A Service Based Approach

WS Slides #5

Characterization Review

Application composed of language/platform independent resources

No traditional API, just resources accessible by URLs
- Possible a mashup, compose application of external resources

Current trend: RESTful single page application using HTML5 is very hot

Putting it Together

Need some architecture/design

-The resources
- How to distribute the responsibilities, Client vs Server

Also

- Separation of concerns (many techniques, “languages” involved)

Issues

- Quality … testing?
- Authorization (optional)

Unobtrusive JavaScript

Unobtrusive JavaScript =
HTML + CSS + JavaScript in a disciplined manner

- No JavaScript, CSS in HTML
- Style/layout in CSS
- JavaScript in *.js files
- Using AngularJS for this will work out nicely

We always use unobtrusive style! (solved by Angular, except possibly some very small "glue" JS snippets)

Resource URLs

Identification of resources

- Which URLs to which resources? Which methods?
- Hard, similar to OO design (if unlucky we end up with a bad design)

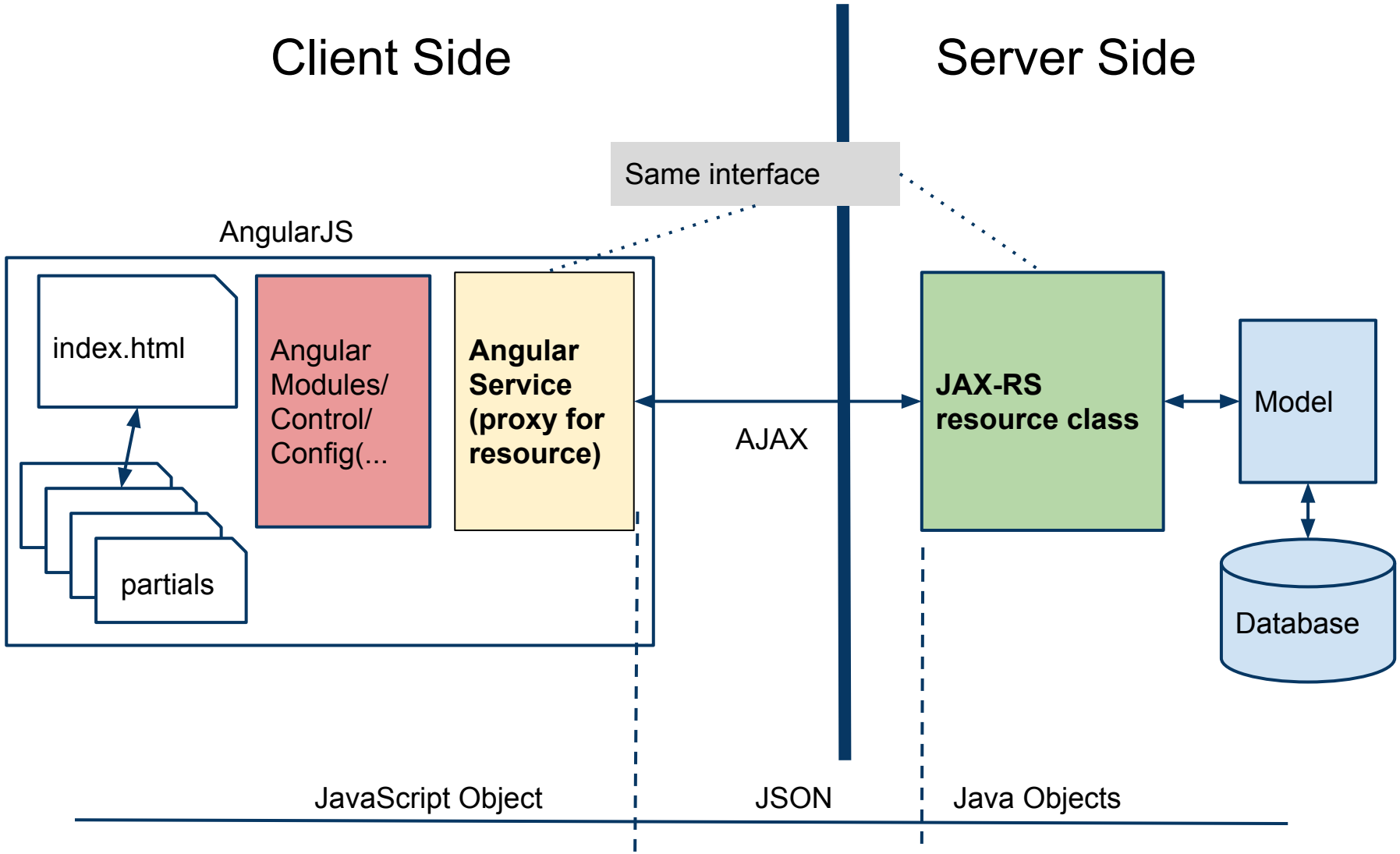
Example: Web Service for books and music

```
ex.org/review?type=cd&title=help
ex.org/cd-review?title=help
ex.org/review/cd?title=help
ex.org/review/cd/help/beatles
GET seems ok, but what about PUT, POST?
```

The debate: No verbs in URI's

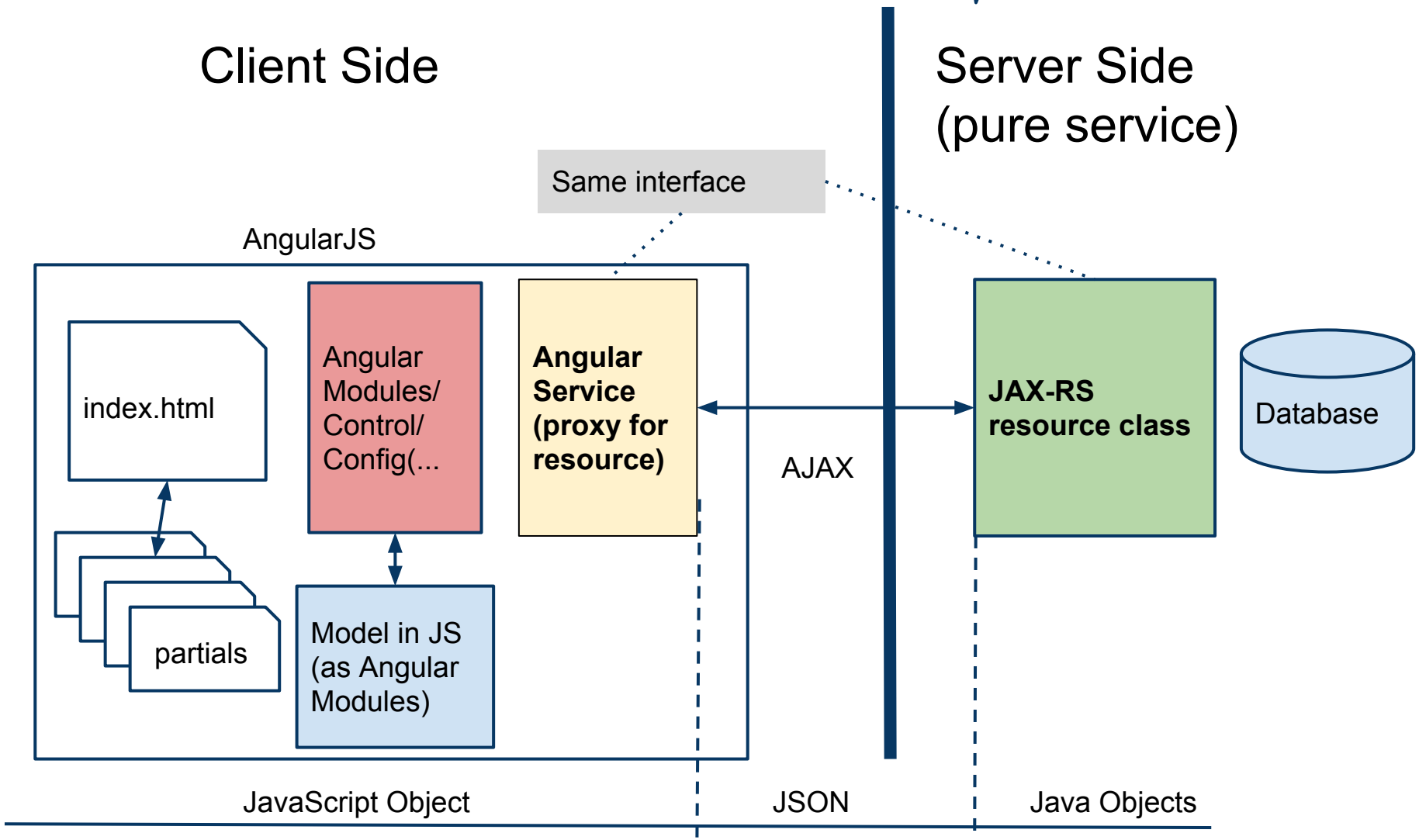
- Resources are not operations (they are nouns)
- Operations are GET, POST, ...)

System Architecture I



System Architecture II

For the brave



REST PRG pattern

No full page returned for POST

- Single page application, probably no problem
- After POST page reload/backbutton have no impact
- Normally navigate away from any POST form (to confirmation or alike)

The "Master/Detail" Problem

Using router, routeParams, links and scoped data for selection

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

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


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

User Navigation

Navigation in long lists (spread over many pages)

```
// In JS
$scope.currentPage = 0

// In Page (expressions possible)
<button … ng-disabled="currentPage >= count / pageSize - 1"
          ng-click="currentPage = currentPage + 1"> Next </button>
```

Also : Possible higher level JS components for tables

JavaScript Testing

JS development heavily dependent on testing, normally need large test suites

Many possibilities we (Angular) use Jasmine
- See workshop skeleton code for use

Authentication and Authorization

REST should be stateless, can't store credentials (name/password) in session

- Solution: Send something (time limited) representing the credentials with each request
- Possible with HTTP basic authentication (request header) and SSL

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

OAuth Confusion

Situation seems very confusing
- Versions: OAuth 1.0 (1.0A a revision of 1.0) and OAuth 2.0 (not backward compatible).
- Are spec really finished (?)
- Spec leader (and others) resigned from OAuth 2.0
- Some API's use 1.0 (Twitter) some use 2.0 (Facebook, Google, Twitter)
- Marked as simple to use ... at least not for a newbie...

Introduction OAuth 1.0
OAuth 2.0 tutorial

Example OAuth with REST

Using SocialAuth library

Before use a bit of setup
- Need account on Twitter
- Need to register your application with Twitter
- See Twitter Developers

HATEOAS

More of a logical design of application (HCI)

- User should be led through application presented with possible next actions

Technique

- I.e embed all links useful for client in the response
- Using Atom-links (XML based format for lists of related information)
- Links embedded in XML document (JSON, …) or …
- … using HTTP Response “link” header (if link in response don't need to parse to get the link)

JEE HATEOAS

Not much support in JAX-RS (HATEOAS defined by application)

REST Documentation

Article series