

PULSE ON



EIGEN LEERRITME



EEN TIENMINUTENGES PREK IS NOOIT MEER HETZELFDE

KOMT DIT ZIEN REIS= doel, middelen & tijdsduur



Lessons learned

from a large scale java web app

Jago de Vreede

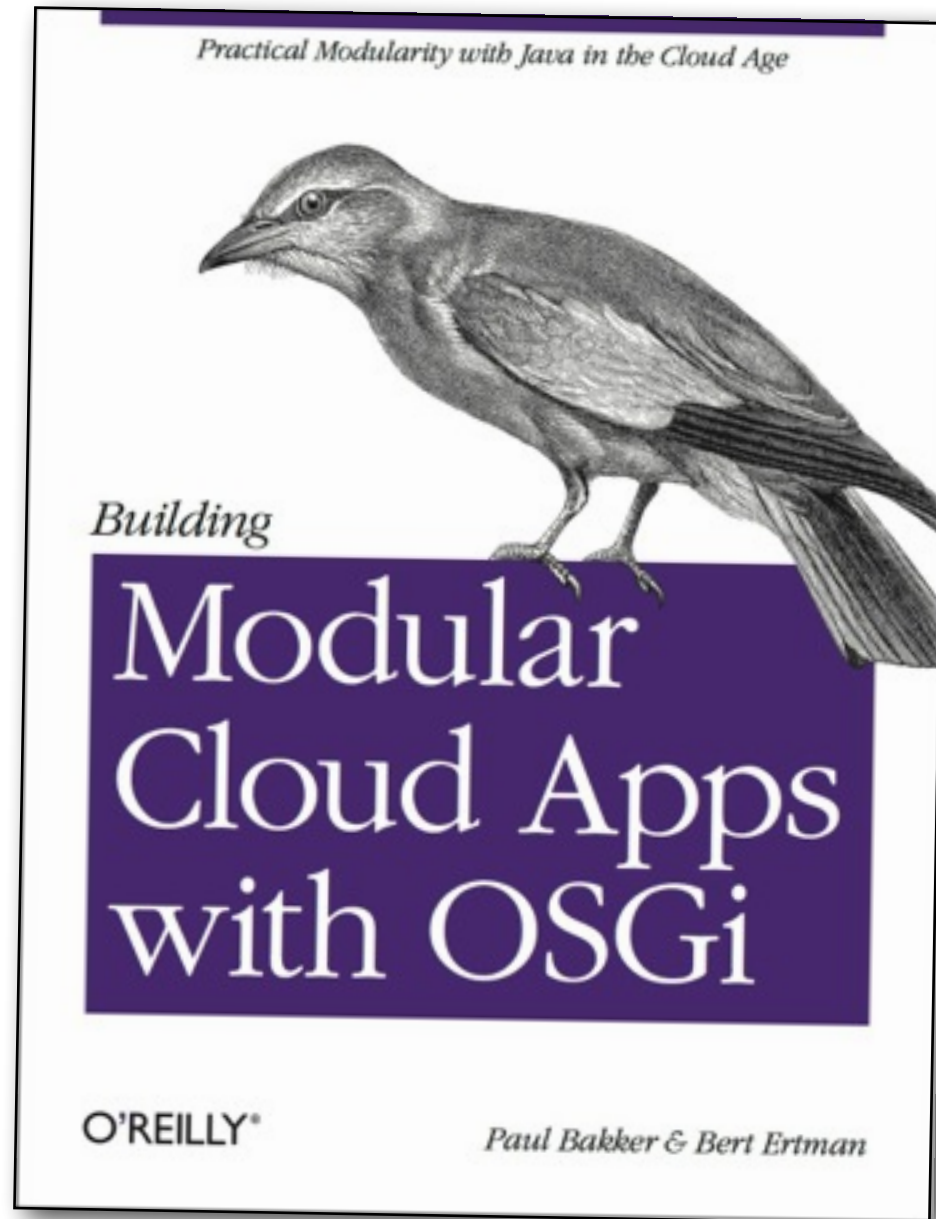
Paul Bakker

Luminis

Conversing worlds

Utebouwe





Paul Bakker



@pbakker



Jago de Vreede



Agenda

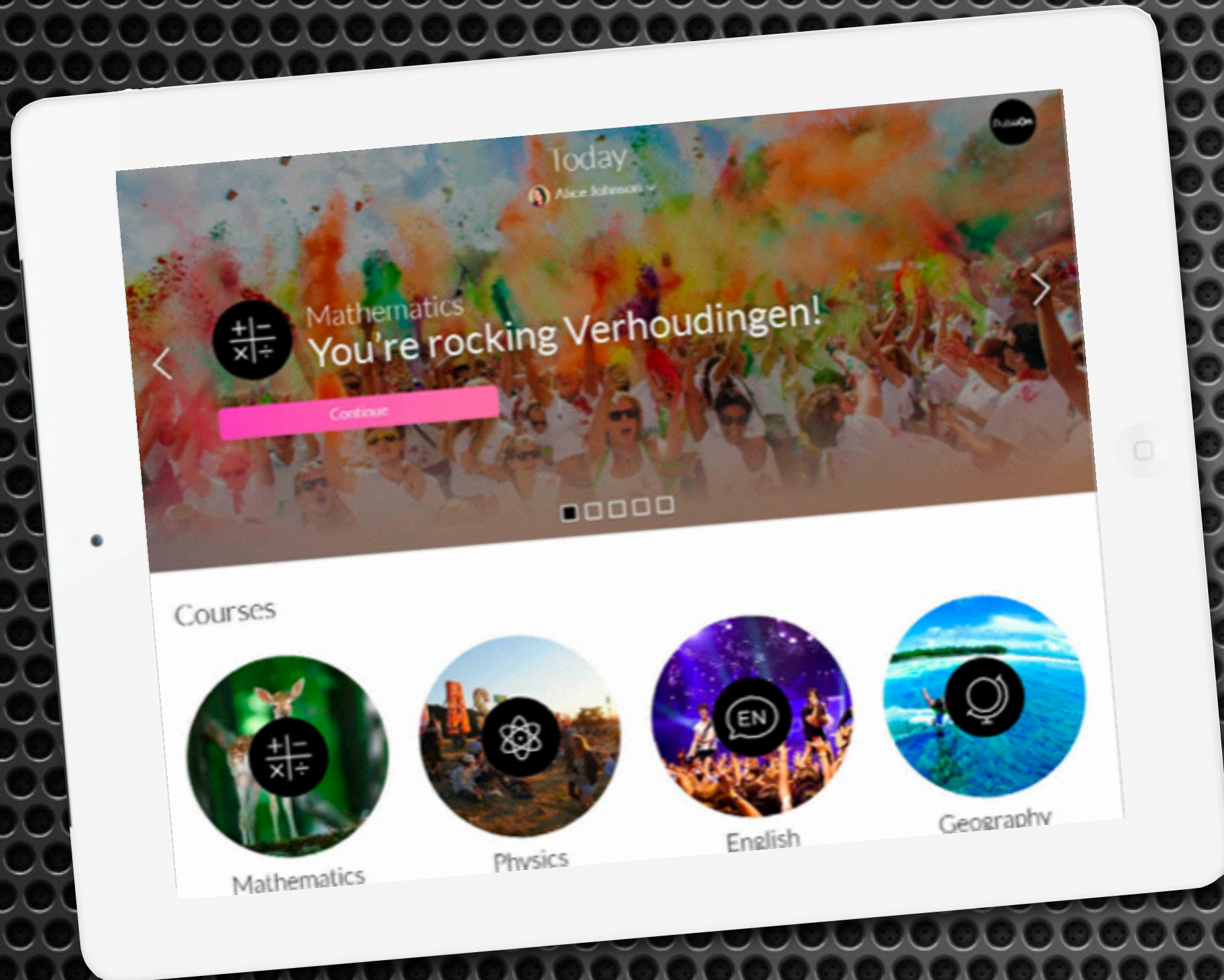
- High level architecture
- Modular architecture
- Modularity with OSGi
- Cloud deployments
- Frontend frameworks

ANY
QUESTIONS

?

Ask whenever
you want!

The case



PulseOn

- Educational system focussed on personalized learning
- used in high schools in the Netherlands
- Expand to other countries in the near future

Tools

bndtools

Eclipse OSGi plugin

<http://bndtools.org/>



Eclipse

<http://eclipse.org/>



Source control



cloud provisioning

<http://ace.apache.org/>



WebStorm

<http://jetbrains.com/webstorm/>

[webstorm/](http://jetbrains.com/webstorm/)



Stash, Jira, Bamboo

<http://atlassian.com/>



Requirements:

- Agile and modular
- Modern web app
- UI mostly offloaded to clients or devices
- Integration via REST API
- Horizontally scalable

High level architecture

A
m
d
a
t
u

HTML 5 + JavaScript

RESTful services

OSGi services

Apache Felix

Mongo

S3

Components

- JPA
- Auth
- Blob stores
- MONGODB
- Multi-tenancy
- Search
- Remote Services
- REST
- Template
- Web
- ...



amdatu

Deployment

Load Balancer

PulseOn

PulseOn

PulseOn

School A

Mongo

Load Balancer

PulseOn

PulseOn

PulseOn

School B

Mongo

Load Balancer

DAMS

DAMS

DAMS

Content backend

Mongo



Profiles Rest

Profiles API

Profiles Service

MongoDB

Progress Rest

Progress API

Progress Service

MongoDB

services

all the way down

Curriculum API

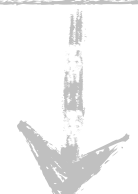
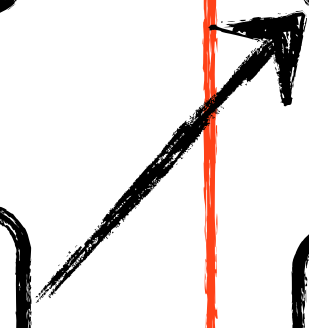
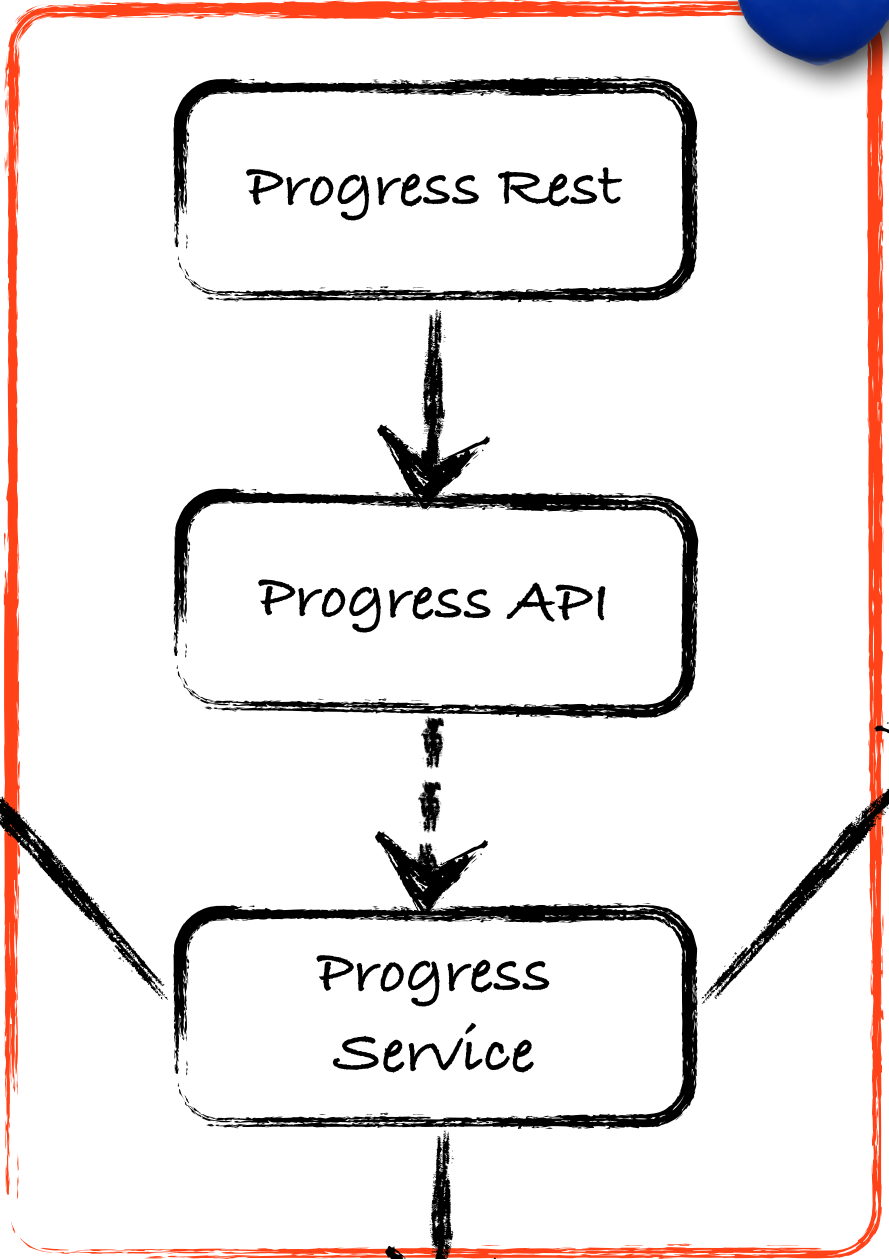
Curriculum Service


MongoDB

... Rest

... API

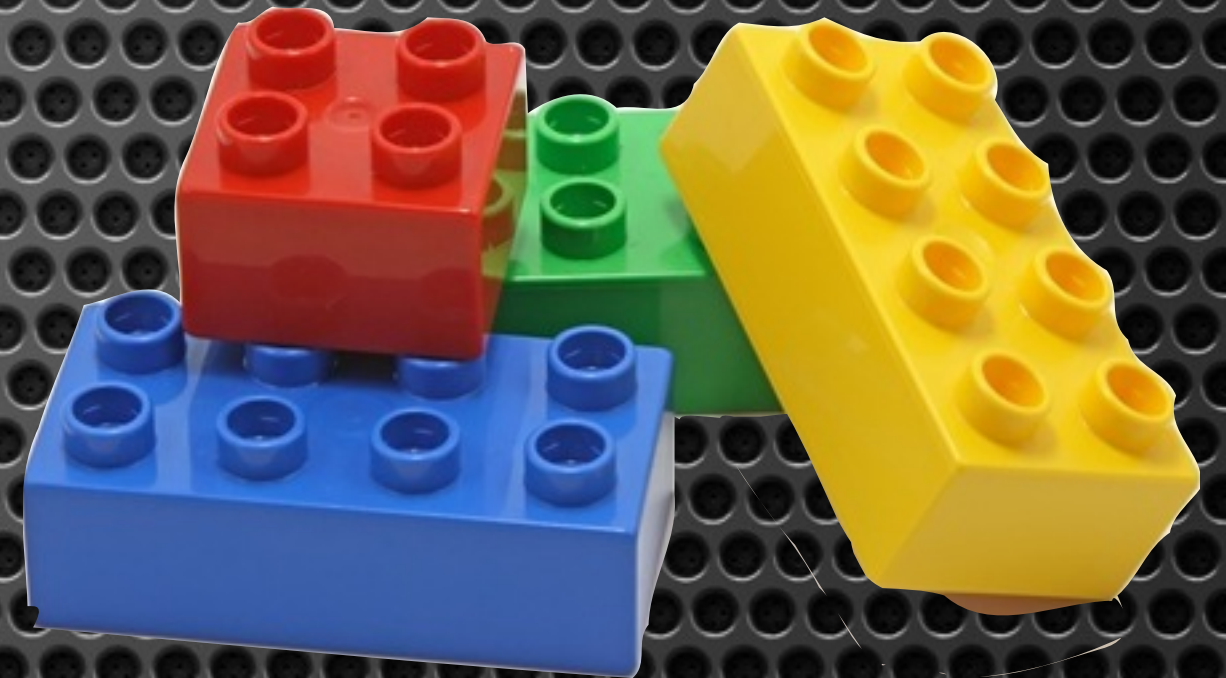
... Service






A service should only
do **One** single thing

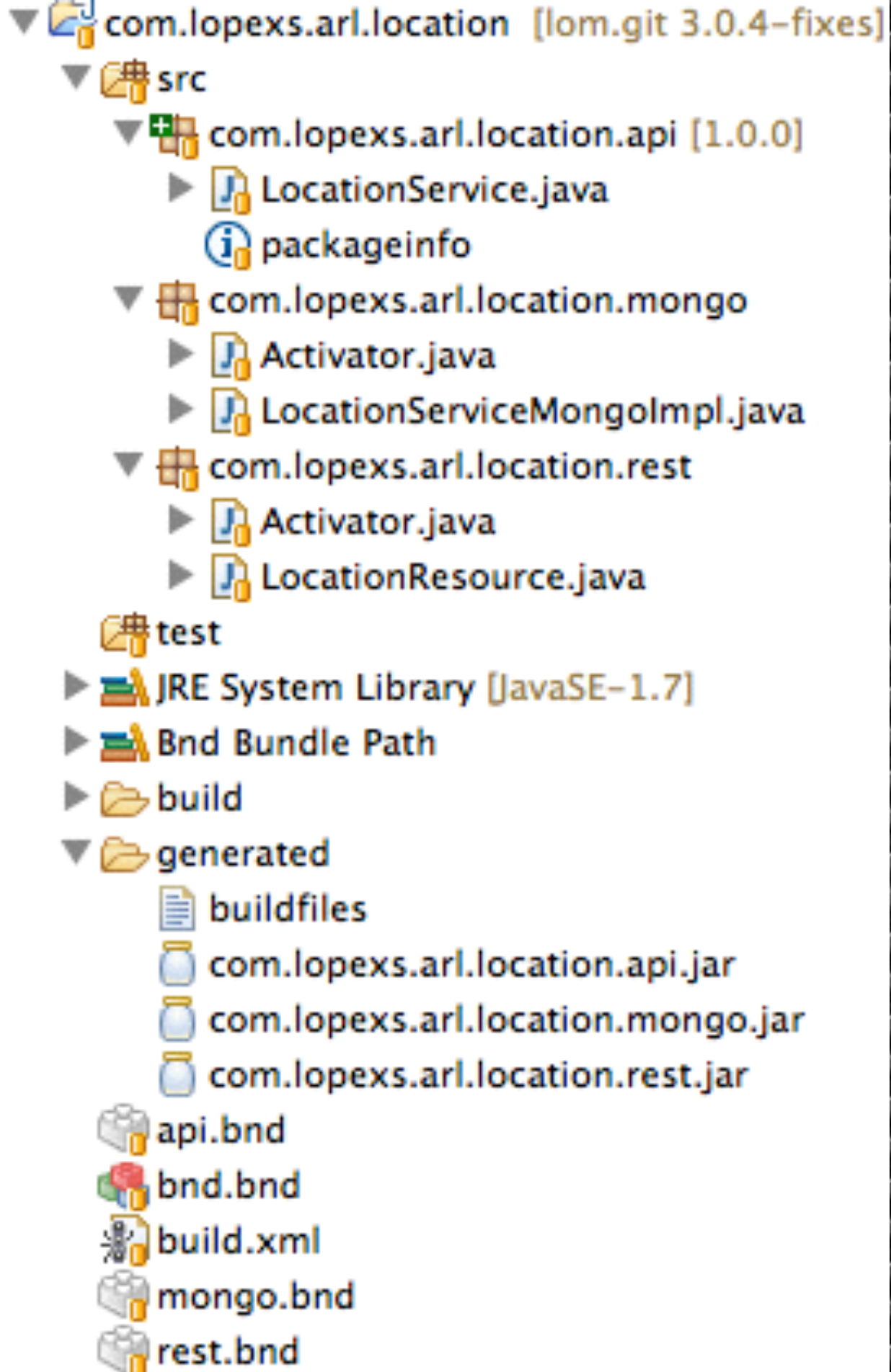
Services are the
(reusable) **building**
blocks of our
application





Benefits of a services based architecture

- Small services are easy to maintain
- Small services are composable
- Services are easily testable



Packaging services

- Services are packaged in small bundles
- Related bundles may be generated from a single Bndtools project

OSGi Services

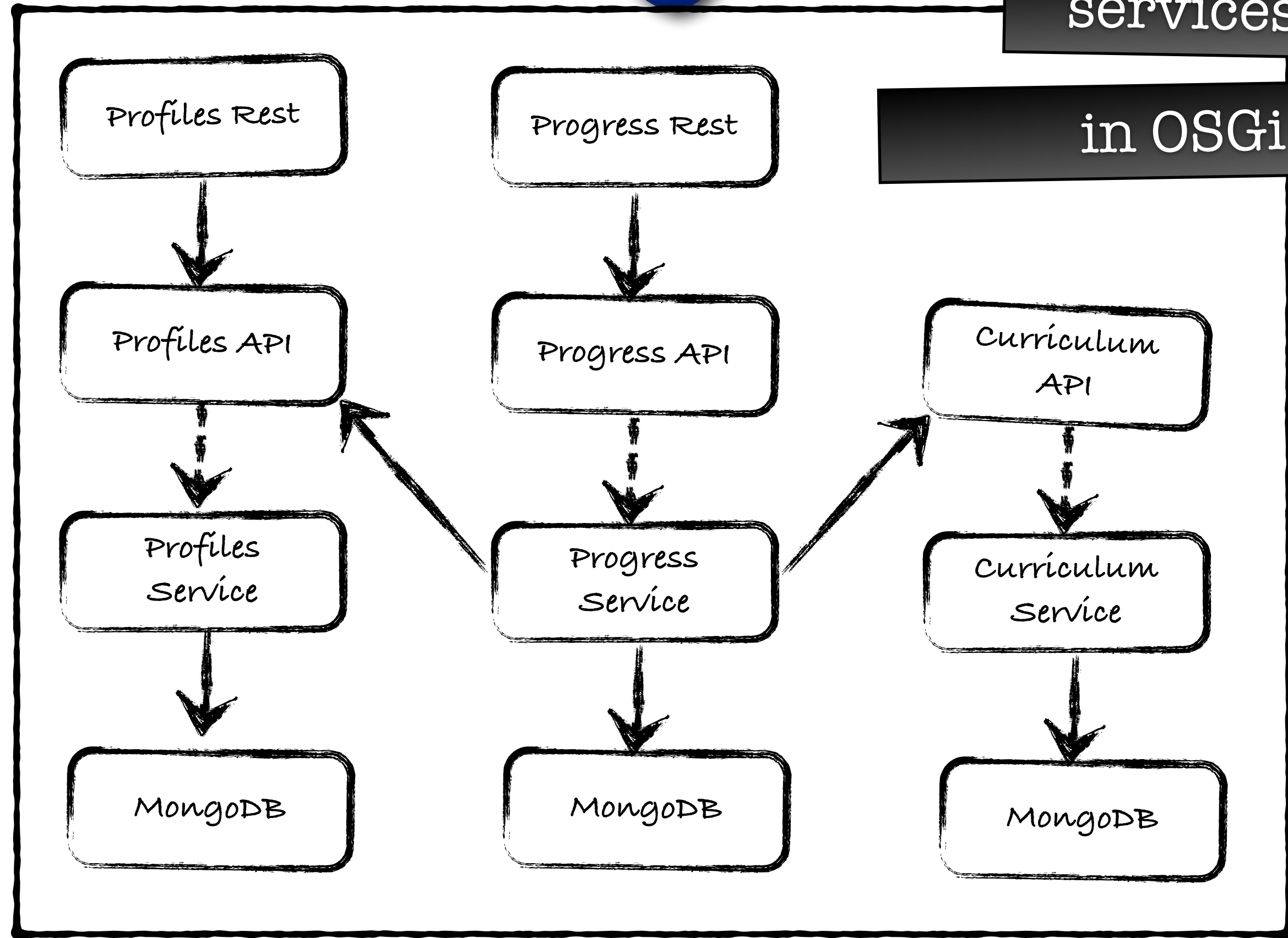
VS

Micro Services

<http://paulonjava.blogspot.nl/2014/04/micro-services-vs-osgi-services.html>

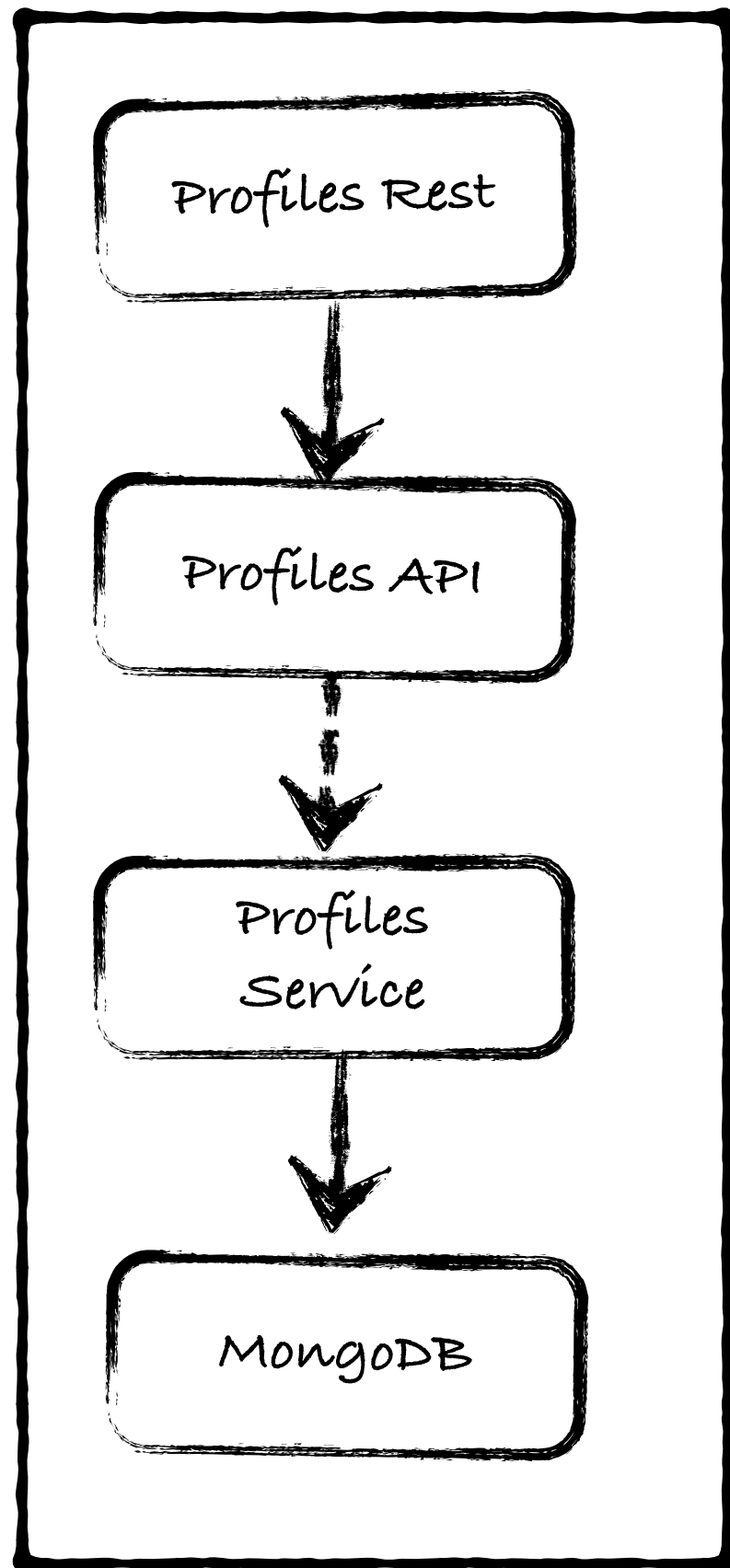
services

in OSGi

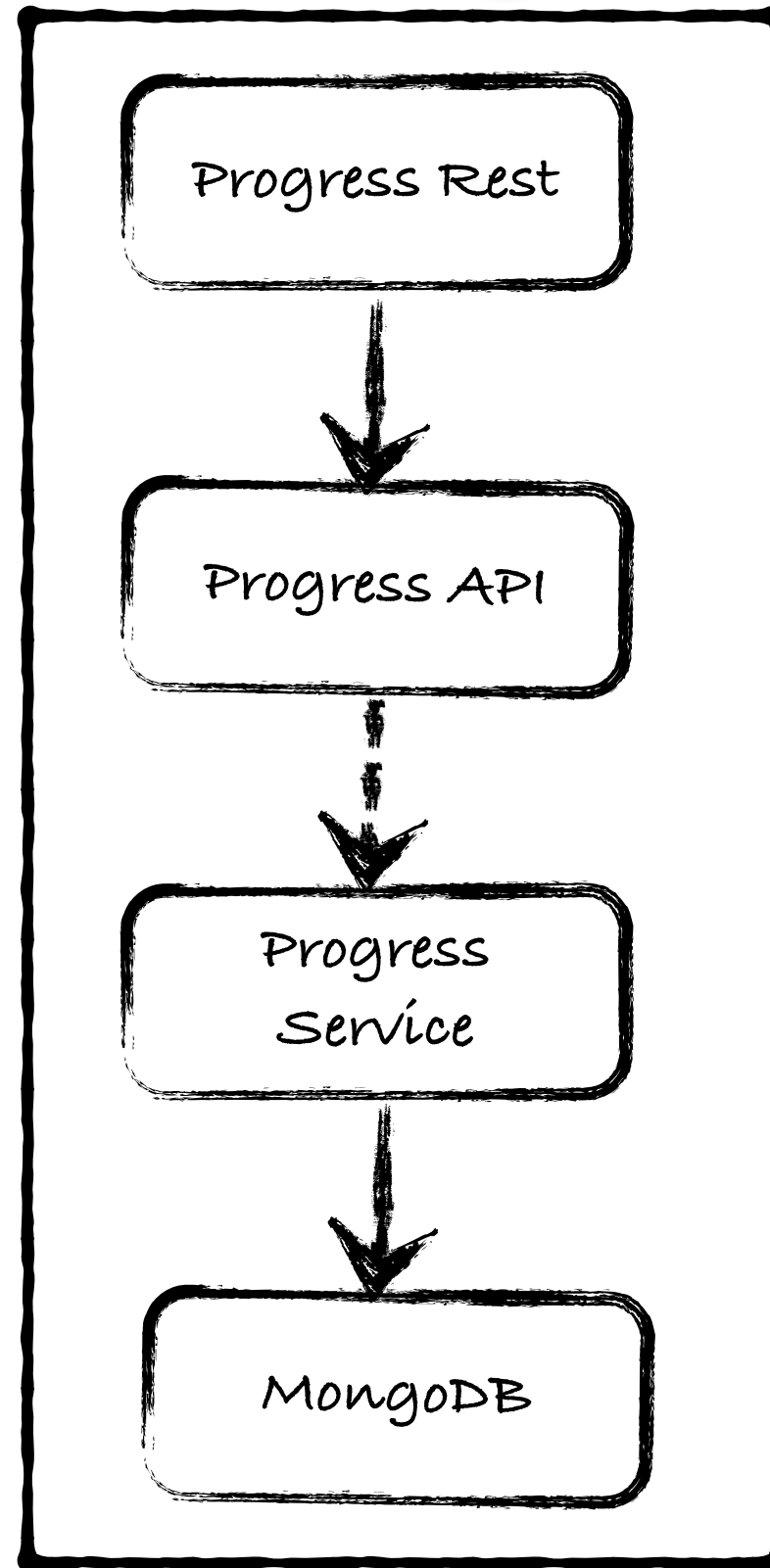


Micro

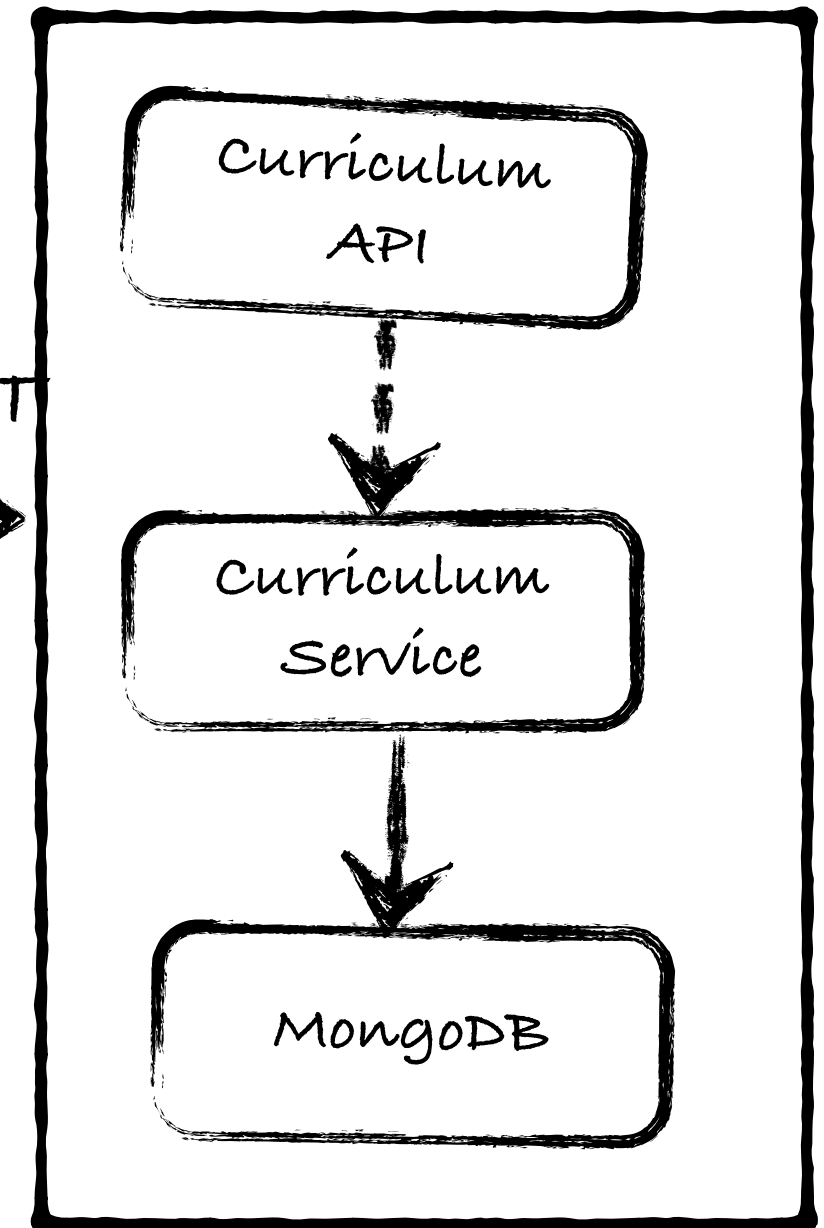
Services



REST
←



REST
→



Benefits of OSGi services

- ❑ Modular architecture!
- ❑ No remoting overhead
- ❑ Much simpler DevOps

Benefits of Micro Services

- ❑ Modular architecture!
- ❑ Teams can own a service

Data Storage



Semantic data store
(triple store)



MySQL
(with JPA)



MONGODB




Beginning

Now

Relational databases

in a modular system



A part of data should be
owned by 1 service

How to set boundaries in a relational world!?

Relational databases

in the modern web

- ORM is REALLY difficult to get right
- Relational databases don't scale well

MongoDB: our database of choice

- ❑ Object Mapping trivial
- ❑ Easy to scale
- ❑ Powerful aggregation/map-reduce framework

Code example: MongoDB with Amdatu

Amdatu
Mongo Service

Setup Object
Mapper

Execute query

```
@Component
public class MongoProducts implements ProductService{

    @ServiceDependency
    private volatile MongoDBService mongoDbService;
    private volatile DBCollection collection;
    private volatile JacksonDBCollection<Product, String> products;

    @Start
    public void start() {
        collection = mongoDbService.getDB().getCollection("products");
        products = JacksonDBCollection.wrap(collection, Product.class, String.class);
    }

    @Override
    public List<Product> listProducts(String category) {

        DBCursor<Product> dbCursor = products.find().is("category", category);
        List<Product> result = new ArrayList<>();

        dbCursor.forEach(result::add);

        return result;
    }
}
```


Release & Deployment



F 4
B 2
X G
U 1

51C
181

141070

517

AV
Flex
8%

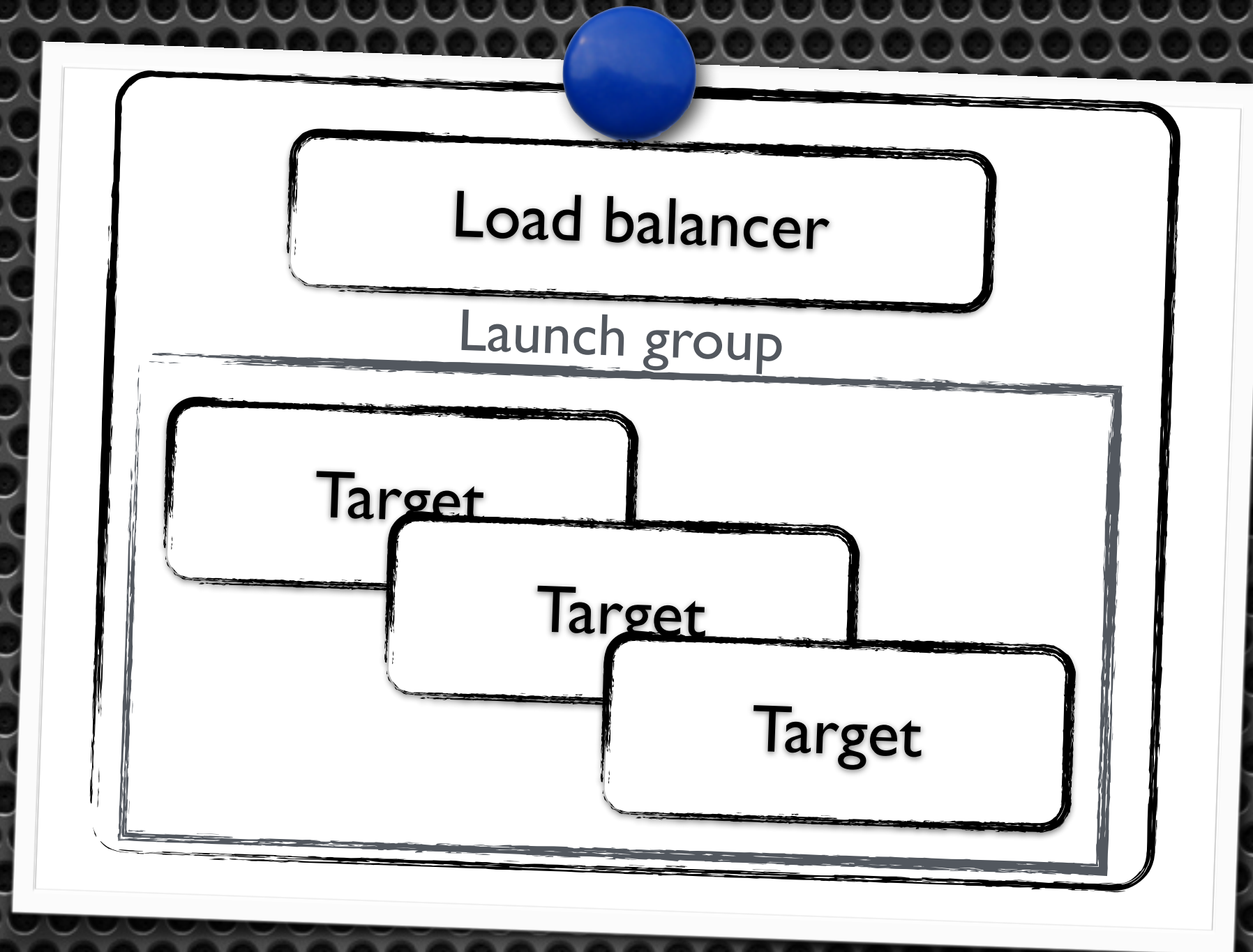
1173

FBXU 514107 0
4261

MAX. GR.	1480 KG
	3280 LB
TARE	3750 KG
	8270 LB
NET	26730 KG
	58930 LB
U. CAP.	15000 CBFT



Autoscaling



Binaries

 **Bamboo**

 **gradle**



Build

Binaries

Repository



Deployment



- ❑ *Software distribution framework*
- ❑ *Manages the installation and upgrade of bundles, configuration, etc. to heterogenous targets*

Deployment

Load Balancer

PulseOn

PulseOn

PulseOn

School A

Mongo

Load Balancer

PulseOn

PulseOn

PulseOn

School B

Mongo

Load Balancer

DAMS

DAMS

DAMS

Content backend

Mongo



Deployment

Load Balancer

PulseOn

PulseOn

PulseOn

School

Load Balancer

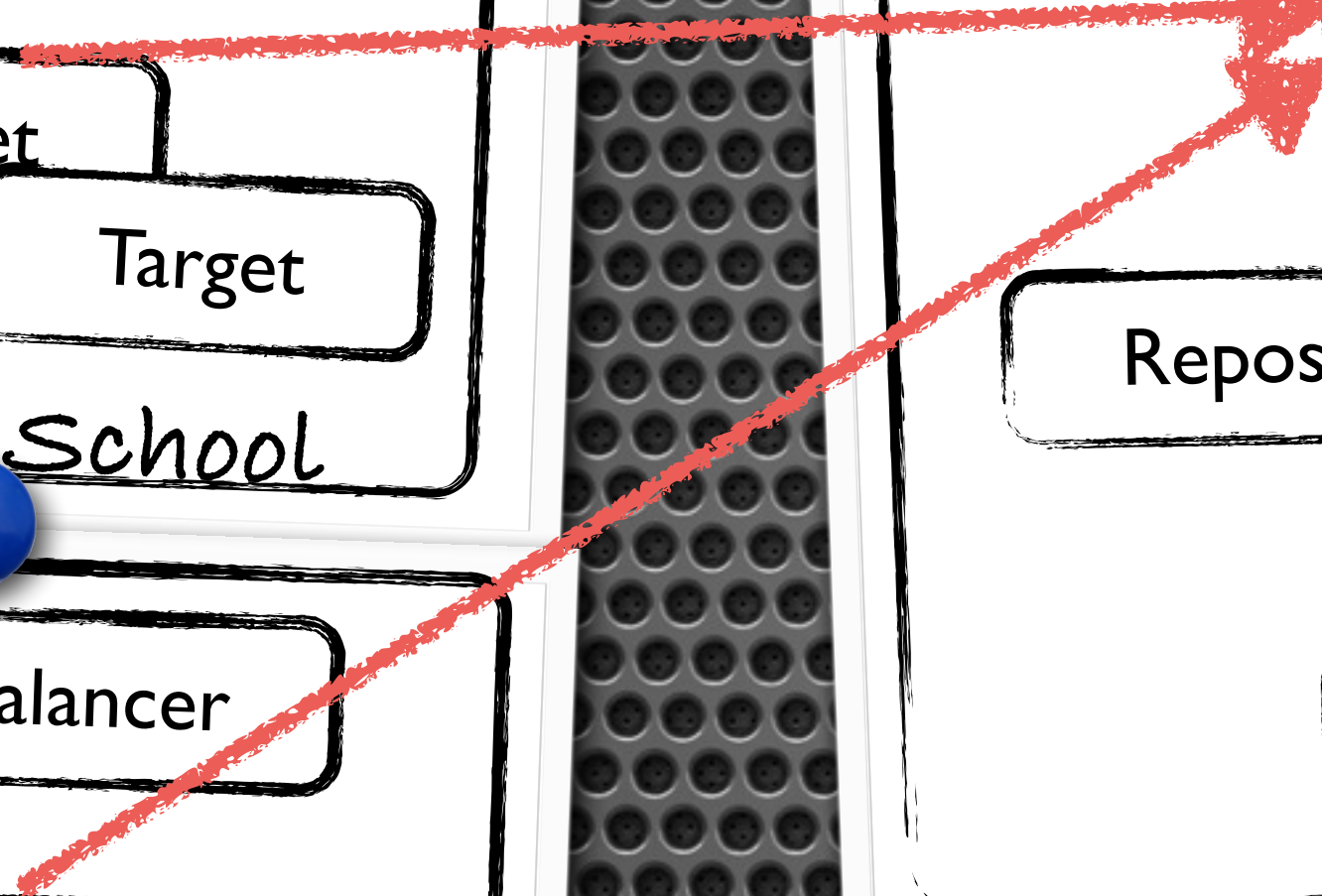
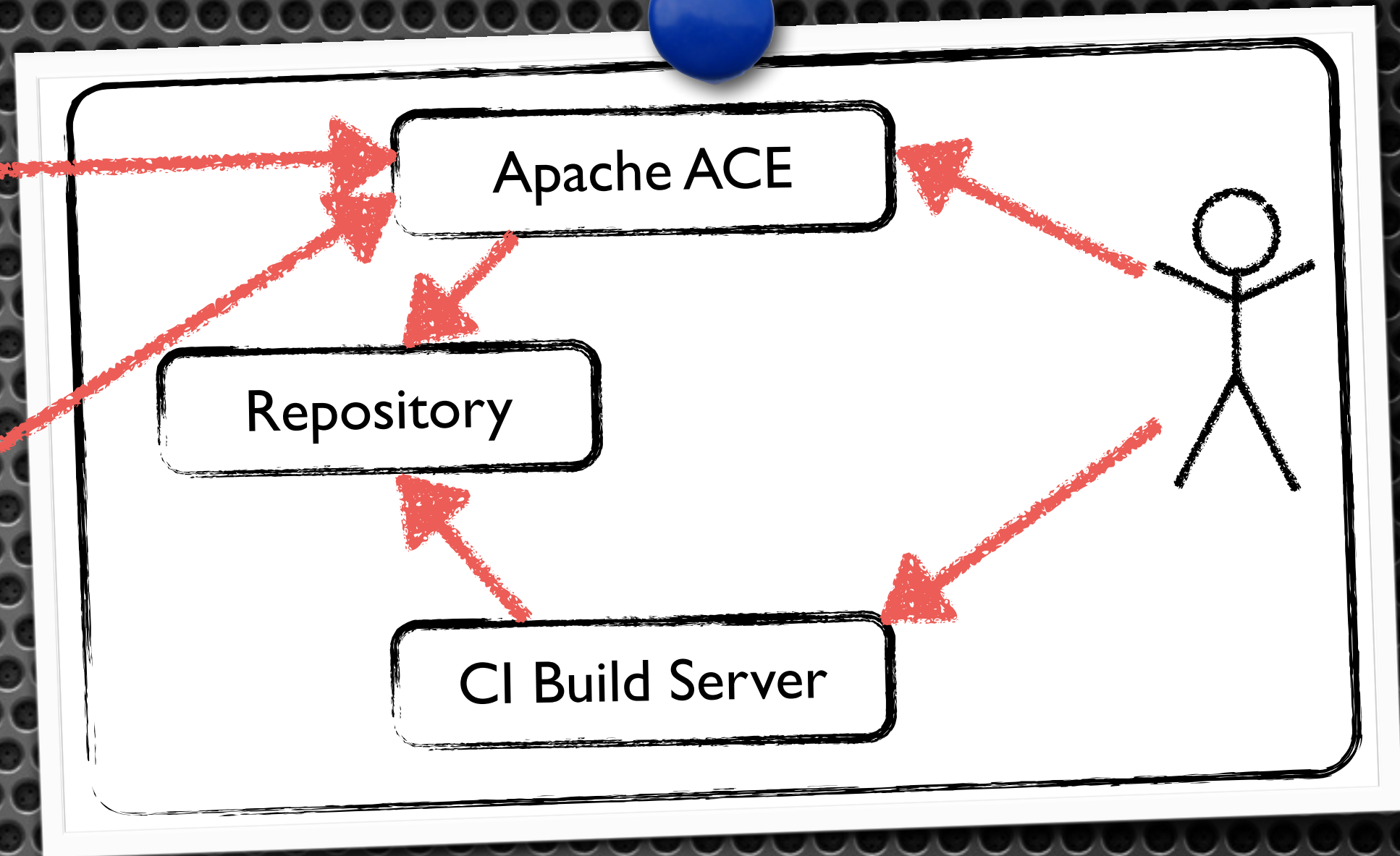
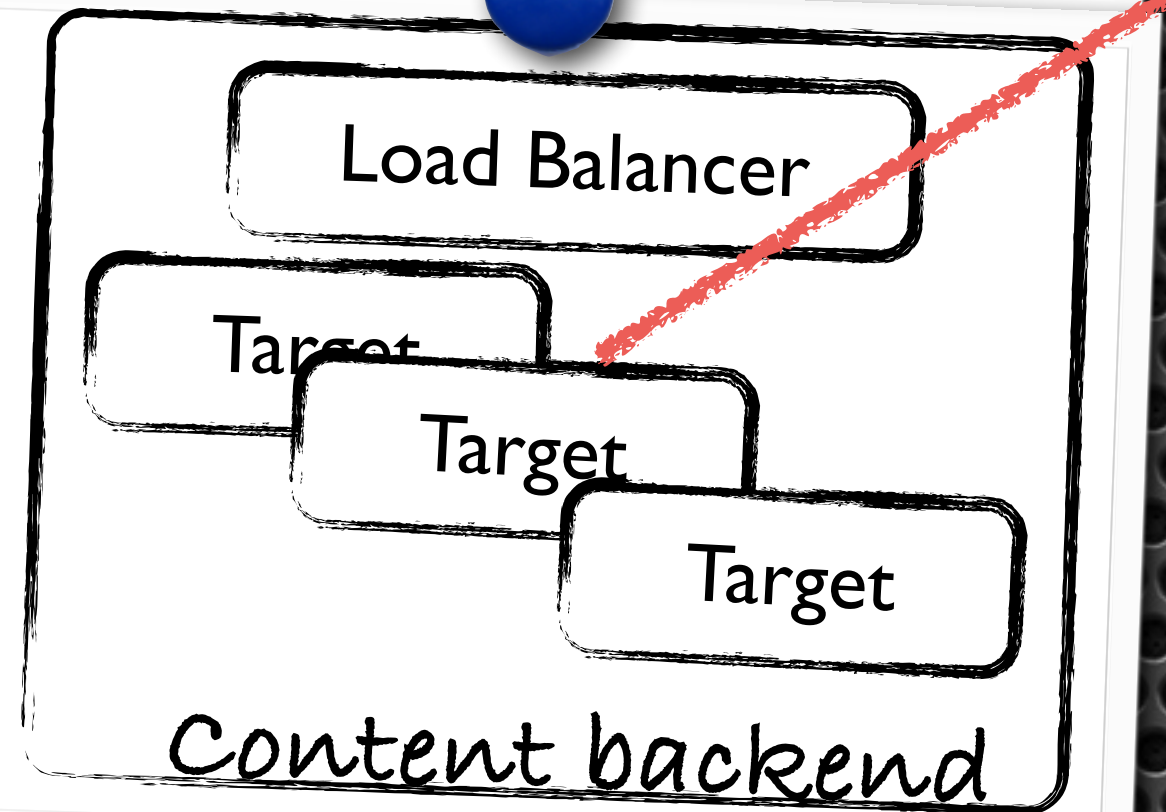
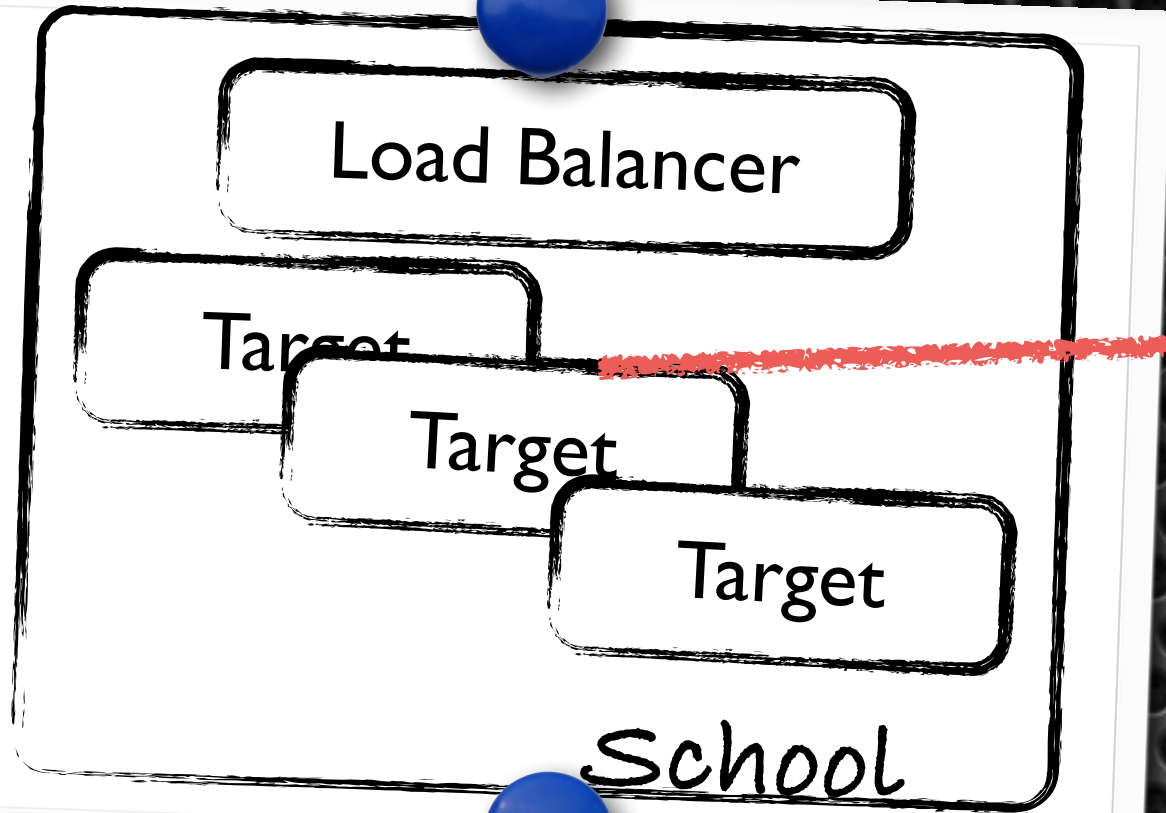
DAMS

DAMS

DAMS

Content backend

Deployment



Continuous Deployment

Load Balancer

Target

School

Load Balancer

Target

Content backend

Apache ACE

Snapshot Repo

CI Build Server

Like to know more?

Continuous Automated Deployment with Apache ACE
youtube.com/watch?v=4S_zvgG_MLW

A white rectangular sign is pinned to a dark grey perforated metal surface with two blue pushpins at the top corners. The sign features the text '*TESTING*' in large, bold, black, hand-drawn capital letters at the top. Below this, the words 'PLEASE', 'DO NOT', and 'DISTURB' are written in the same style, stacked vertically. On the left and right sides of the sign, there are vertical illustrations of school supplies: a globe at the top, a stack of papers, a pencil, and a blue folder or book. The entire sign is framed by a white border.

TESTING

**PLEASE
DO NOT
DISTURB**

Unit testing OSGi code

is trivial

But what if we want

more?

Integration testing turns

out to be trivial as well!

```
import org.amdatu.testing.configurator.OSGiTestConfigurator.
```

```
public class ProductServiceTest {
```

```
    private volatile ProductService m_productService;
```

```
    @Before
```

```
    public void setUp() throws Exception {
```

```
        configureTest(this,  
            configureFactory("org.amdatu.mongo").set("dbName", "amdatu-chat"),  
            inject(ProductService.class));
```

```
    }
```

```
    @Test
```

```
    public void listByTag() {
```

```
        List<Product> products = m_productService.listProductsByTag("OSGi");
```

```
        Assert.assertEquals(1, products.size());
```

```
    }
```

```
}
```

Configure and
inject service

JUnit style
asserts

Run tests from Bndtools

The screenshot shows the 'Run Bundles' dialog box in Bndtools. At the top, there are buttons for 'Run OSGi', 'Debug OSGi', and 'Export'. Below the title bar, the text reads 'The listed bundles will be added to the runtime.' A list of bundles is displayed, including:

- org.mongodb.mongo-java-driver
- org.amdatu.mongo
- org.apache.felix.configadmin
- org.jongo
- de.undercouch.bson4jackson
- org.apache.felix.dependencymanager
- org.apache.felix.dependencymanager.runtime
- org.apache.felix.dependencymanager.shell
- org.apache.felix.metatype
- org.apache.felix.eventadmin
- org.apache.felix.log
- org.amdatu.chat.products latest
- org.amdatu.testing.configurator
- com.fasterxml.jackson.core.jackson-annotations
- com.fasterxml.jackson.core.jackson-core
- com.fasterxml.jackson.core.jackson-databind
- org.amdatu.chat.products.itest

The screenshot shows the Bndtools IDE interface. The top toolbar includes 'Run OSGi', 'Debug OSGi', and 'Export'. The main workspace displays a project structure with the following items:

- org.amdatu.chat.products.itest [javaone2014 master]
 - src
 - org.amdatu.chat.products.itest
 - ProductServiceTest.java
- test
- JRE System Library [JavaSE-1.8]
- Bnd Bundle Path
- generated
- bnd.bnd

A context menu is open over the 'bnd.bnd' bundle, showing the following options:

- New
- Open (F3)
- Open With
- Show In (⌘W)
- Copy (⌘C)
- Copy Qualified Name
- Paste (⌘V)
- Delete (⌘X)
- Build Path
- Refactor (⌘T)
- Import...
- Export...
- Refresh (F5)
- Assign Working Sets...
- Release Bundle
- Debug As
- Run As
 - 1 Bnd OSGi Run Launcher
 - 2 Bnd OSGi Test Launcher (JUnit)
- Coverage As
- Team
- Compare With
- Replace With
- Properties (⌘I)

At the bottom, there is a 'Repositories' section and a search bar with the text 'Enter search...'. The bottom right corner contains icons for refresh, save, and other actions.

Frontend frameworks



```
32 <div class="row row-offpage">
33   <div class="carrouselTestWindow">
34     <div class="left" data-ng-class="{width:width, height:height}>
35     <div class="right" data-ng-class="{width:width, height:height}>
36
37     <div class="carrouselTestContainer" style="width:100%; height:100%;>
38       <div class="carrouselRow" data-ng-repeat="item in items">
39         <div class="container">
40           <div class="carrouselItem" data-ng-repeat="item in items">
41             <h2 data-ng-bind="item.title">
42             <h1 data-ng-bind="item.title">
43             <div class="text container" data-ng-bind="item.description">
44           </div>
45         </div>
46       </div>
47     </div>
48   </div>
49 </div>
50
51 <div class="container">
52   <div class="row">
53     <div class="carrouselTestWindow">
```


jQuery

Pros

Easy to get started

Can do a lot with not much code

Lot of helpful plugins available

```
1 <form>
2   <label>Enter a Todo and hit enter!</label>
3   <input type="text" />
4 </form>
5 <ul></ul>
```

```
1 $('form').submit(function () {
2   if ($('#input').val() !== '') {
3     var input_value = $('#input').val();
4     $('#ul').append('<li>' + input_value +
5                   '<a href="">x</a></li>');
6   };
7   $('#input').val('');
8   return false;
9 });
10
11 $(document).on('click', 'a', function (e) {
12   e.preventDefault();
13   $(this).parent().remove();
14 });
```


jQuery

Cons

```
1 <form>
2   <label>Enter a Todo and hit enter!</label>
3   <input type="text" />
4 </form>
5 <ul></ul>
```

```
1 $('form').submit(function () {
2   if ($('#input').val() !== '') {
3     var input_value = $('#input').val();
4     $('#ul').append('<li>' + input_value +
5                   '<a href="">x</a></li>');
6   };
7   $('#input').val('');
8   return false;
9 });
10
11 $(document).on('click', 'a', function (e) {
12   e.preventDefault();
13   $(this).parent().remove();
14 });
```

Hard to
maintain

Hard to test

AngularJS

```
1 <div class="container" ng-controller="TodoController">
2 <ul id="todos" class="unstyled">
3   <li ng-repeat="todo in todos">
4     <label class="checkbox">
5       <input type="checkbox" ng-model="todo.completed"
6         ng-change="changeCompleted(todo)" />{{todo.title}}
7     </label>
8   </li>
9 </ul>
10 <form class="form-inline">
11   <input id="todo-title" type="text" ng-model="newTodoTitle" />
12   <button id="add-btn" class="btn btn-success" ng-click="addTodo(newTodoTitle)">Add</button>
13 </form>
14 </div>
```


AngularJS

Pros

```
1 var todoApp = angular.module('todoApp', []);
2 todoApp.controller('TodoController', function($scope, $http) {
3   $scope.todos = [];
4
5   $http.get('/todos').success(function(todos) {
6     $scope.todos = todos;
7   });
8
9   $scope.addTodo = function(title) {
10    $scope.todos.push({ title : title });
11  };
12
13  $scope.removedTodo = function(todo) {
14    $scope.todos = $scope.todos.splice(
15      $scope.todos.indexOf(todo) ,1);
16  }
17 });
```

Well
structured

Testable

Designed
towards
consuming
REST data

No
guessing
in html

TypeScript

Pros

```
class ItemService {  
  private items : string[] = [];  
  public getItems() : string[] {  
    return this.items;  
  }  
}  
  
export = ItemService;
```

compile
time checks

code
completion

```
define(["require", "exports"], function(require, exports) {  
  var ItemService = (function () {  
    function ItemService() {  
      this.items = [];  
    }  
    ItemService.prototype.getItems = function () {  
      return this.items;  
    };  
    return ItemService;  
  })();  
  
  return ItemService;  
});
```


TypeScript

```
class ItemService {  
  private items : string[] = [];  
  public getItems() : string[] {  
    return this.items;  
  }  
}  
  
export = ItemService;
```



```
define(["require", "exports"], function(require, exports) {  
  var ItemService = (function () {  
    function ItemService() {  
      this.items = [];  
    }  
    ItemService.prototype.getItems = function () {  
      return this.items;  
    };  
    return ItemService;  
  })();  
  
  return ItemService;  
});
```

Cons

Need to
have a
definition
to work

Slow-ish

Wrap up

A modular architecture gives us:

- Maintainability
- Extensibility
- Freedom to change

But what if I want

Spring

EJB

...?



just don't... You really don't need to.

cloud provisioning

<http://ace.apache.org/>



cloud OSGi services

<http://www.amdatu.org/>



Paul Bakker

paul.bakker@luminis.eu



Eclipse OSGi plugin

<http://bndtools.org/>



That's us

<http://luminis-technologies.com>



Jago de Vreede

jago.devreede@luminis.eu