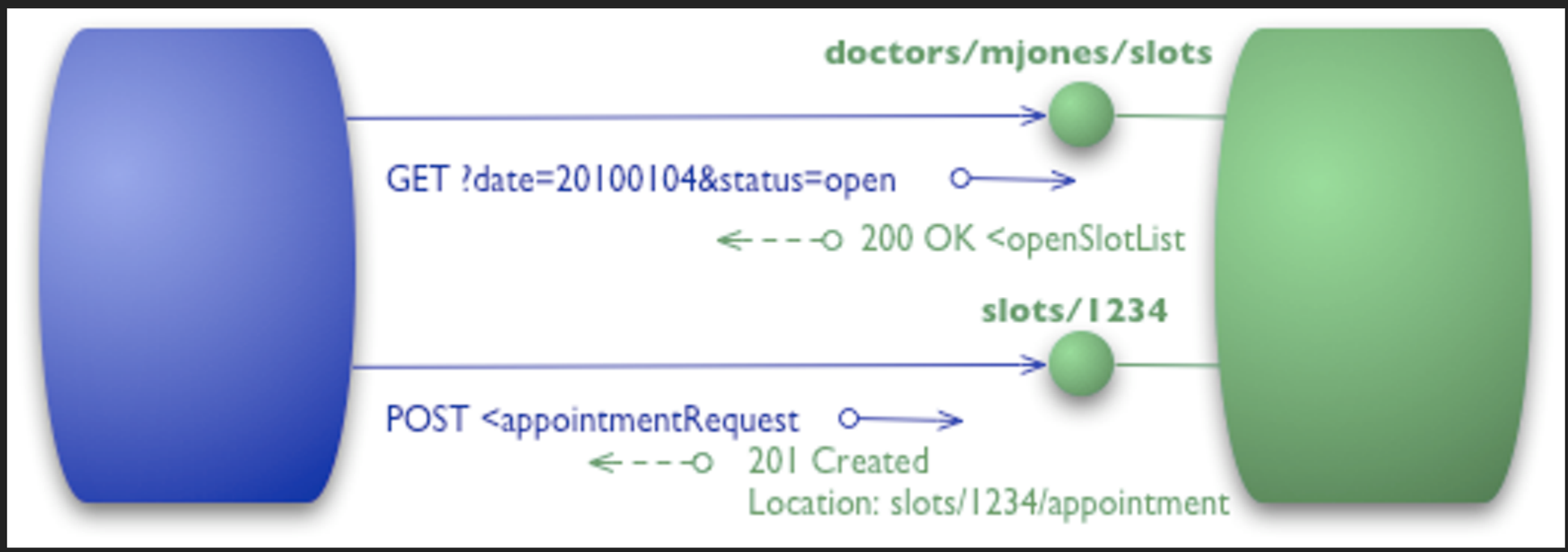
```
POST /slots/1234 HTTP/1.1
[various other headers]

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>
```

```
HTTP/1.1 200 OK
[various headers]

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
</appointment>
```

LEVEL 2 - HTTP VERBS



```
GET /doctors/mjones/slots?date=20100104&status=open HTTP/1.1
Host: royalhope.nhs.uk
```

```
HTTP/1.1 200 OK
[various headers]

<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
</openSlotList>
```

```
POST /slots/1234 HTTP/1.1
[various other headers]

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>
```

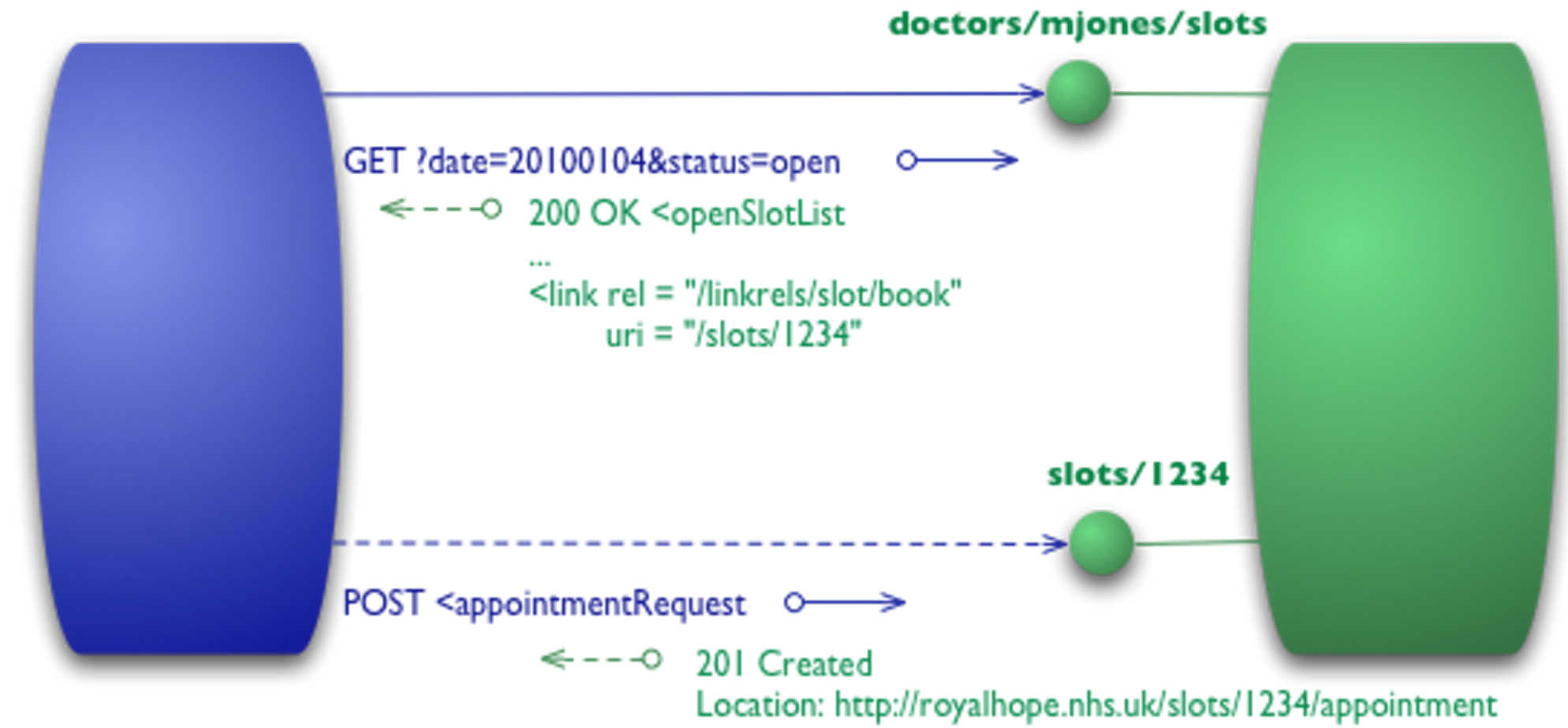
```
HTTP/1.1 201 Created
Location: slots/1234/appointment
[various headers]

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
</appointment>
```

```
HTTP/1.1 409 Conflict
[various headers]

<openSlotList>
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650"/>
</openSlotList>
```

LEVEL 3 - HYPERMEDIA CONTROLS



```
GET /doctors/mjones/slots?date=20100104&status=open HTTP/1.1
Host: royalhope.nhs.uk
```

```
HTTP/1.1 200 OK
[various headers]

<openSlotList>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450">
     <link rel = "/linkrels/slot/book"
           uri = "/slots/1234"/>
  </slot>
  <slot id = "5678" doctor = "mjones" start = "1600" end = "1650">
     <link rel = "/linkrels/slot/book"
           uri = "/slots/5678"/>
  </slot>
</openSlotList>
```

```
POST /slots/1234 HTTP/1.1
[various other headers]

<appointmentRequest>
  <patient id = "jsmith"/>
</appointmentRequest>
```

```
HTTP/1.1 201 Created
Location: http://royalhope.nhs.uk/slots/1234/appointment
[various headers]

<appointment>
  <slot id = "1234" doctor = "mjones" start = "1400" end = "1450"/>
  <patient id = "jsmith"/>
  <link rel = "/linkrels/appointment/cancel"
        uri = "/slots/1234/appointment"/>
  <link rel = "/linkrels/appointment/addTest"
        uri = "/slots/1234/appointment/tests"/>
  <link rel = "self"
        uri = "/slots/1234/appointment"/>
  <link rel = "/linkrels/appointment/changeTime"
        uri = "/doctors/mjones/slots?date=20100104@status=open"/>
  <link rel = "/linkrels/appointment/updateContactInfo"
```

EXAMPLE 1

Resource	Collection URI, such as http://api.example.com/v1/resources/
GET	List the URIs and perhaps other details of the collection's members.
PUT	Replace the entire collection with another collection.
POST	Create a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.
DELETE	Delete the entire collection.

Resource	Element URI, such as http://api.example.com/v1/resources/item17
GET	Retrieve a representation of the addressed member of the collection, expressed in an appropriate Internet media type.
PUT	Replace the addressed member of the collection, or if it does not exist, create it.
POST	Not generally used. Treat the addressed member as a collection in its own right and create a new entry in it.
DELETE	Delete the addressed member of the collection.

EXAMPLE 2

A request for the base resource / might return something like this:

- Request

```
GET /
Accept: application/json+userdb
```

- Response

```
200 OK
Content-Type: application/json+userdb

{
    "version": "1.0",
    "links": [
        {
            "href": "/user",
            "rel": "list",
            "method": "GET"
        },
        {
            "href": "/user",
            "rel": "create",
            "method": "POST"
        }
```

we can find a user list by making another request for /user:

- Request

```
GET /user
Accept: application/json+userdb
```

- Response

```
200 OK
Content-Type: application/json+userdb

{
    "users": [
        {
            "id": 1,
            "name": "Emil",
            "country": "Sweden",
            "links": [
                {
                    "href": "/user/1",
                    "rel": "self",
                    "method": "GET"
                },
                {
```

we can create a new user by POSTing to /user:

- Request

```
POST /user
Accept: application/json+userdb
Content-Type: application/json+userdb

{
    "name": "Karl",
    "country": "Austria"
}
```

- Response

```
201 Created
Content-Type: application/json+userdb

{
    "user": {
        "id": 3,
        "name": "Karl",
        "country": "Austria",
        "links": [
            {
                "href": "/user/3",
                "rel": "self",
                "method": "GET"
            },
            {
                "href": "/user/3",
```

we can change existing data:

- Request

```
PUT /user/1
Accept: application/json+userdb
Content-Type: application/json+userdb

{
    "name": "Emil",
    "country": "Bhutan"
}
```

- Response

```
200 OK
Content-Type: application/json+userdb

{
    "user": {
        "id": 1,
        "name": "Emil",
        "country": "Bhutan",
        "links": [
            {
                "href": "/user/1",
                "rel": "self",
                "method": "GET"
            },
            {
                "href": "/user/1",
```

EXERCISE

为FizzGame设计一套RESTful APIs

- 实现以下功能：
  - 可以显示游戏列表
  - 可以创建一个游戏，创建时指定三个特殊数
- 包括内容：
  - 每个API的请求和响应，包括HTTP头和体
  - 考虑到异常情况

参考实现

```
GET /
Accpect: application/vnd.qixi.fizz+json
```

```
200 OK
Content-Type: application/vnd.qixi.fizz+json

{
  "links": [
    {
      "href": "/fizz-games",
      "rel": "list",
      "method": "GET"
    },
    {
      "href": "/fizz-games",
      "rel": "create",
      "method": "POST"
    }
  ]
```

```
GET /fizz-games
Accpect: application/vnd.qixi.fizz+json
```

```
200
Content-Type: application/vnd.qixi.fizz+json

{
  "fizzGames": [
    {
      "id": 1,
      "specialNumbers": [3, 5, 7],
      "links": [
        {
          "href": "/fizz-games/1",
          "rel": "self",
          "method": "GET"
        },
        {
          "href": "/fizz-games/1",
```

```
POST /fizz-games
Accpect: application/vnd.qixi.fizz+json

{
  "specialNumbers": [4, 5, 6]
}
```

```
201 Created
Content-Type: application/vnd.qixi.fizz+json
Location: /fizz-games/2

{
  "fizzGame": {
    "id": 2,
    "specialNumbers": [4, 5, 6],
    "links": [
      {
        "href": "/fizz-games/2",
        "rel": "self",
        "method": "GET"
      },
      {
        "href": "/fizz-games/2",
```

```
POST /fizz-games
Accpect: application/vnd.qixi.fizz+json

{
  "specialNumbers": [5, 7, 10]
}
```

```
422 Unprocessable Entity
Content-Type: application/vnd.qixi.fizz+json

{
  "error": {
    "message": "the special number should match [1-9]"
  }
}
```

课后练习

- 实现以下功能：
  - 可以显示游戏列表
  - 可以创建一个游戏，创建时指定三个特殊数
  - 在游戏中，可以查看某个学生应该说的话
  - 在游戏中，可以验证某个学生应否应该说Fizz
- 包括内容：
  - 每个API的请求和响应，包括请求和响应的HTTP头和体
  - 考虑到异常情况
- 发Merge Request

参考资料

- Representational state transfer
- Architectural Styles and the Design of Network-based Software Architectures
- 介绍Web基础架构设计原则的经典论文《架构风格与基于网络的软件架构设计》导读
- Richardson Maturity Model
- List of HTTP status codes
- What exactly is RESTful programming?