ObjectRiver
METADATA COMPILERS

Angular6 Microservices

Experiences, Demo & Details

Steven Lemmo @ObjectRiver

Introduction

I'm not selling anything, it's all Free!

- Introduction
  - Steven Lemmo. Java, Salesforce, Backend, Middleware Architect.
  - ObjectRiver is a boutique consulting company that leverages its metadata compiler technology to differentiate it's consulting services. **Consulting solutions are now freely available from objectriver.net**
  - Demo shows Angular 6.1 REST integration to Node/Docker.
  - Details show Angular architecture, and TypeScript interaction with the json layer.
  - Experiences, Problems solved along the way.

ObjectRiver
METADATA COMPILERS

1st attempt 2014

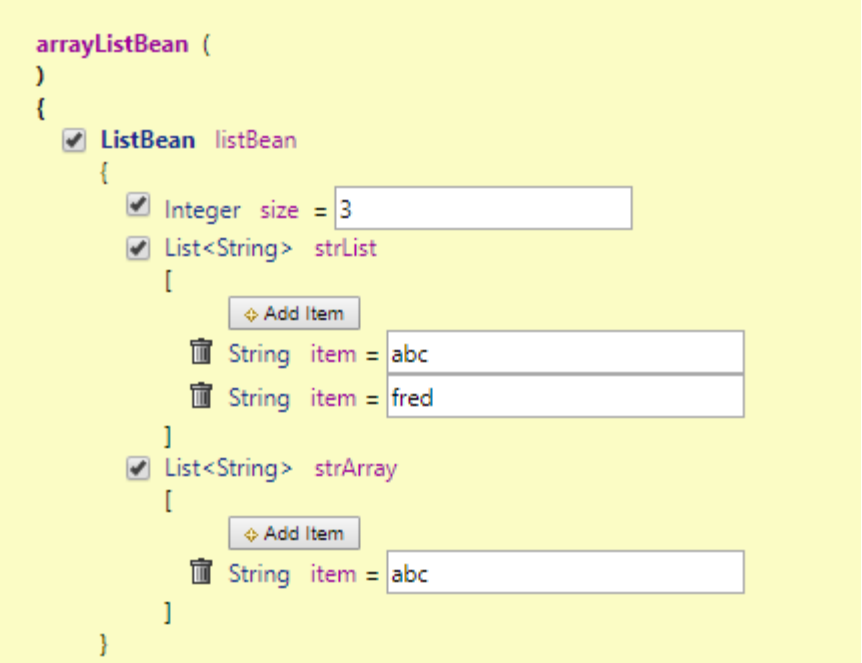
I tried this.

- 2014 Built Angular WebSocket's -> Node Express/Java
  - We got it going!
  - Never used it.
    - WebSocket/JS is not portable across browser
    - JavaScript ☹ Promises ☹
      - We tried chaining promises ☹ ☹
    - HTML pages were kind of amateur.
    - Couldn't use WebSocket binary
    - No bi-directional multiplexing

2nd attempt 2018

Then I tried this

- Angular6 REST -> Docker/Node & Java.
- 6 months nights and weekends Angular6.1 immersion.
  - 1st time I almost gave up
  - Built HTML pattern and it didn't look doable.



Angular 6.1

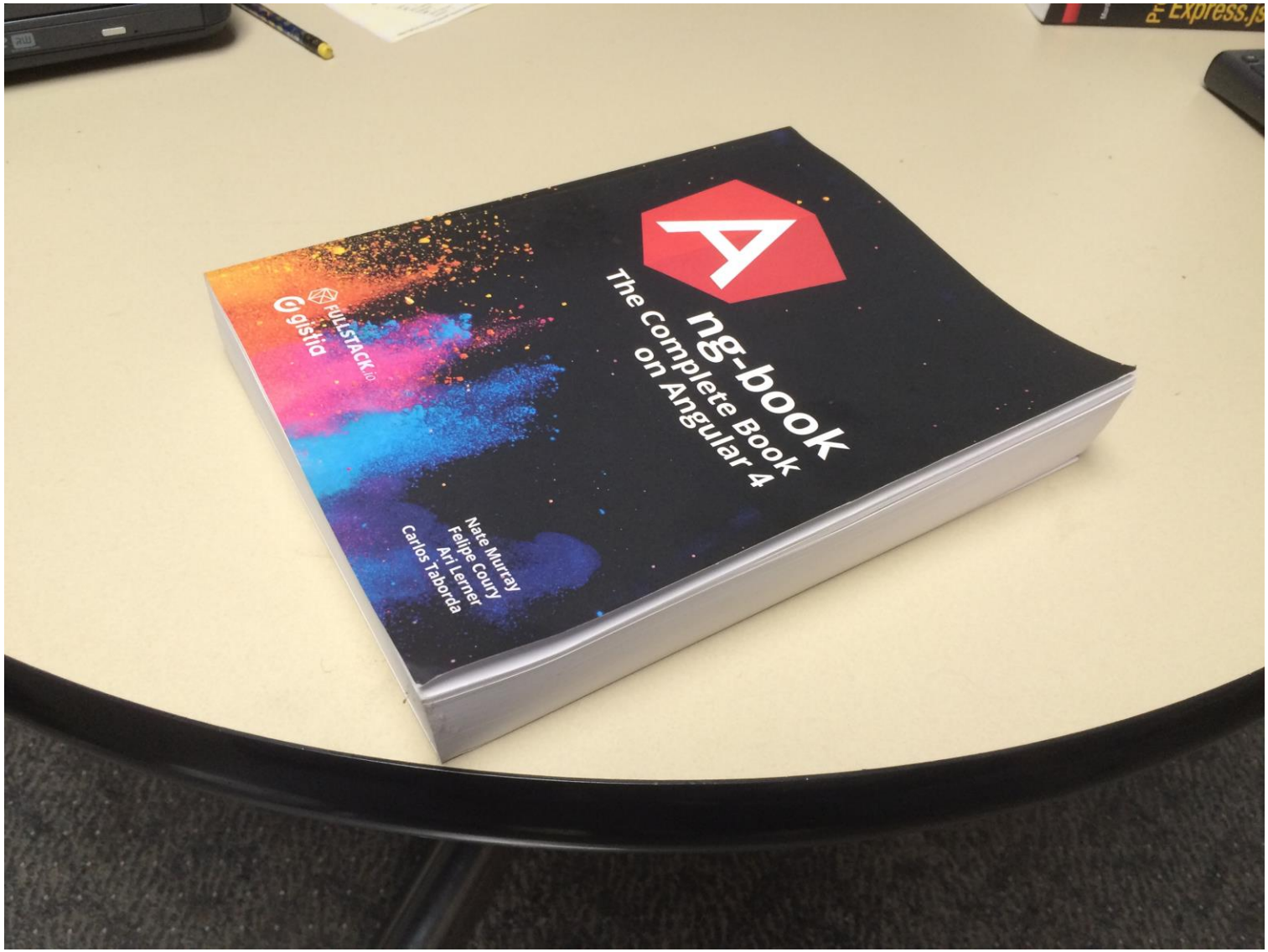
AngularJS !== Angular 6.1

- Reboot – Old prototype is AngularJS and worthless.
- WOW, I've got to learn Angular all over again!

ObjectRiver
METADATA COMPILERS

Don’t buy this book!

For the Newbies



ObjectRiver
METADATA COMPILERS

Watch these YouTubes

I watched all of these

Angular 7 Tutorial - 23 - Routing and Navigation
Codevolution • 106K views

Angular 6 Basics 25 - Building an Angular project
Java Brains • 10K views

OBSERVABLES, OBSERVERS & SUBSCRIPTIONS | RxJS TUTORIAL
Academind 132K views • 1 year ago

RxJS Observables are subscribed by Observers...Wait...what? Let's understand how that all w

Target Lady: Meets Her First
Saturday Night Live 6.1M views • 5 y

Target Lady (Kristen Wiig), whose odd be
Bryant) and thinks up ...

ObjectRiver
METADATA COMPILERS

Success & Demo

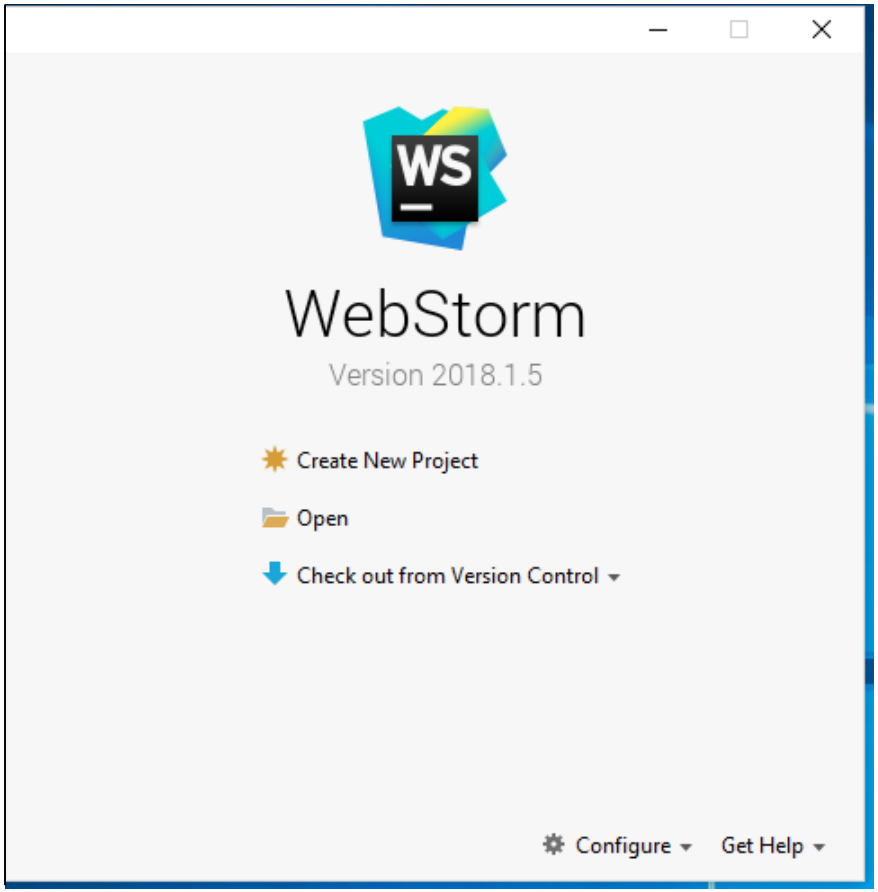
Why it works

- CLI integration with browser
- Angular 6 is modular.
- Typescript! It's Java! 
- json2typescript package open source

ObjectRiver
METADATA COMPILERS

IDE's

Intellij & WebStorm



ObjectRiver METADATA COMPILERS

Demo

Metadata REST definition

```
Application Beans = com.companyname {
    Endpoint BeanIface = 1 {
        Enumeration MyMessageType { Empty=1, Full=2 };
        Exception BeanException {};
        Bean TestBean {
            String[] member1;
            Integer member2;
        };

        Server Singleton {
            @Path("beanMethod")
                @Produces(JSON)
                @Consumes(JSON)
                @POST  Boolean beanMethod(
                    @In TestBean arg,
                    @In TestBean[] argArray,
                    @In MyMessageType option
                ) raises BeanException;

            @Path("beanInReturn")
                  @Produces(JSON)
                  @Consumes(JSON)
                  @POST  TestBean beanInReturn(@In TestBean arg) raises BeanException;

            @Path("beanJustReturn")
                  @Produces(JSON)
                  @POST  TestBean beanJustReturn() raises BeanException;
              };
          };
    };
};
```

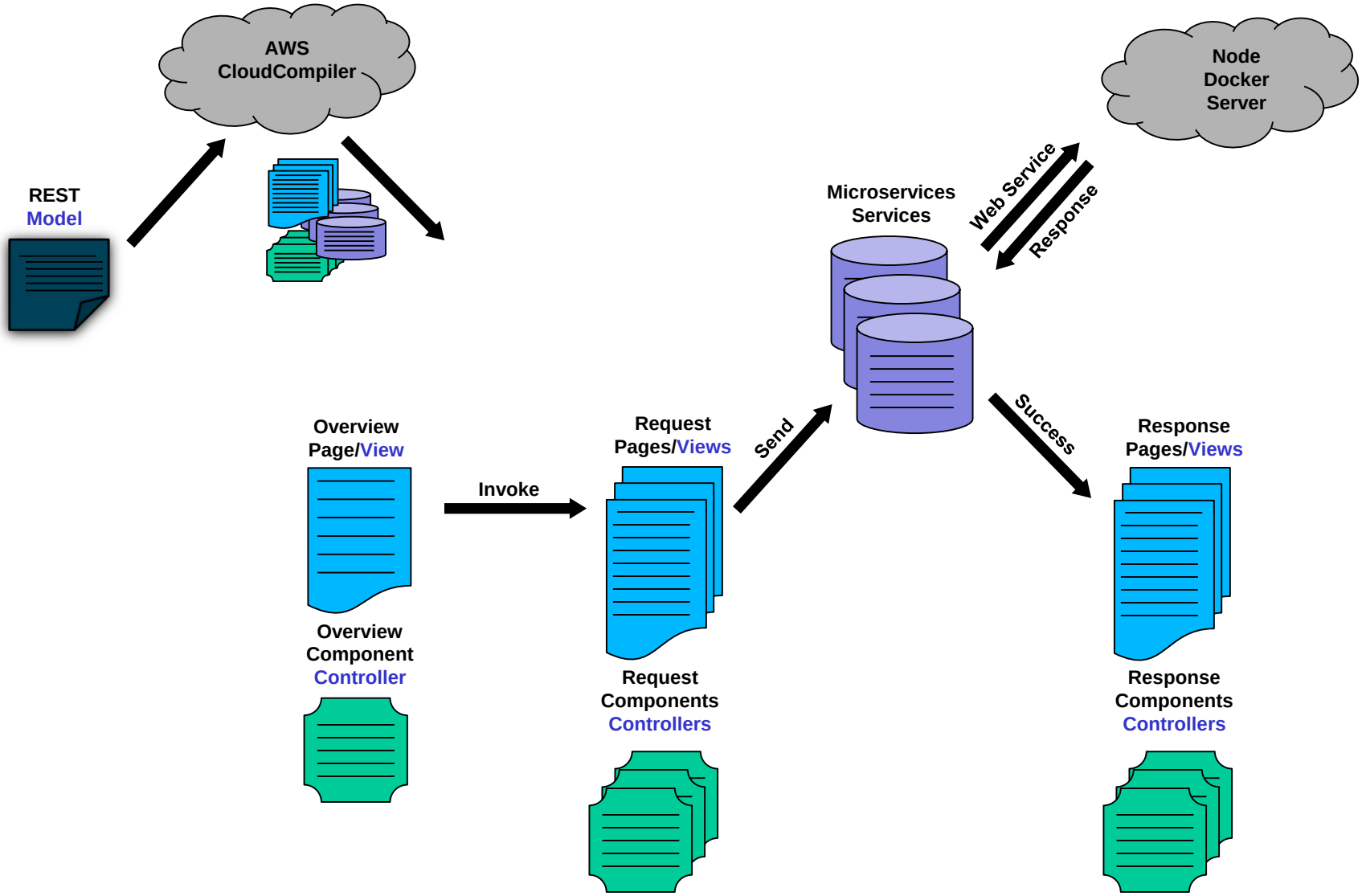
ObjectRiver
METADATA COMPILERS

Demo

Angular6 → Node/Docker

Architecture

Model/View/Controller Architecture

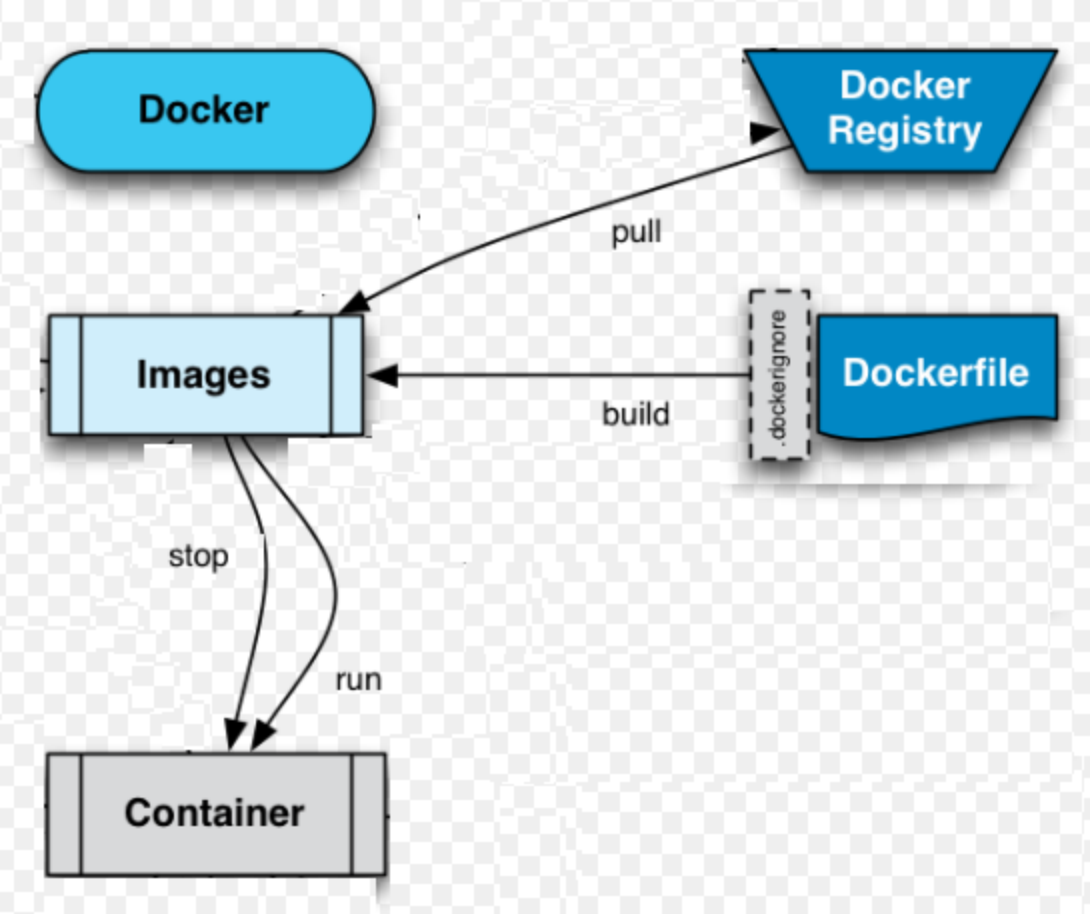


ObjectRiver
METADATA COMPILERS

Docker

One Docker Slide ☺

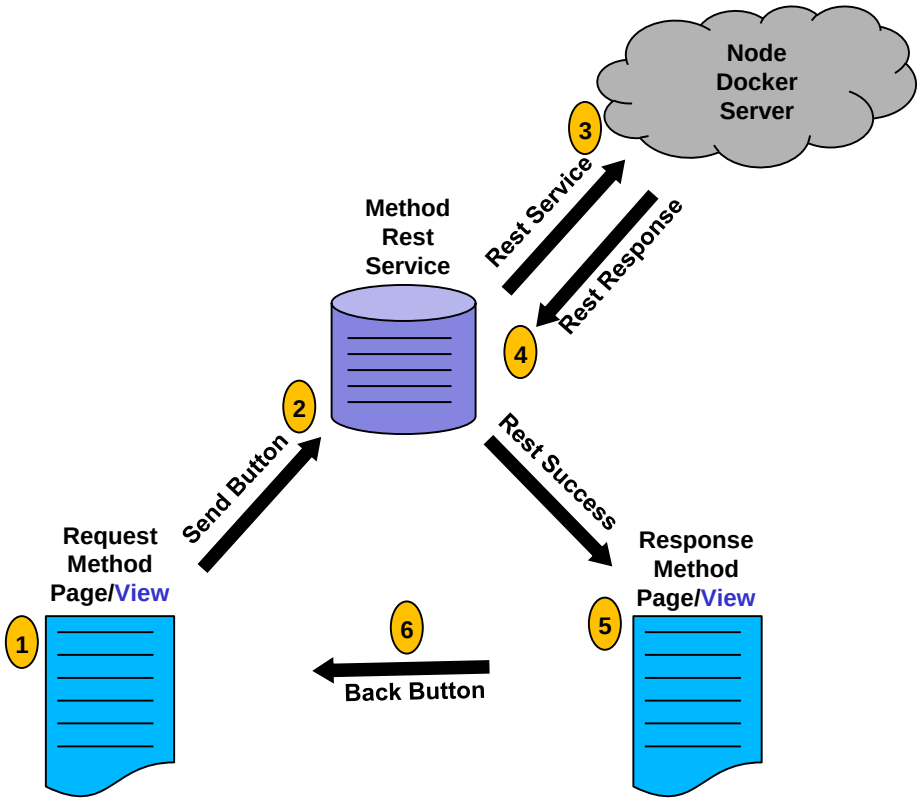
- Build Image w/ Dockerfile
  - Pulls Node/Carbon image
  - Adds generates node code
  - Creates new application image
- Run
  - Runs application image in docker demon
  - Application becomes running contain
- Stop
  - Stops application container



Architecture

Service is uncoupled from pages/application!

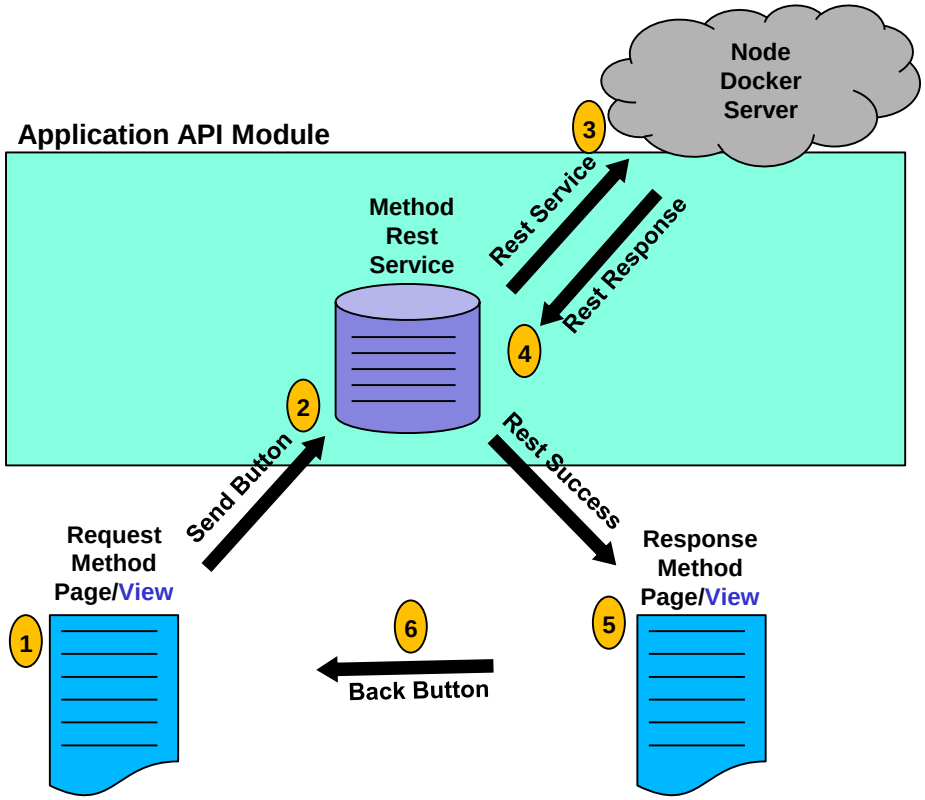
1. Request page input
2. Sent input to Rest Service
    1. Store inputs into service
3. Execute Rest request
4. Get Rest response
5. Display Response page
    1. Retrieve response from service
6. Back → Display Request page
    1. Retrieve previous inputs from service
7. Service is uncoupled from pages
    1. **Can be re-parented to any application!**
    2. **API can be published and shared**



Next Architecture

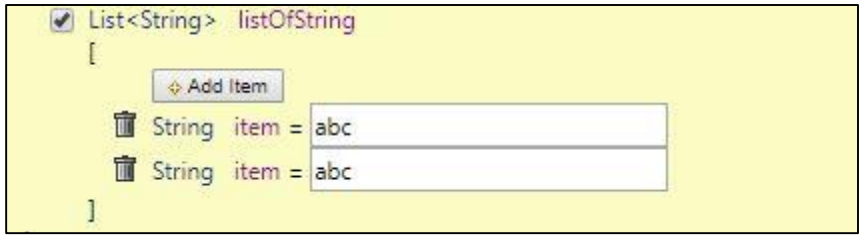
Services need to be bundled in a module!

- Uncoupled services can be re-parented into other applications.
- Service layer needs to be in a module to download for other Angular applications.



List<String>

Not in demo




```
addListItem(ls, value) {
  console.log('Add ' + value + 'to list');
  ls.push(value);
}
deleteListItem(ls, ii) {
  console.log('delete ' + ii + 'item from list');
  ls.splice(ii, 1);
}
trackByListOfString(index: any, item: any) {
  return index;
}
```

List<TestBean>

Add Item `(click)="addBeanItem(argArray,'TestBean')"`

Browser

HTML

```
<button type="button" title="Add an item" value="+"
        (click)="addBeanItem(argArray,'TestBean')" style="...">
    <img align="absmiddle" src="assets/icons/plus-small.gif"
         alt="+"> Add Item
</button>
```

Typescript

```
export class BeanMethodrequestComponent implements OnInit {
  classMap: Map<string,Function> = new Map();
  constructor(public beanMethodService: BeanMethodService) {
    this.classMap['TestBean'] = TestBean;
  }

  deleteListItem(ls, ii) {
    console.log('delete ' + ii + 'item from list');
    ls.splice(ii, deleteCount: 1);
  }
  addBeanItem(ls, classReference) {
    console.log(typeof this.classMap[classReference]);
    ls.push(new this.classMap[classReference]());
    console.log(`addBeanItem(classref=${classReference})`);
  }
  trackByBeanlist(index: any, item: any) {
    return index;
  }
  trackByMember1(index: any, item: any) {
    return index;
  }
```

ObjectRiver METADATA COMPILERS

Request Typescript

Arguments & Invoke

```
@Component({
  selector: 'app-beanmethodrequest',
  templateUrl: './beanmethodrequest.component.html',
  styleUrls: ['../shared/css/shared.component.css'],
  providers: []
})
export class BeanMethodrequestComponent implements OnInit {
  classMap: Map<string,Function> = new Map();
  constructor(public beanMethodService: BeanMethodService) {
    this.classMap['TestBean'] = TestBean;
  }

  // Request
  arg: TestBean = new TestBean();
  argArray: TestBean[] = new Array<TestBean>();
  option: MyMessageType = undefined;

          //.......................

  invoke() {
    const headers = new HttpHeaders( headers: {});
    this.beanMethodService.beanMethod( this.arg, this.argArray, this.option, headers, responsePage: 'beanmethodresponse');
  }
```

Response Typescript

Result

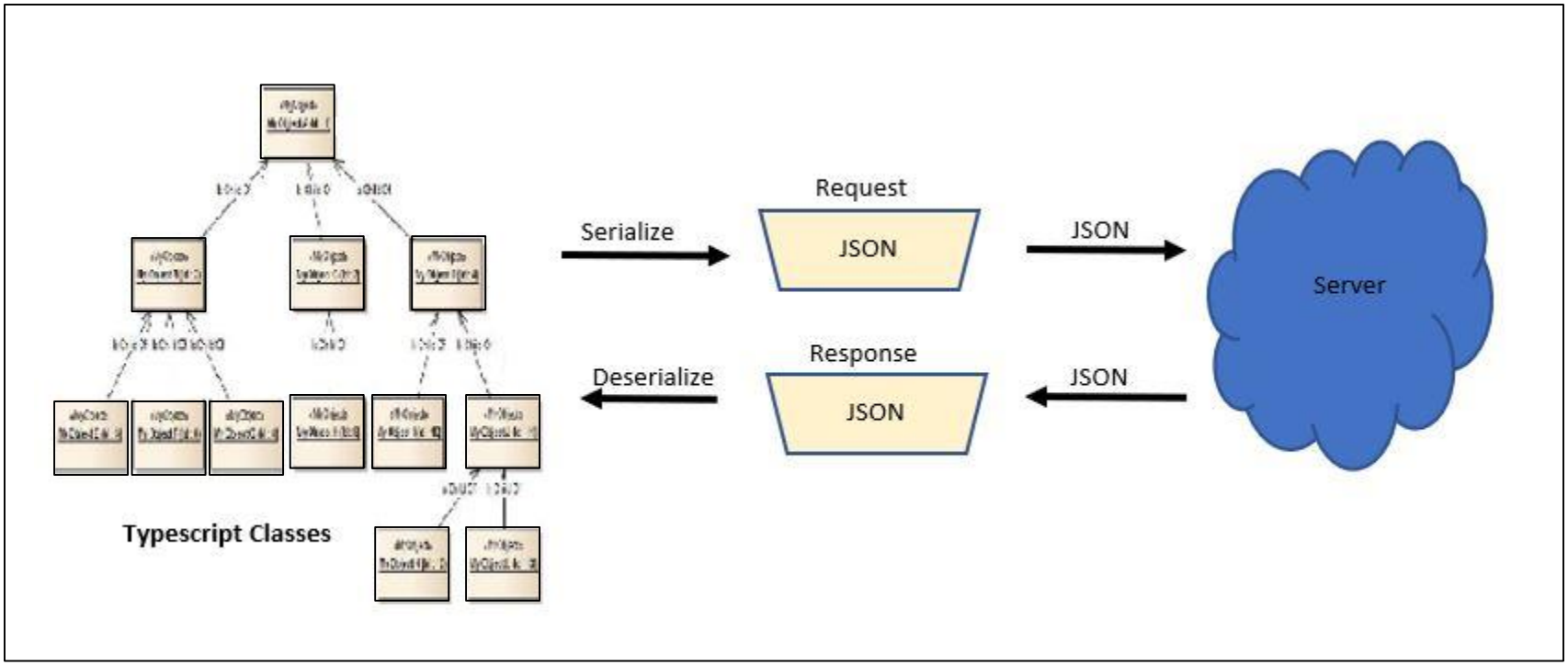
```
@Component({
  selector: 'app-beanmethodresponse',
  templateUrl: './beanmethodresponse.component.html',
  styleUrls: ['../../shared/css/shared.component.css'],
  providers: []
})
export class BeanMethodresponseComponent implements OnInit {
  // Response
  result: boolean;
  constructor(public beanMethodService: BeanMethodService) {
      this.result = this.beanMethodService.result__;
  }

  ngOnInit() {
    this.result = this.beanMethodService.result__;
  }
}
```

ObjectRiver
METADATA COMPILERS

Big Problem

Serialize/Deserialize just like Java ☺



ObjectRiver
METADATA COMPILERS

Json2Typescript

Secret Sauce

Readme | 0 Dependencies | 39 Dependents | 11 Versions

json2typescript

In Angular 2 applications, everyone consumes JSON API's from an external source. Type checking and object mapping is only possible in TypeScript, but not in the JavaScript runtime. As the API may change at any point, it is important for larger projects to verify the consumed data.

json2typescript is a small package containing a helper class that maps JSON objects to an instance of a TypeScript class. After compiling to JavaScript, the result will still be an instance of this class. One big advantage of this approach is, that you can also use methods of this class.

With **json2typescript**, only a simple function call is necessary, as demonstrated in this TypeScript snippet:

install

```
> npm i json2typescript
```

weekly downloads

3,789

version	license
1.0.6	MIT

open issues	pull requests
12	3

homepage | repository

Contributors

This NPM package was originally created by **Andreas Aeschlimann**, scientific researcher of the **Digital Humanities Lab** (DHlab) of **University of Basel** in 2016.

Json2Typescript

TestBean Class Definition

```
import {JsonObject, JsonProperty} from 'json2typescript';

@JsonObject('TestBean')
export class TestBean {
    @JsonProperty('member1', [String], true) public member1: String[] = undefined;
    @JsonProperty('member2', Number, true) public member2: Number = undefined;

    constructor() {
        this.member1 = new Array<string>();
    }
}
```

- Typescript Annotations!
  - @JsonObject
  - @JsonProperty

ObjectRiver METADATA COMPILERS

Json2Typescript

MyMessageType Enumeration Definition

- @JsonConverter
  - Custom converter support
  - Enumeration & Enum Arrays Conversion

```
import {JsonConverter, JsonCustomConvert} from 'json2typescript';

export const enum MyMessageType {
  Empty = 'Empty',
  Full = 'Full'
}

@JsonConverter
export class MyMessageTypeConverter implements JsonCustomConvert<MyMessageType> {
  serialize(value: any): any {
    if(value instanceof Array === true) {
        let MyMessageTypeList: Array<string> = new Array<string>();
        for (let i = 0; i < value.length; i++) {
            MyMessageTypeList.push(this.serialize(value[i]));
        }
        return MyMessageTypeList;
    }
    if (typeof value === 'string') { return value; }
    if (value === MyMessageType.Empty ) { return 'Empty'; }
    if (value === MyMessageType.Full ) { return 'Full'; }
    return value;
  }
  deserialize(value: any): any {
    if(value instanceof Array === true) {
        let MyMessageTypeList: Array<MyMessageType> = new Array<MyMessageType>();
        for (let i = 0; i < value.length; i++) {
            MyMessageTypeList.push(this.deserialize(value[i]));
        }
        return MyMessageTypeList;
    }
    if (value === 'Empty') { return MyMessageType.Empty; }
    if (value === 'Full') { return MyMessageType.Full; }
    return undefined;
```

beanMethod Request

beanmethodservice.ts

```
beanMethod( arg__: TestBean, argArray__: TestBean[], option__: MyMessageType, headers: HttpHeaders, responsePage: string): void {
    var params: HttpParams = new HttpParams();
    const jsonConvert: JsonConvert = new JsonConvert();

    /** Save input */
    this.arg__ = arg__;
    this.argArray__ = argArray__.slice(0);
    this.option__ = option__;

    /** Build body */
    class inDoc {
        @JsonProperty('arg', TestBean, true)  public arg: TestBean = undefined;
        @JsonProperty('argArray', [TestBean], true)  public argArray: TestBean[] = undefined;
        @JsonProperty('option', MyMessageTypeConverter, true) public option: MyMessageType = undefined;
        constructor( arg: TestBean, argArray: TestBean[], option: MyMessageType) {
            this.arg = arg;
            this.argArray = argArray;
            this.option = option;
        };
    };
    const body = jsonConvert.serializeObject(new inDoc( this.arg__, this.argArray__, this.option__ ));

    /** Call web service */
    this.http.post( url: baseUrl+this.url, body, options: {headers: headers, observe: 'response', params: params, responseType:'json'})
        .subscribe( next: (resp: any) => {
            switch ( resp.status ) {
                case 200: {
                    this.response = resp.body;
                    console.log(resp.body);
                    this.result__ = resp.body.result;
                    console.log('beanmethod.service result=' + this.result__);
                    this.router.navigateByUrl(responsePage);
                    break;
                }
```

beanJustReturn Response

beanjustreturn.ts

```
this.http.post( url: baseUrl + this.url,  body: null,  options: {headers: headers, observe: 'response', params: params, responseType: 'json'})
    .subscribe( next: (resp: any) => {
        switch ( resp.status ) {
            case 200: {
                this.response = resp.body;
                this.result__ = jsonConvert.deserializeObject(resp.body, TestBean );
                this.router.navigateByUrl(responsePage);
                break;
            }
            case 409: {
                console.log(`Application Exception: ${resp.statusCode} message="${resp.message}"`);
                this.pageMessage.setMessage(resp.status, resp.message, resp.trace);
                break;
            }
            default: {
                console.log(`Unknown Exception: ${resp.status} message="${resp.message}"`);
                this.pageMessage.setMessage(resp.status, resp.message,  trace: null);
            }
        }
    },
```

- This is the response from the beanJustReturn web service
- Shows jsonConvert.deserializeObject() called directly.

ObjectRiver
METADATA COMPILERS

That’s’ all folks

- Steven Lemmo
  - @ObjectRiver twitter (Follow me and re-tweet ☺)
  - steve@objectriver.net email support
  - https://www.linkedin.com/in/stevenlemmo/ (Follow me and share)
  - steveoriver Instagram (Off-Roading Jeep Pictures)
  - 508 655-6366 (Call or Text)