SUPINFO International University

Sign of success

Representational State Transfer

REST, ROA, and HATEOAS

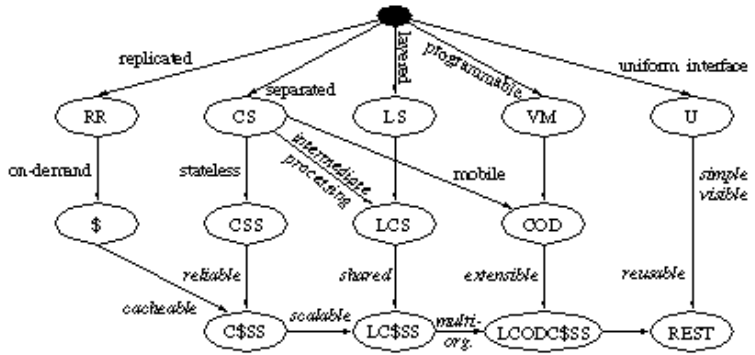


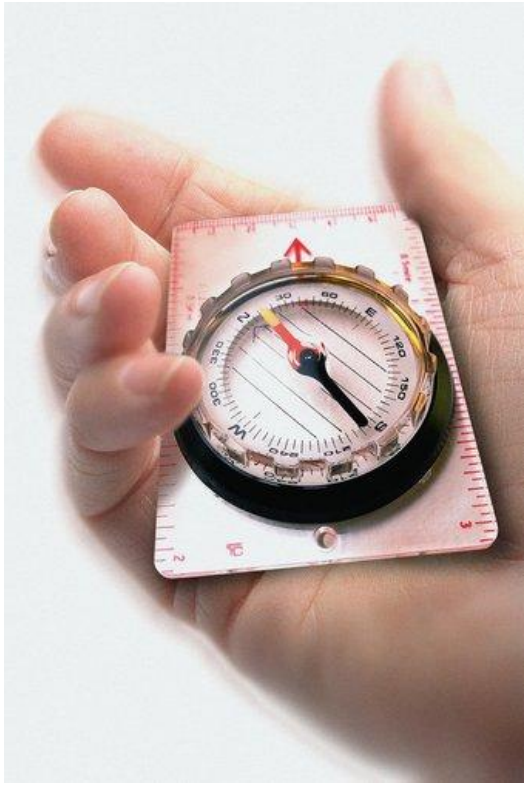
Figure 5-9. REST Derivation by Style Constraints

Course objectives

By completing this course, you will be able to:

- Explain how HTTP works
- Define REST constraints
- Consume Web APIs

Course topics



Course's plan:

- HTTP Reminders
- RESTful Architecture

The Web Protocol

HTTP REMINDERS

The Protocol

- **H**yper**T**ext **T**ransfer **P**rotocol
- Communications protocol developed for Web
- Request/Response protocol
- Stateless

Request Message

- Composed of:
  - A request line composed of:
    - The request method used
    - The resource URI
    - The protocol and the version used
  - Several Headers
  - An empty line
  - An optional message body

HTTP request message

```
POST /en/html/index.html HTTP/1.1
Host: www.website.com
User-Agent: Mozilla/5.0 (Windows;en-GB; rv:1.8.0.11)
Accept:
text/xml,text/html;q=0.9,text/plain;q=0.8,image/png,
*/*;q=0.5
Accept-Language: en-gb,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded
Content-Length: 39

name=MyName&male=yes
```

HTTP request methods

- HTTP defines nine methods (or verbs):
  - **GET**: Request a representation of the resource
  - **POST**: Submit data to be processed to the identified resource
  - **PUT:** Uploads a representation of the specified resource
  - **DELETE**: Deletes the specified resource
  - …

Response Status Codes

- Status codes are divided into five classes :
  - 1xx : Informational
    - Indicates a provisional response
  - 2xx : Success
    - Indicates the request was received, understood, accepted and processed successfully
    - Examples :
      - 200 OK
      - 201 Created

Response Status Codes

- Status codes are divided into five classes :
  - 3xx : Redirection
    - Indicates that further action needs to be taken by the user agent in order to fulfill the request
    - Example :
      - 301 Moved Permanently

Response Status Codes

- Status codes are divided into five classes :
  - 4xx : Client Error
    - Intended for cases in which the client seems to have erred
    - Examples :
      - 403 Forbidden
      - 404 Not Found
      - 405 Method Not Allowed

Response Status Codes

- Status codes are divided into five classes :
  - 5xx : Server Error
    - Indicate the server is aware that it has encountered an error or is otherwise incapable of performing the request
    - Examples :
      - 500 Internal Server Error
      - 501 Not Implemented

Questions ?

Representational State Transfer

RESTFUL ARCHITECTURE

What is REST ?

- The term comes from Roy Fielding's doctoral dissertation
  - **RE**presentational **S**tate **T**ransfer

- REST was initially described in the context of HTTP, but it is not limited to that protocol

What is REST ?

- Roy defines a RESTful system with the following constraints :
  - It must be a client-server system
  - It has to be stateless
  - It has to support a caching system
  - It has to be uniformly accessible
  - It has to be layered (support scalability)
  - It may be able to transfer executable code

Client-Server system

- Separation of concerns
  - Separation between the clients & data storage
  - Simple server components
    - Portability improvement
    - Scalability improvement

Stateless

- Each request must contain all of the information necessary to understand it
- No stored context on the server
- **Session state is on the client**

Cache

- Responses must be implicitly or explicitly labeled as **cacheable** or **non-cacheable**
- Potential intermediary components:
  - Proxies, Shared caches
- Always think about the durability

Uniform Interface (UCCSS)

- Standard format and interactions

- 4 constraints:
  - Identification of resources (URI)
  - Standard representations (media-type)
  - Self-descriptive (semantic, metadata)
  - Hypermedia driven (HATEOAS)

Uniform Interface (UCCSS)

- Advantages
  - Easily interoperable
  - Independent evolvability
- Disadvantages
  - Not optimal for specific interactions

Layered System (ULCCSS)

- Promote substrate independence
- Protect new services from legacy clients
- Enable load balancing

Code-On-Demand

- Optional
- Scripts for the client (applets/JavaScript)
  - Improve system extensibility
- Enable load balancing

Implementations

- HTTP 1.1
- WebDAV (partial)
- APP / Atom
- GData, OData

Resource

- A RESTful resource is anything that is addressable over the Web

- Resource examples are :
  - The temperature in Paris at 8:00 PM
  - A blog post
  - A list of Bug Reports inside a BTS
  - A search result in Google

Representation

- A Representation is typically a document that captures the current or intended state of a resource

- It’s what is sent back and forth between clients and servers

Representation

- Can take various form such as :
  - HTML Document
  - Plain Text
  - XML Stream
  - JSON Stream
  - …
- One resource can have more than one representation
  - But with the same URI

RESTful Web Services

- Also called RESTful web API, it’s a simple web service implemented using HTTP and the principles of REST

- The URI of a resource is available as an hyperlink in the representations

RESTful Web Services

- The HTTP protocol provides methods on which we can map CRUD operations
  - And so apply different actions with the same URI !

Data Action	HTTP protocol equivalent
CREATE	POST
RETRIEVE	GET
UPDATE	PUT
DELETE	DELETE

A Web API Example

- An example of a Web API is available at the following URL :
  - http://restful-example.appspot.com/
- It provides CRUD operations on a student list
- We'll use them as example in the course

GET / Retrieve

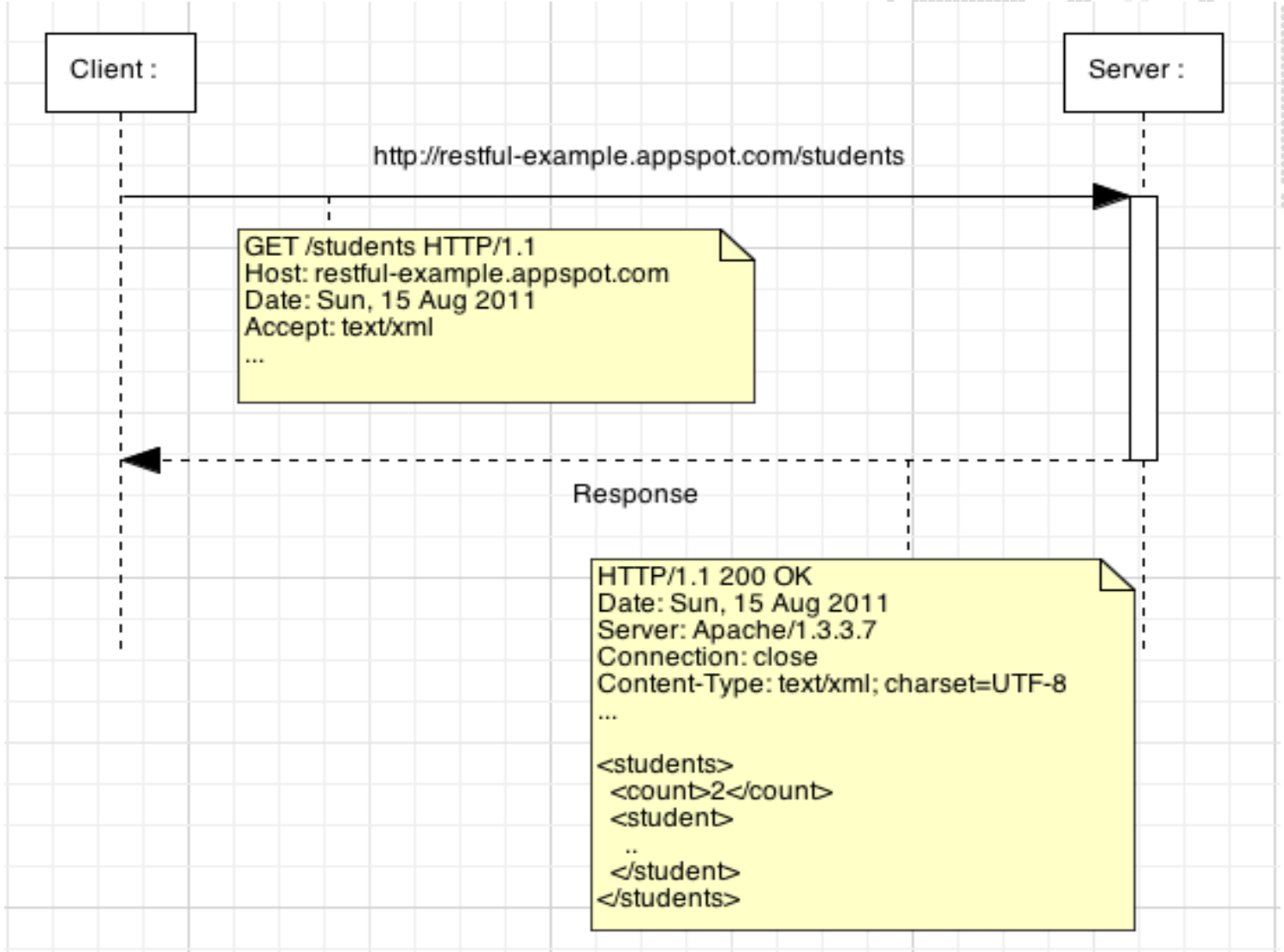
- A GET request to the URI */students* return all the students :

```
<students>
  <count>2</count>
  <student>
    <birthDate>5287-09-15T22:00:13.524Z</birthDate>
    <firstName>Jack</firstName>
    <idBooster>59253</idBooster>
    <lastName>Harkness</lastName>
  </student>
  ...
</students>
```

GET / Retrieve

- And a GET request to the URI */students/59253* returns Jack:

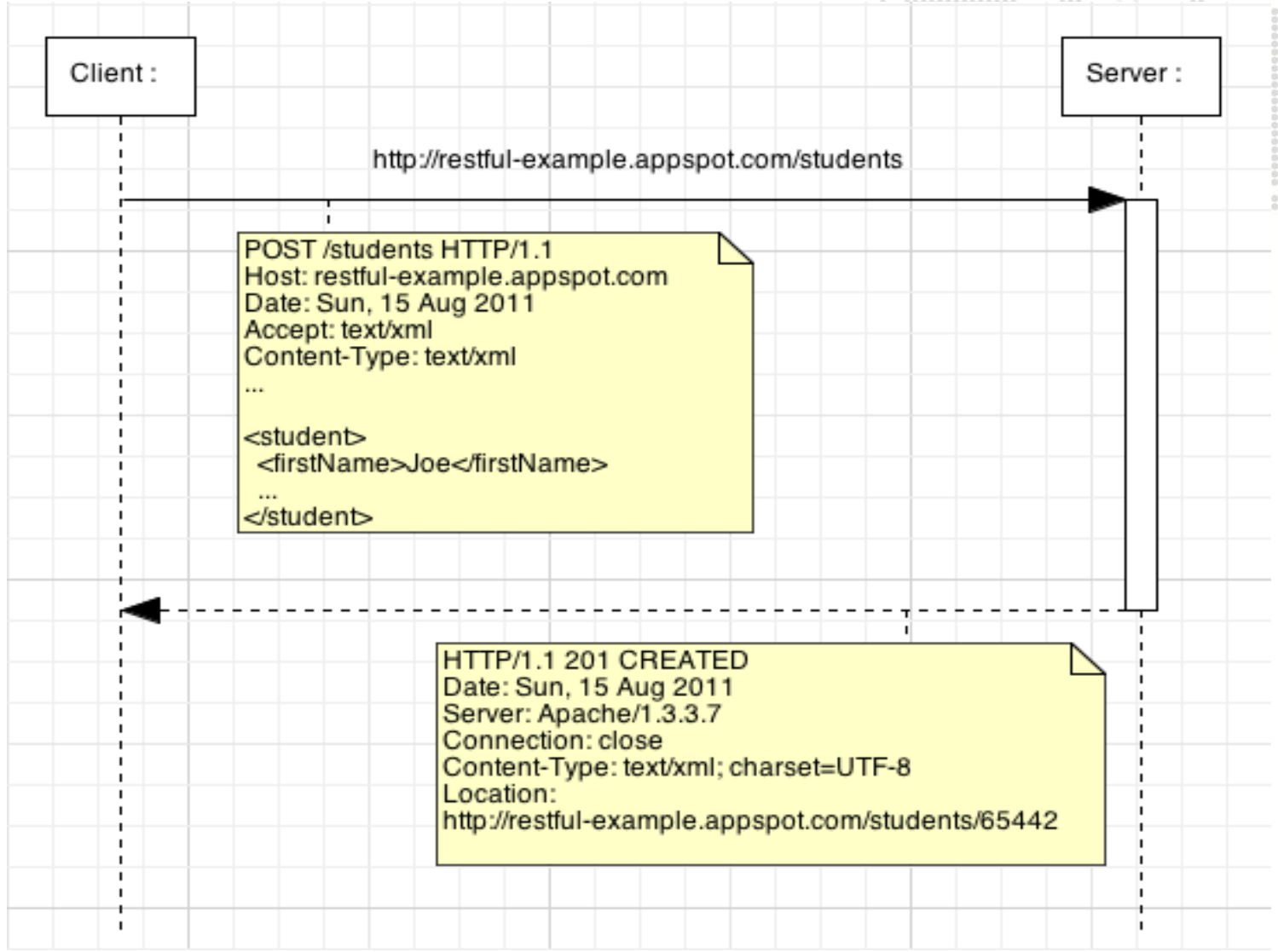
```
<student>
  <birthDate>5287-09-15T22:00:13.524Z</birthDate>
  <firstName>Jack</firstName>
  <idBooster>59253</idBooster>
  <lastName>Harkness</lastName>
</student>
```



POST / Create

- We want to add Joe
  - A Joe XML representation looks like this :

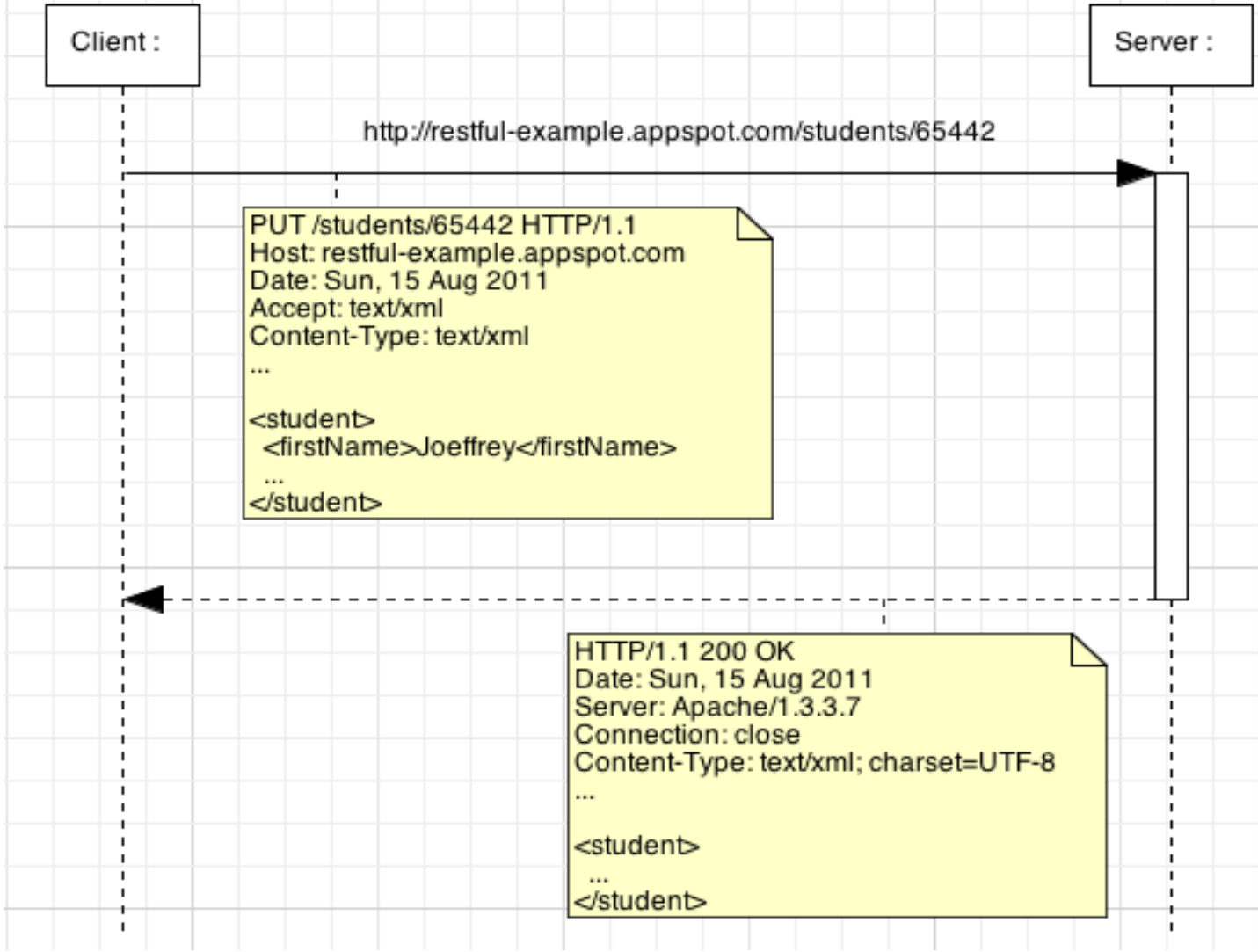
```
<student>
  <birthDate>1946-06-28T22:00:13.524Z</birthDate>
  <firstName>Joe</firstName>
  <idBooster>65442</idBooster>
  <lastName>Dalton</lastName>
</student>
```



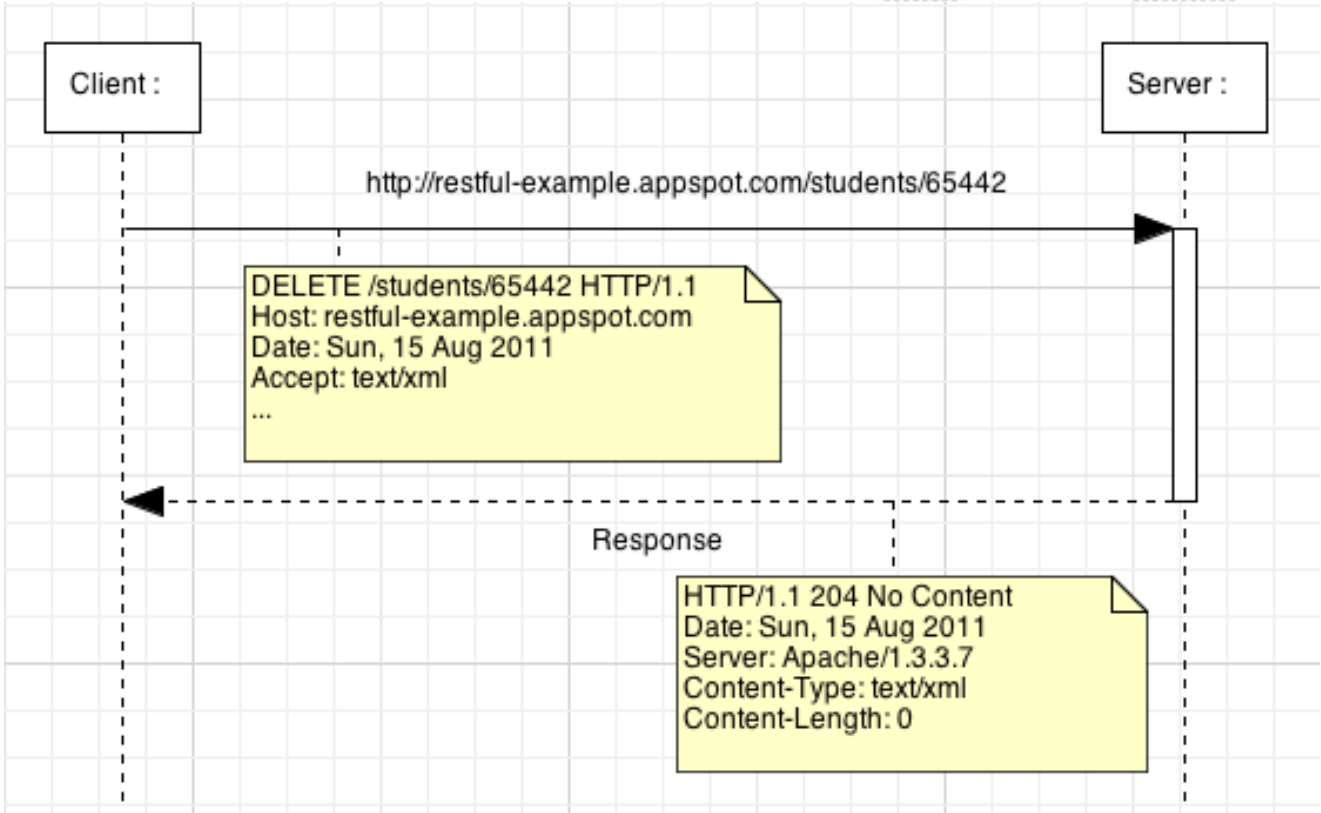
PUT / Update

- We want to update Joe
  - Updated Joe representation looks like this :

```
<student>
  <birthDate>1946-06-28T22:00:13.524Z</birthDate>
  <firstName>Joeffrey</firstName>
  <idBooster>65442</idBooster>
  <lastName>Dalton</lastName>
</student>
```



DELETE / Delete



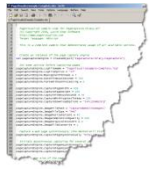
RESTful Web Services

- Web API are not always simple CRUD
  - Choose the verb that best matches to the behavior of your service !

Questions ?

Exercise (1/2)

- Create a Web application that uses the Web API of http://restful-example.appspot.com



Exercise (2/2)

- No need to use a server side technology
  - Just use JavaScript !
- Your application must have the following features:
  - List all the students
  - Add a new student
  - Remove a student
- Use polling to fetch students list updates !