REST Web Services
Week 11 Lecture

Outline

- Examples of published public web services
- Creating REST web services in Expressjs Application
- Consuming REST web services in Expressjs application

What are web services

- Web services is a distributed architectural paradigm for applications
- It provides a simple and open way of **integrating** functions or data from various systems
- It can be used **within** an organization and/or **across** the public Internet
- When it was first proposed, it consists of several basic standards
  - SOAP: A messaging protocol for transferring information
  - WSDL: A model and an XML format for describing Web services
  - UDDI: A registry and protocol for publishing and discovering web services (**not really used!!)**
  - WSDL and UDDI are in tension with the idea of using URI to address web resources
  - Original design of Web Services is very **application centric** in contrast to the **resource centric** Web and REST style.
- The term web services has much broader meaning now
  - At least two implementations: SOAP based vs. RESTful

Typical Use Case of Web Services API

What web services technology achieve can be done using basic network programming

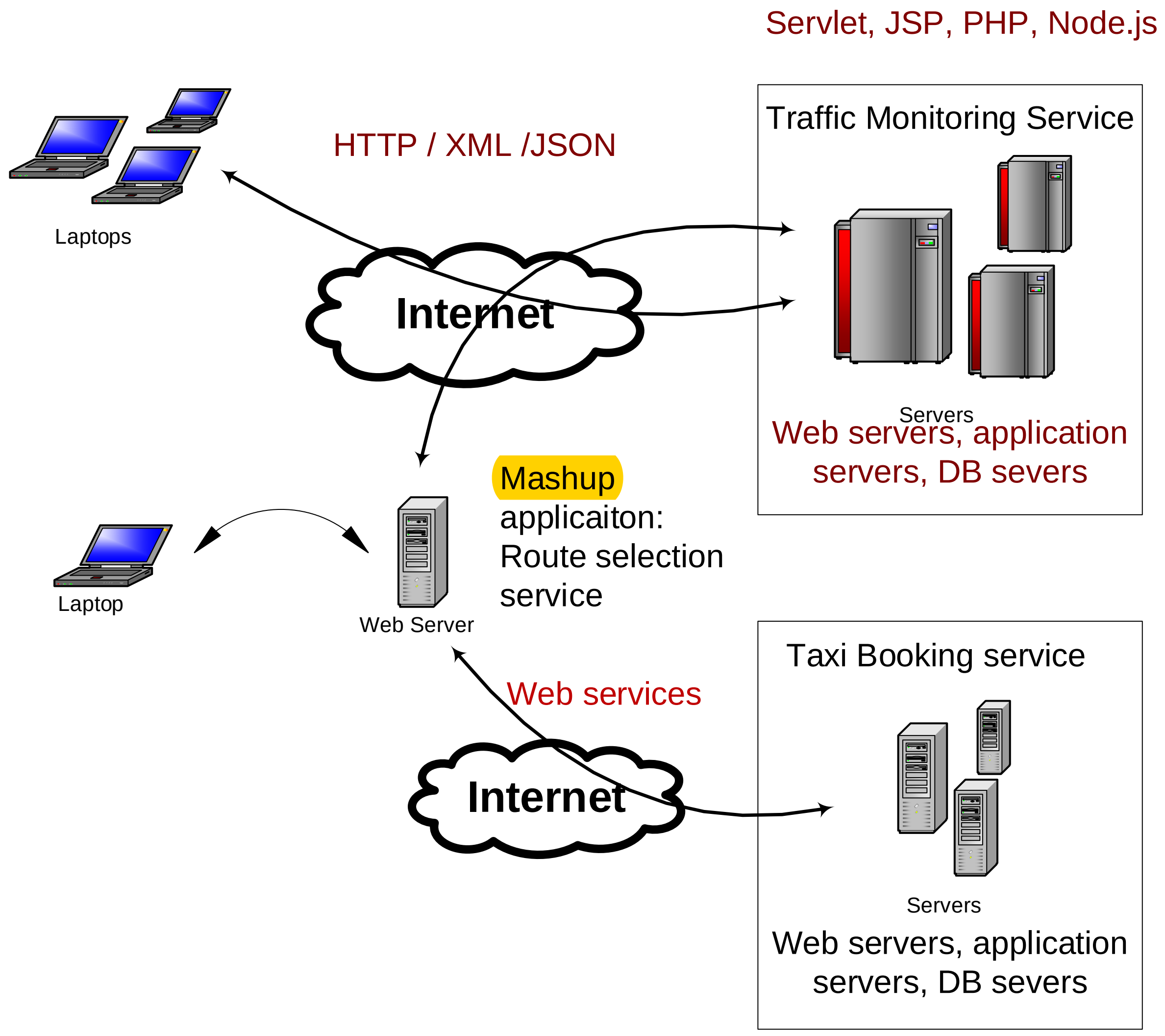
Web services provide a better way of integrating component

Standard communication protocol: HTTP

Relative standard message format: XML, JSON

Ability to utilize existing system/applications

Web services API also provides a convenient way of getting structured data for analysis purpose


Servlet, JSP, PHP, Node.js
Laptops
HTTP / XML /JSON
Traffic Monitoring Service
Internet
Servers
Web servers, application servers, DB severs
Mashup applicaiton: Route selection service
Laptop
Web Server
Web services
Taxi Booking service
Internet
Servers
Web servers, application servers, DB severs

Example Web Service APIs

- Twitter API
  - https://dev.twitter.com/rest/public
- MediaWiki API
  - https://www.mediawiki.org/wiki/API:Main_page
- Flickr API
  - http://www.flickr.com/services/api/
- Amazon product advertising API
  - https://affiliate-program.amazon.com/gp/advertising/api/detail/main.html#details
- New York Times API
  - http://developer.nytimes.com/docs
- Youtube API
  - https://developers.google.com/youtube/getting_started#data_api

MediaWiki API

- **https://en.wikipedia.org/w/api.php**?**action=query**&prop=revisions&rvprop=ids|timestamp&rvstart=2016-12-01T00:00:00Z&rvend=2017-01-01T00:00:00Z&rvdir=newer&format=jsonfm&titles=cat&rvlimit=max

```
{
    "batchcomplete": "",
    "query": {
        "normalized": [
            {
                "from": "cat",
                "to": "Cat"
            }
        ],
        "pages": {
            "6678": {
                "pageid": 6678,
                "ns": 0,
                "title": "Cat",
                "revisions": [
                    {
                        "revid": 752709621,
                        "parentid": 752304215,
                        "timestamp": "2016-12-02T20:51:06Z"
                    },
                    {
                        "revid": 752713783,
                        "parentid": 752709621,
                        "timestamp": "2016-12-02T21:17:08Z"
                    },
```

What is REST

- **Representational State Transfer**
- REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses. Requests and responses are built around the transfer of representations of resources. A resource can be essentially any coherent and meaningful concept that may be addressed. A representation of a resource is typically a document that captures the current or intended state of a resource.

Based on Roy Fielding's doctoral dissertation, rephrased by wikipedia
http://en.wikipedia.org/wiki/Representational_State_Transfer

Early day REST API format

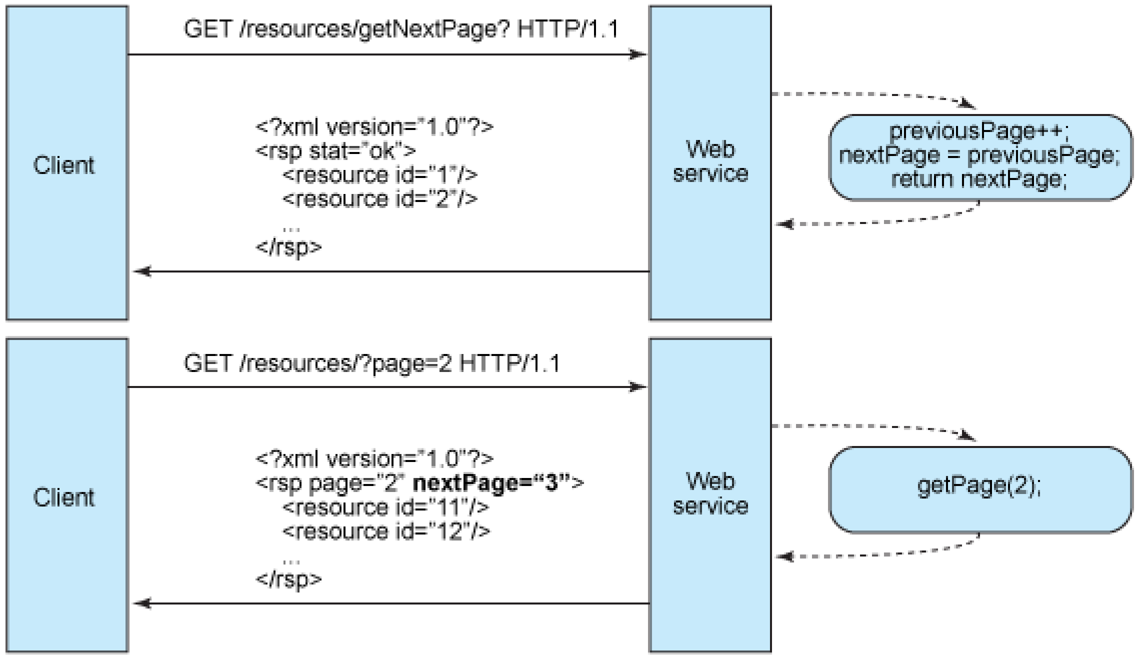
- REST is an architectural style rather than a strict protocol
- The commonly agreed format comes after many APIs have been published and used by large communities
- Many early days RESTful API's URL has a format consists of
  - API end point (a concept coming from SOAP)
  - a parameter to specify the action: query, update, etc..
  - and many action specific parameters
  - Most of which are expressed as query strings
- Many API providers provide API sandbox or API explorer to help developer build the request URL
- Example:
  - **https://api.flickr.com/services/rest/?method=flickr.test.echo&name=value**
  - **https://en.wikipedia.org/w/api.php?action=query**&name=value

Commonly agreed REST API format

- The commonly agreed REST API URL format conforms to general web architecture
  - Using URI (or URL) to specify resources
  - Using HTTP method to indicate action
- A URI (Uniform Resource Identifier) as a resource identifier is one of the central concepts of WWW
  - A predominant use of the World Wide Web is pure information retrieval, where the representation of an available resource, identified by a URI, is fetched using a HTTP GET request without affecting the resource in any way.
- The simplicity and scalability of the Web is largely due to the fact that there are a few "generic" methods (GET, POST, PUT, DELETE) which can be used to interact with any resource made available on the Web via a URI.

Basic REST design principles

- Use HTTP methods explicitly
- Be stateless
  - Address the resources explicitly in the request message



```
<?xml version="1.0"?>
<rsp stat="ok">
   <resource id="1"/>
   <resource id="2"/>
   ...
</rsp>
```

```
<?xml version="1.0"?>
<rsp page="2" nextPage="3">
   <resource id="11"/>
   <resource id="12"/>
   ...
</rsp>
```

- Expose directory structure-like URIs
  - http://www.myservice.org/discussion/topics/{topic}
- Transfer XML, JSON, or both

http://www.ibm.com/developerworks/webservices/library/ws-restful/

Resource types

- Most of the time we can differentiate between collection type of resources and individual resource
  - Revisions and revision
  - Articles and article
- The URL's directory structure is based on that
- This can be nested and it is up to developers to decide the nesting direction
  - /movies/ForrestGump/actors/TomHanks
  - /directors/AngLee/movies/LifeOfPi

Request URLs and methods

Action	URL path	Parameters	Example
Create new revision	/revisions		http://localhost:3000/revisions
Get all revisions	/revisions		http://localhost:3000/revisions
Get a revision	/revisions	revision_id	http://localhost:3000/revisions/123
Update a revision	/revisions	revision_id	http://localhost:3000/revisions/123
Delete a revision	/revisions	revision_id	http://localhost:3000/revisions/123

Request Method	Use case	Response
POST	Add new data in a collection	New data created
GET	Read data from data source	Data objects
PUT	Update existing data	Updated object
DELETE	Delete an object	NULL

Outline

- Examples of published public web services
- Creating REST web services in Expressjs Application
- Consume REST web services in Expressjs application

Create REST API in ExpressJs

- Additional express route feature: route parameters
  - Route parameters are named URL segments that are used to capture the values specified at their position in the URL
  - The values are populated in **`req.params`** object
  - Example
    - Route path: `/users/:userId/books/:bookId`
    - Request URL: `http://localhost:3000/users/34/books/8989`
    - req.params: { "userId": "34", "bookId": "8989" }

```
app.get('/users/:userId/books/:bookId', function (req, res)
{
  res.send(req.params)
})
```

Specifying client data

- Now we have three ways of sending data from client to server
  - Route parameter
    - Route path: `/users/:userId/books/:bookId`
    - Url: `http://localhost:3000/users/34/books/8989`
    - `req.params.useId`
    - `req.params.bookId`
  - Query String
    - url: http://localhost:3000/usersbooks**?**userId=34&bookId=8989
    - `req.query.userId`
    - `req.query.bookId`
  - Request body
    - data {userId:34, bookId:8989} is sent as part of request body
    - if using body-parser middleware
      - `req.body.userId`
      - `Req.body.bookId`

Create REST API using ExpressJs

model

```
RevisionSchema.statics.getByTitle = function(title, callback){
        return this.find({'title':title}).exec(callback)
}
```

controller

```
module.exports.getByTitle=function(req,res){
     title = req.params.title
     Revision.getByTitle(title,function(err,result){
          if (err){
               console.log("Cannot find revisions of title: " + title)
          }else{
               res.json(result)
          }
     })
}
```

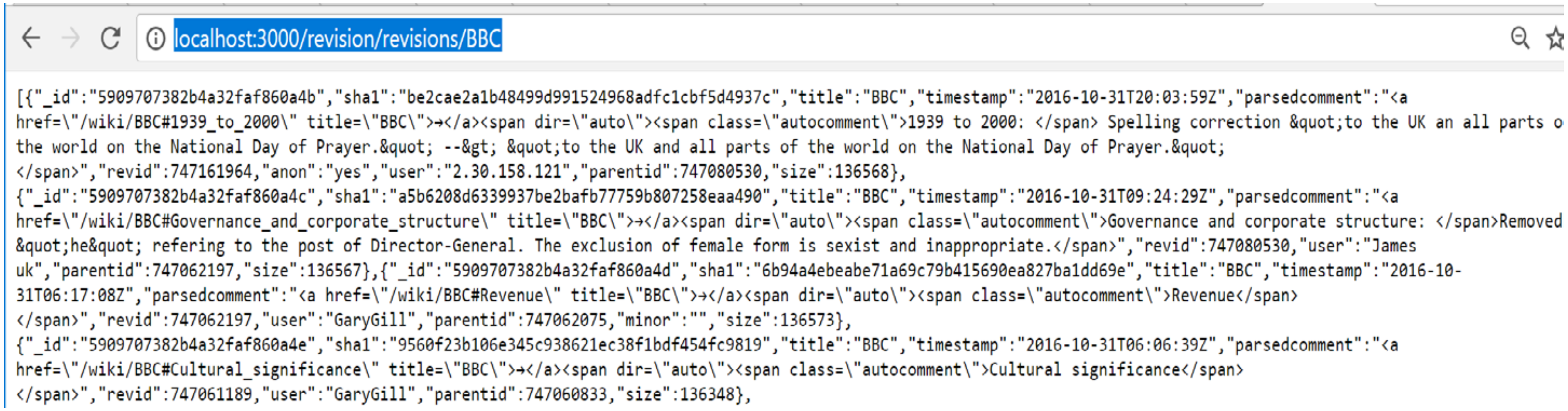
route

```
router.get('/revisions/:title', controller.getByTitle)
```

Response

http://localhost:3000/revision/revisions/BBC

app.use('/revision',revroutes)



localhost:3000/revision/revisions/BBC

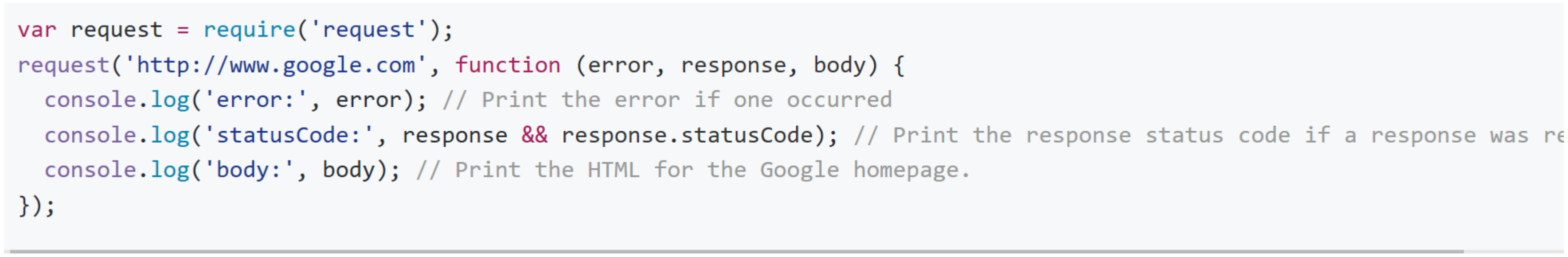
```
[{"_id":"5909707382b4a32faf860a4b","sha1":"be2cae2a1b48499d991524968adfc1cbf5d4937c","title":"BBC","timestamp":"2016-10-31T20:03:59Z","parsedcomment":"<a
href=\"/wiki/BBC#1939_to_2000\" title=\"BBC\">→</a><span dir=\"auto\"><span class=\"autocomment\">1939 to 2000: </span> Spelling correction &quot;to the UK an all parts o
the world on the National Day of Prayer.&quot; --&gt; &quot;to the UK and all parts of the world on the National Day of Prayer.&quot;
</span>","revid":747161964,"anon":"yes","user":"2.30.158.121","parentid":747080530,"size":136568},
{"_id":"5909707382b4a32faf860a4c","sha1":"a5b6208d6339937be2bafb77759b807258eaa490","title":"BBC","timestamp":"2016-10-31T09:24:29Z","parsedcomment":"<a
href=\"/wiki/BBC#Governance_and_corporate_structure\" title=\"BBC\">→</a><span dir=\"auto\"><span class=\"autocomment\">Governance and corporate structure: </span>Removed
&quot;he&quot; refering to the post of Director-General. The exclusion of female form is sexist and inappropriate.</span>","revid":747080530,"user":"James
uk","parentid":747062197,"size":136567},{"_id":"5909707382b4a32faf860a4d","sha1":"6b94a4ebeabe71a69c79b415690ea827ba1dd69e","title":"BBC","timestamp":"2016-10-
31T06:17:08Z","parsedcomment":"<a href=\"/wiki/BBC#Revenue\" title=\"BBC\">→</a><span dir=\"auto\"><span class=\"autocomment\">Revenue</span>
</span>","revid":747062197,"user":"GaryGill","parentid":747062075,"minor":"","size":136573},
{"_id":"5909707382b4a32faf860a4e","sha1":"9560f23b106e345c938621ec38f1bdf454fc9819","title":"BBC","timestamp":"2016-10-31T06:06:39Z","parsedcomment":"<a
href=\"/wiki/BBC#Cultural_significance\" title=\"BBC\">→</a><span dir=\"auto\"><span class=\"autocomment\">Cultural significance</span>
</span>","revid":747061189,"user":"GaryGill","parentid":747060833,"size":136348},
```

Outline

- Examples of published public web services
- Creating REST web services in Expressjs Application
- Consuming REST web services in Expressjs application

Consume REST API in ExpressJS

- Complex API calls may benefit from using a package that wrap up the API
- Simple GET type of queries can always be implemented using general modules designed for handling http requests
  - Core **node.js** modules: **http**, **https**
  - **request** module
- The **request** module (https://github.com/request/request)
  - To install
    - npm install request –save
  - To make a request: **request(options, callback)**



```
var request = require('request');
request('http://www.google.com', function (error, response, body) {
  console.log('error:', error); // Print the error if one occurred
  console.log('statusCode:', response && response.statusCode); // Print the response status code if a response was re
  console.log('body:', body); // Print the HTML for the Google homepage.
});
```

Sample code with request module

```
var request = require('request');

var wikiEndpoint = "https://en.wikipedia.org/w/api.php",
    parameters = ["action=query",
                  "format=json",
                  "prop=revisions",
                  "titles=australia",
                  "rvstart=2016-11-01T11:56:22Z",
                  "rvdir=newer",
                  "rvlimit=max",
                  "rvprop=timestamp|userid|user|ids"]
var url = wikiEndpoint + "?" + parameters.join("&")
console.log("url: " + url)
var options = {
    url: url,
    Accept: 'application/json',
    'Accept-Charset': 'utf-8‘
}
```

End point

Action and all parameters

Constructing an URL

Request header

Making request

```
request(options, function (err, res, data){
    if (err) {
        console.log('Error:', err);
    } else if (res.statusCode !== 200) {
        console.log('Status:', res.statusCode);
    } else {
        json = JSON.parse(data);
        pages = json.query.pages
        revisions = pages[Object.keys(pages)[0]].revisions
        console.log("There are " + revisions.length + " revisions.");
        var users=[]
        for (revid in revisions){
            users.push(revisions[revid].user);
        }
        uniqueUsers = new Set(users);
        console.log("The revisions are made by " + uniqueUsers.size + " 
    unique users");
    }
});
```

Send request

Call back function

Convert JSON format string into JavaScript object

Object.keys(obj) returns a array of obj's property names. We only need the first one.

https version

```
var https = require('https')

var wikiEndpointHost = "en.wikipedia.org",
    path = "/w/api.php"
    parameters = ["action=query",
        "format=json",
        "prop=revisions",
        "titles=australia",
        "rvstart=2016-11-01T11:56:22Z",
        "rvdir=newer",
        "rvlimit=max",
        "rvprop=timestamp|userid|user|ids"],
   headers = {
        Accept: 'application/json',
        'Accept-Charset': 'utf-8'
   }

var full_path = path + "?" + parameters.join("&")

var options = {
     host: wikiEndpointHost,
     path: full_path,
     headers: headers}
```

https://nodejs.org/api/https.html

https version (cont'd)

```
https.get(options,function(res){
    var data ='';
    res.on('data',function(chunk){
       data += chunk
    })
    res.on('end',function(){
        json = JSON.parse(data);
        pages = json.query.pages
        revisions = pages[Object.keys(pages)[0]].revisions
        console.log("There are " + revisions.length + " revisions.");
        var users=[]
        for (revid in revisions){
            users.push(revisions[revid].user);
        }
        uniqueUsers = new Set(users);
        console.log("The revisions are made by " + uniqueUsers.size + " unique users");
  })
}).on('error',function(e){
    console.log(e)
})
```

If the response contains a large body, the **data** event may fire multiple times each with a chunk of the actual body. We need to concatenate all chunks. See lecture 6 slide on form data.

end means no more data, the rest of the processing is the same

The 'error' event fires on the request object, not the incomingMessage **res**

Admin

- There will be a quiz tonight starting from 7:30pm
- It is closed book, paper based
  - 5 minutes reading time
  - 1 hour writing time
- Please check Elearning for venues and seat allocation
- Please do not ask invigilators any content related question
- If you have doubt about certain question, write it down next to the question
- Write your answers on the space provided
  - If you use extra page
    - make sure you have your name and SID on the page
    - insert it in the quiz script