Service Based Approach
Wrap up
WS Slides #6

Content

- Designing a RESTful API
- Application design
- Mater detail interface
- Pagination
- Caching
- Conditional GET and PUT
- OAuth

Designing a REST API

```
Get todos for a list
GET api/todolist/:id/todos

Add new todo to specific list
POST api/todolist/:id/todos

Get all todolists
GET api/todolists

Get specific todolists
GET api/todolist/:id

Delete specific todolists
DELETE api/todolist/:id

Create new list
POST api/todolist
```

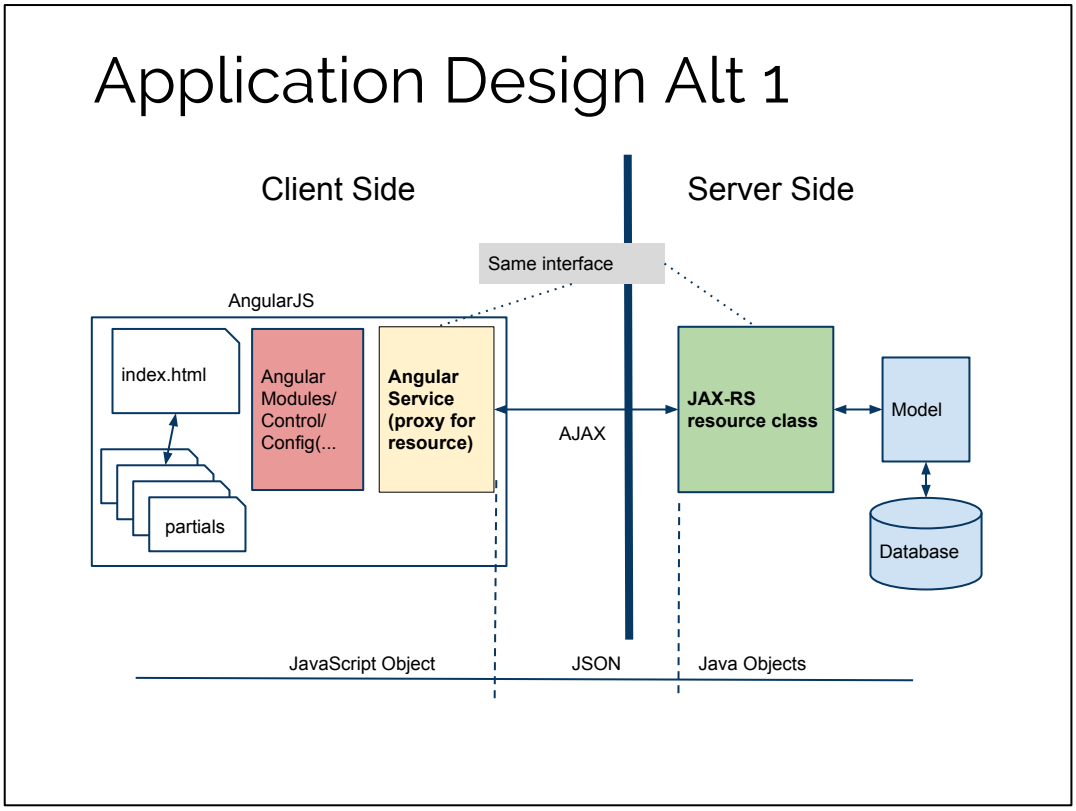
Designing the backend. CRUD operations not too hard but others?

- This also should solves clean URI's

Restful URI Design

- The URLs should be to resources (nouns), not actions (verbs)
- .. much more see link

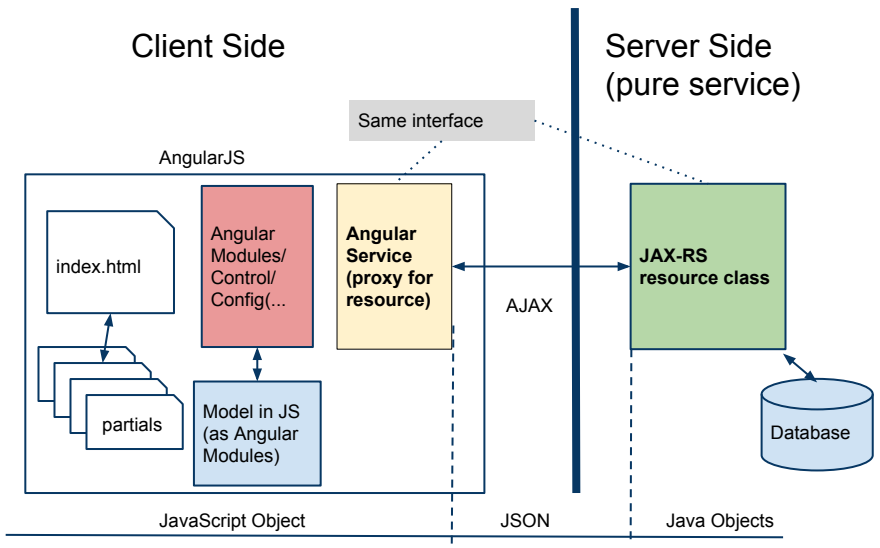
Design RESTful API



Alt 1

- Model on server side (thin client)
- Resource class connecting client and model (wrapping the model)
- Client has a (remote) proxy for the resource class
- Client code calls proxy

Application Design Alt 2



Alt 2 (not recommend)

- Model on client (fat client)
- Server side just handling persistence (wrap database)

Master Detail Interface

```
// app.js
$routeProvider.when('/persons/:id', {
   templateUrl: 'partials/person-detail.html',
   controller: 'PersonDetailCtrl'
})...

// controllers.js
function($scope, $routeParams, ... ) {
    ...find($routeParams.id)
         .success(function(person) {
             $scope.person = person;  // Got person with id
                }).error(function() {
            console.log("selectByPk: error");
        });

// In master page
<td><a href="#/persons/{{person.id}}">{{person.fname}}</a></td>
```

Solved by Angular

- Specification of partial page and controller using $routeProvider, so detail page will be shown using the controller
- Access id of selected detail from $routeParams in controller
- In master page access id from binding and expression

Pagination

```
// control.js
... controller('PersonListCtrl'...){
   $scope.currentPage = 0



// In list page (master)
<button ... ng-disabled="currentPage >=
                            count / pageSize - 1"
    ng-click="currentPage = currentPage + 1">
   Next
</button>
```

Pagination using Angular and JS

- currentPage attribute of $scope
- Using Angular directive in page

Caching

" … If, every time a browser processed a URI beginning http://..., it actually fired off a GET at a server, and if, every time a GET hit a server, the server actually recomputed and sent the data, the Web would melt down PDQ. There is a lot of machinery available, on both the client and server side, to detect when the work of computing results and sending them over the network can be avoided. Normally, we use the term "**caching**" to refer to all this stuff."

// Tim Bray

Caching

- Improve speed, because we want to deliver fast content to our consumer
- Fault tolerance, because we want our service to deliver content also when it encounters internal failures
- Scalability, because the WWW scales to billions of consumers through hypermedia documents and we just want to do the same thing
- Reduce server load, because we don't want our servers to compute without the need of it

Types of caches

- Local cache, your browser's local copy
- Proxy cache, a copy on some server on the way to the origin (the original Server), a middleman
- Content Delivery Network (CDN), large distributed system of servers deployed in multiple data centers in the Internet. The goal of a CDN is to serve content to end-users with high availability and high performance (example: Akamai)

Caching Tutorial

Cache Control Header

Cache-Control:	Description
private	A cache mechanism may cache this page in a Private cache and resend it only to a single client. This is the default value. Most proxy servers will not cache pages with this setting.
public	Shared caches, such as proxy servers, will cache pages with this setting. The cached page can be sent to any user.
no-cache	Do not cache this page at all, even if for use by the same client.
no-store	The response and the request that created it must not be stored on any cache, whether shared or private. The storage inferred here is non-volatile storage, such as tape backups. This is not an infallible security measure.
max-age	How long resource is valid

```
HTTP/1.1 200 OK
Cache-Control: private, no-store, max-age=300
```

HTTP response Cache-Control , some few parameters in slide

- Header values set by server.

Example

- Only client may cache
- Must not be stored on disk

Cache introduces inconsistency!

- Resource previously served to a consumer is different from the one actually held by the server

HTTP Caching in Java with JAX-RS (samples upcoming)

Conditional GET

First Request

```
GET /rest_backend/webresources/cond/11 HTTP/1.1
...
Accept: */*

// Response
HTTP/1.1 200 OK
...
ETag: "053fde96fcc4b4ce72d7739202324cd49"
```

Second Request (conditional)

```
GET /rest_backend/webresources/cond/11 HTTP/1.1
If-None-Match: "053fde96fcc4b4ce72d7739202324cd49"

// Response
HTTP/1.1 304 Not Modified
ETag: "053fde96fcc4b4ce72d7739202324cd49"
```

Conditional GETs (this is latest JAX-RS, we do it a bit different)

- A conditional GET*) method requests that the entity be transferred only under the circumstances described by the conditional header field(s). The conditional GET method is intended to reduce unnecessary network usage by allowing cached entities to be refreshed without requiring multiple requests or transferring data already held by the client
- Headers are: If-Modified-Since, If-Unmodified-Since, If-Match, If-None-Match, or If-Range (If-* headers)

*) Also conditional updates (PUT)

ETag

- An ETag is an opaque identifier assigned by a web server to a specific version of a resource found at a URL.
- Example:

Conditional PUT

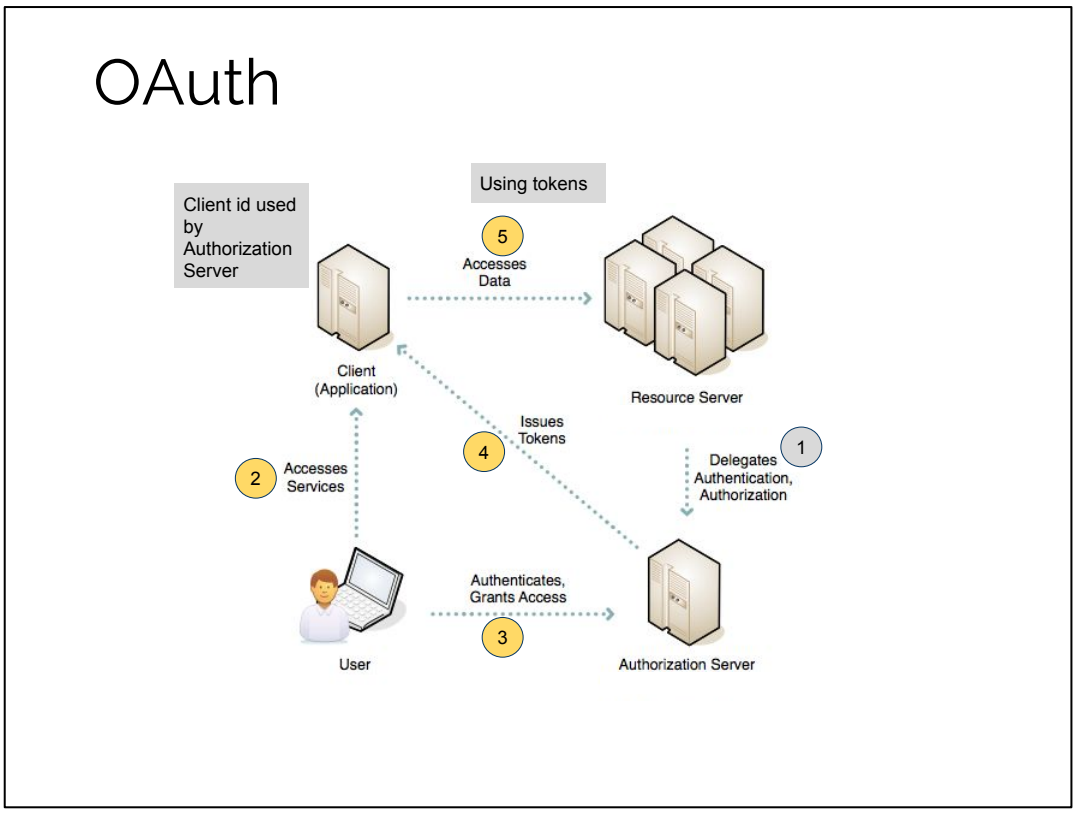
First Request

```
GET /rest_backend/webresources/cond/11 HTTP/1.1
...
Accept: */*

// Response
HTTP/1.1 200 OK
...
ETag: "053fde96fcc4b4ce72d7739202324cd49"
```

Put Request (conditional)

```
PUT /rest_backend/webresources/cond/11 HTTP/1.1
If-Match: "053fde96fcXXX"    (other ETag value)

// Response
HTTP/1.1 412 Precondition Failed
```

If ETag on server doesn't match sent ETag (i.e. dirty data on client)



REST should be stateless, can't store credentials (name/password) in session
- Solution: Send something (preferable time limited) representing the credentials with each request
- One possibility, use HTTP basic authentication (request header) and HTTPS (no cookies)

Mashup applications
- The application should be able to call other applications (on behalf of actual application or actual user, possibly restricted calls)
- Must have single sign-on for user

Possible solution using external Auth/Auth, OAuth

Inspection if picture in slide
Client
- The client (the third-party application) is the application that is attempting to get access to the user's account. It needs to get permission from the user before it can do so.

Resource Server
- The resource server (the API server) is used to access the user's information.

Resource Owner
- The resource owner (the user) is the person who is giving access to some portion of their account.

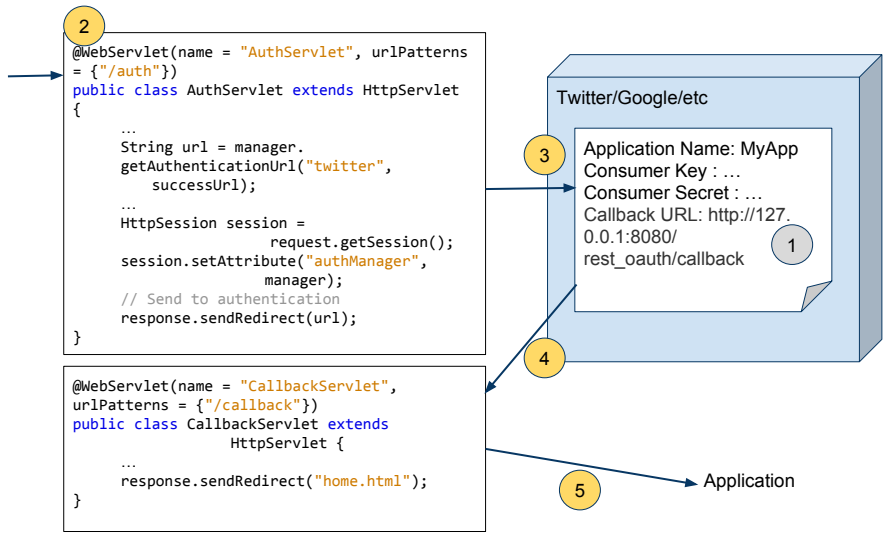
"For example, an end-user (resource owner) can grant a printing service (client)

access to her protected photos stored at a photo-sharing service (resource server), without sharing her username and password with the printing service. Instead, she authenticates directly with a server trusted by the photo-sharing service (authorization server), which issues the printing service delegation specific credentials (access token)."//rfc6749

OAuth

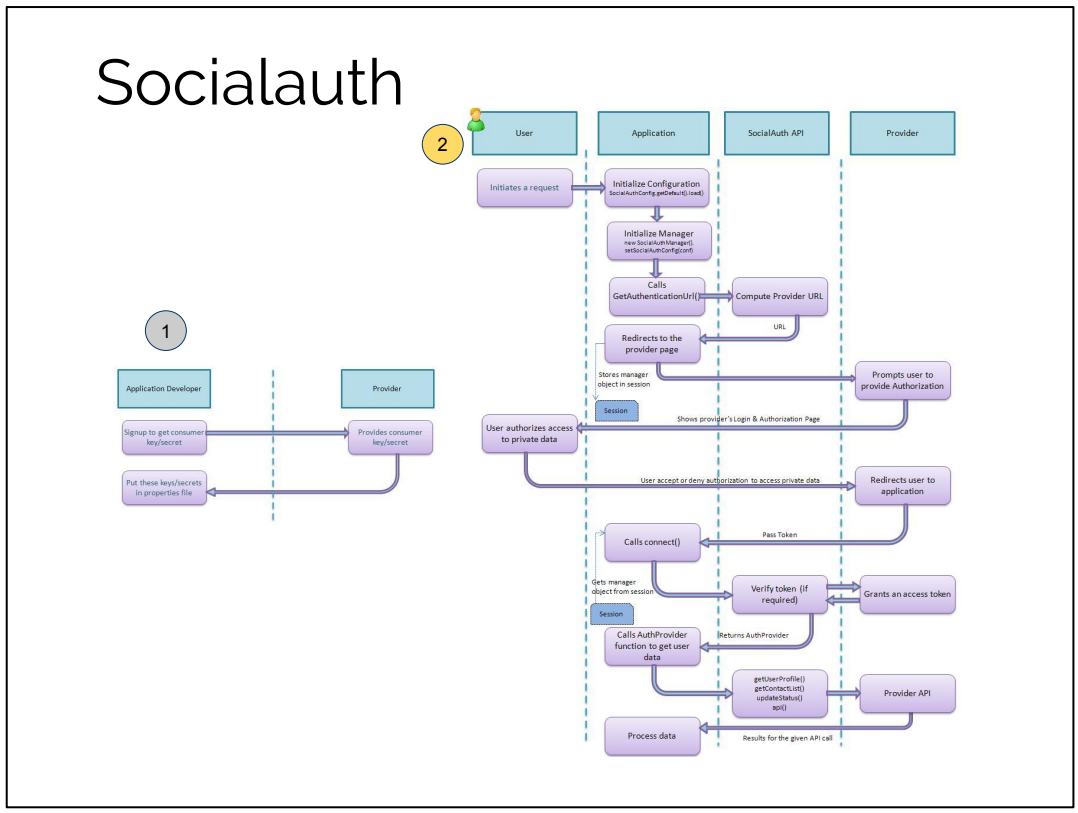
- There are version 1.0 and 2.0 not compatible
- There is severe criticism of 2.0
- Situation confusing …

OAuth and JEE



Overview over sample code, se rest_oauth

- Application registered at Twitter, Google, etc



[Socialauth](#) is a generic library for OAuth