JAVA FAKTURA

S02: Stranger Spring

E03 : Spring MVC - REST in peace

Tomek Owczarek

12/11/2019

Tomek Owczarek

JAVA FAKTURA

Expert Software Engineer @ TOMTOM

Java Trainer @ software development academy

KODUJMY https://kodujmy.pl

https://github.com/towczare

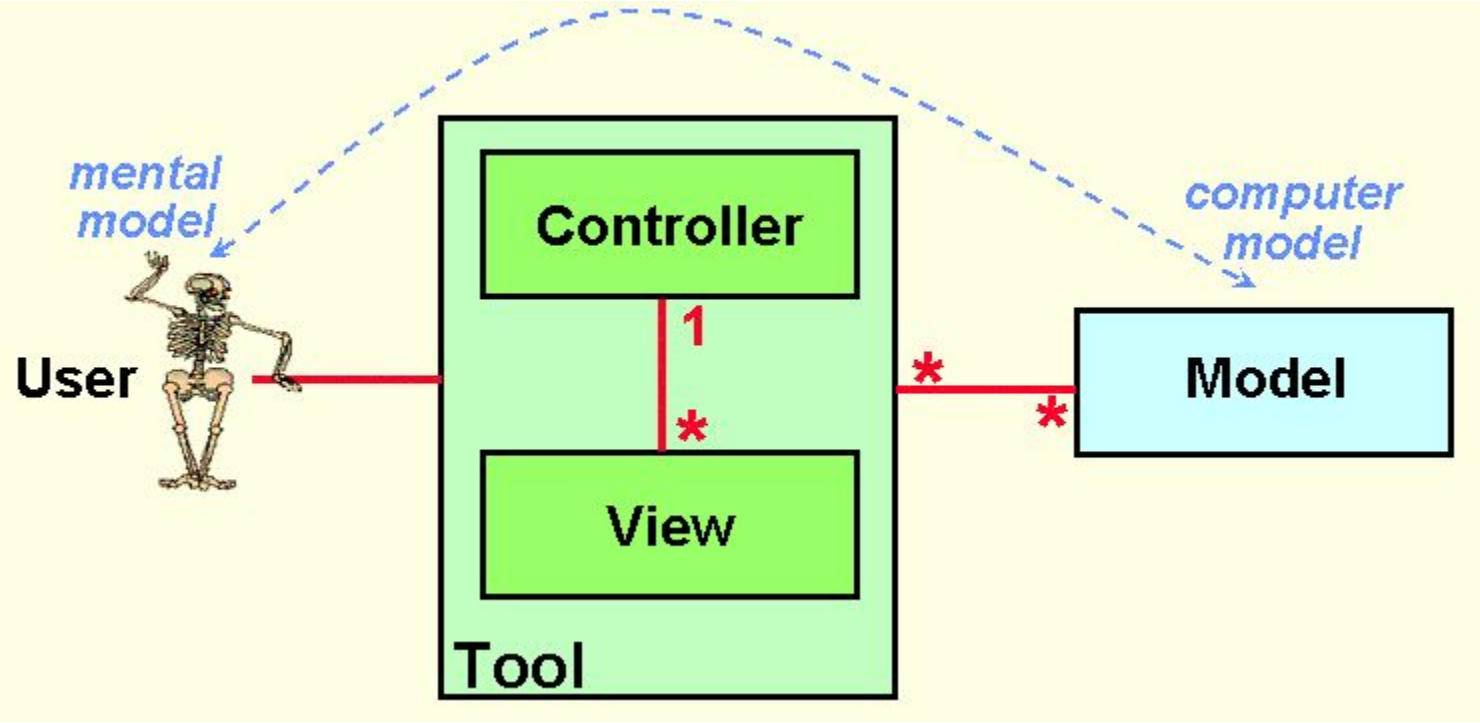
https://www.linkedin.com/in/towczarek/

MVC

JAVA FAKTURA

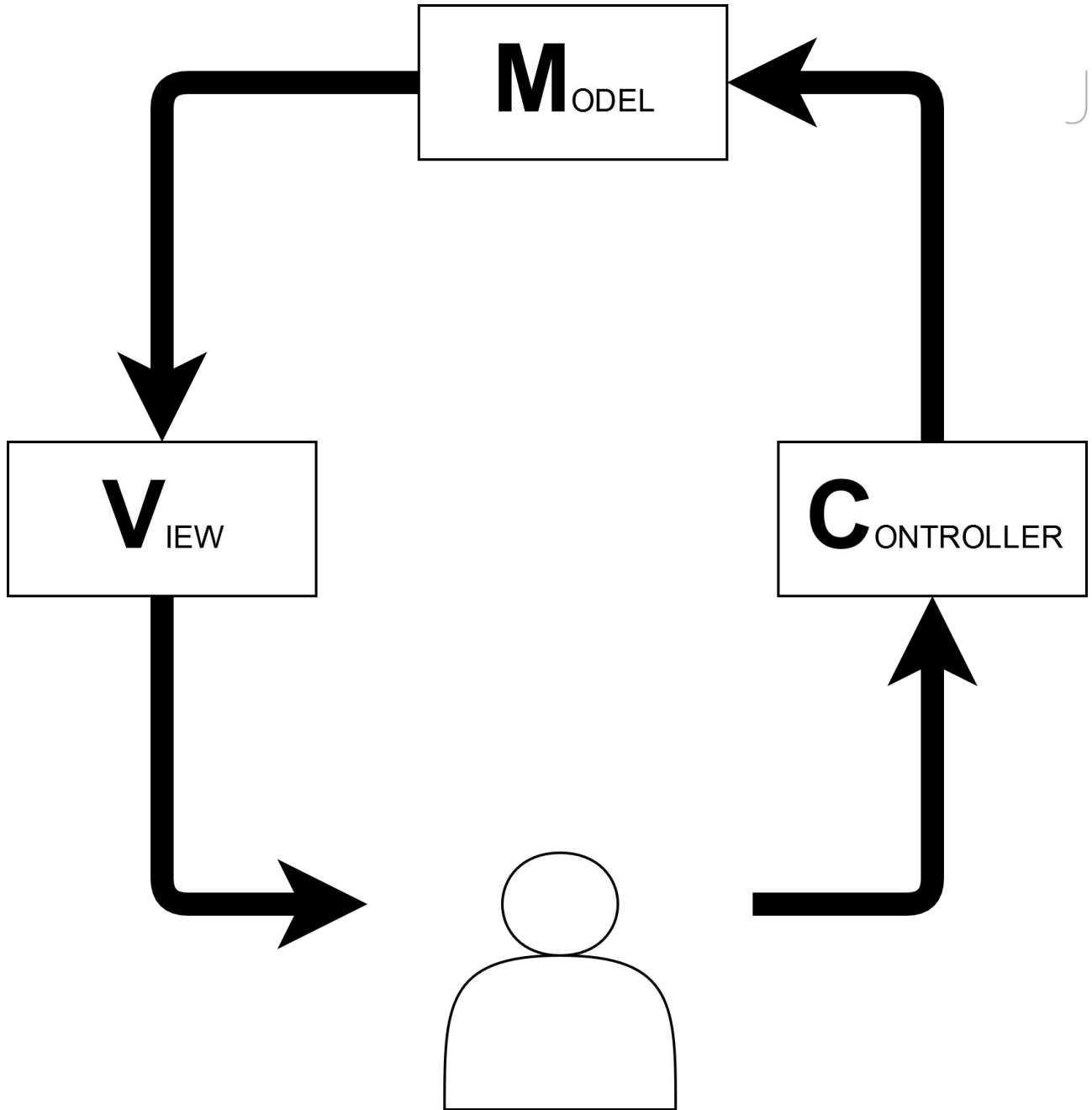
MVC

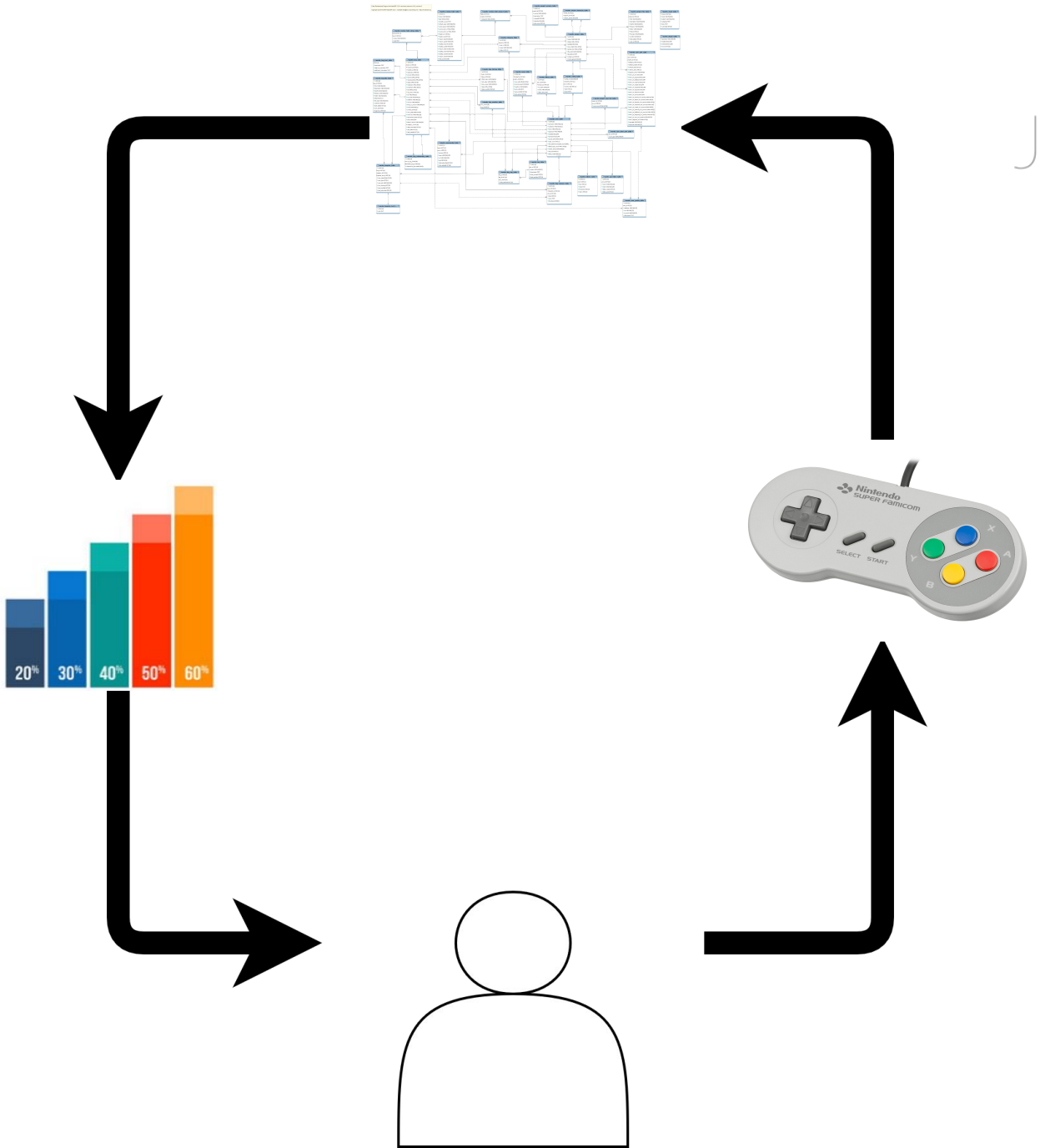
JAVA FAKTURA



1979 Trygve Reenskaug

http://heim.ifi.uio.no/~trygver/2007/MVC_Originals.pdf





JAVA FAKTURA

JAVA FAKTURA

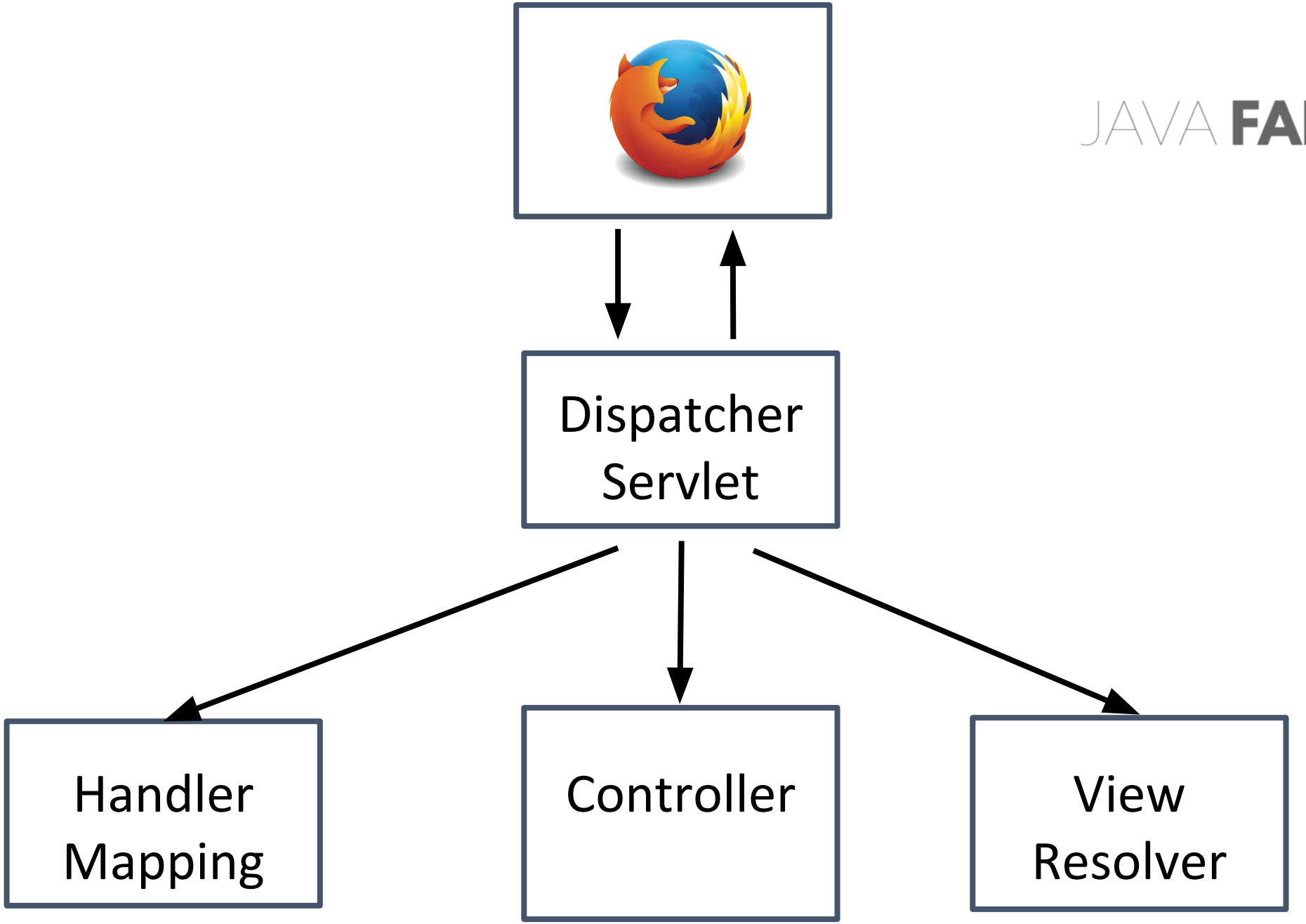
Spring MVC

JAVA FAKTURA

DispatcherServlet

JAVA FAKTURA

JAVA FAKTURA



JAVA FAKTURA

JAVA FAKTURA

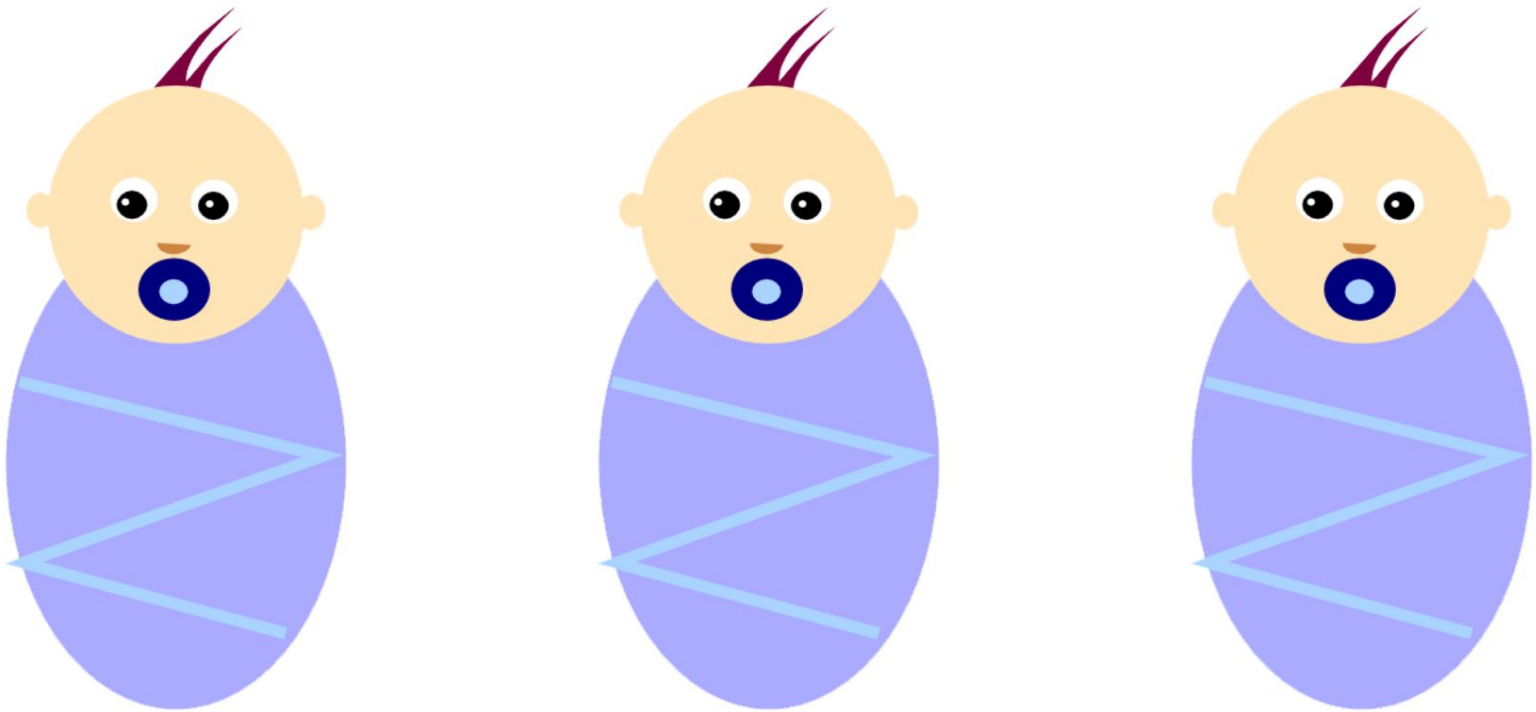
`org.springframework.web.servlet.DispatcherServlet`

JAVA FAKTURA

Demo?

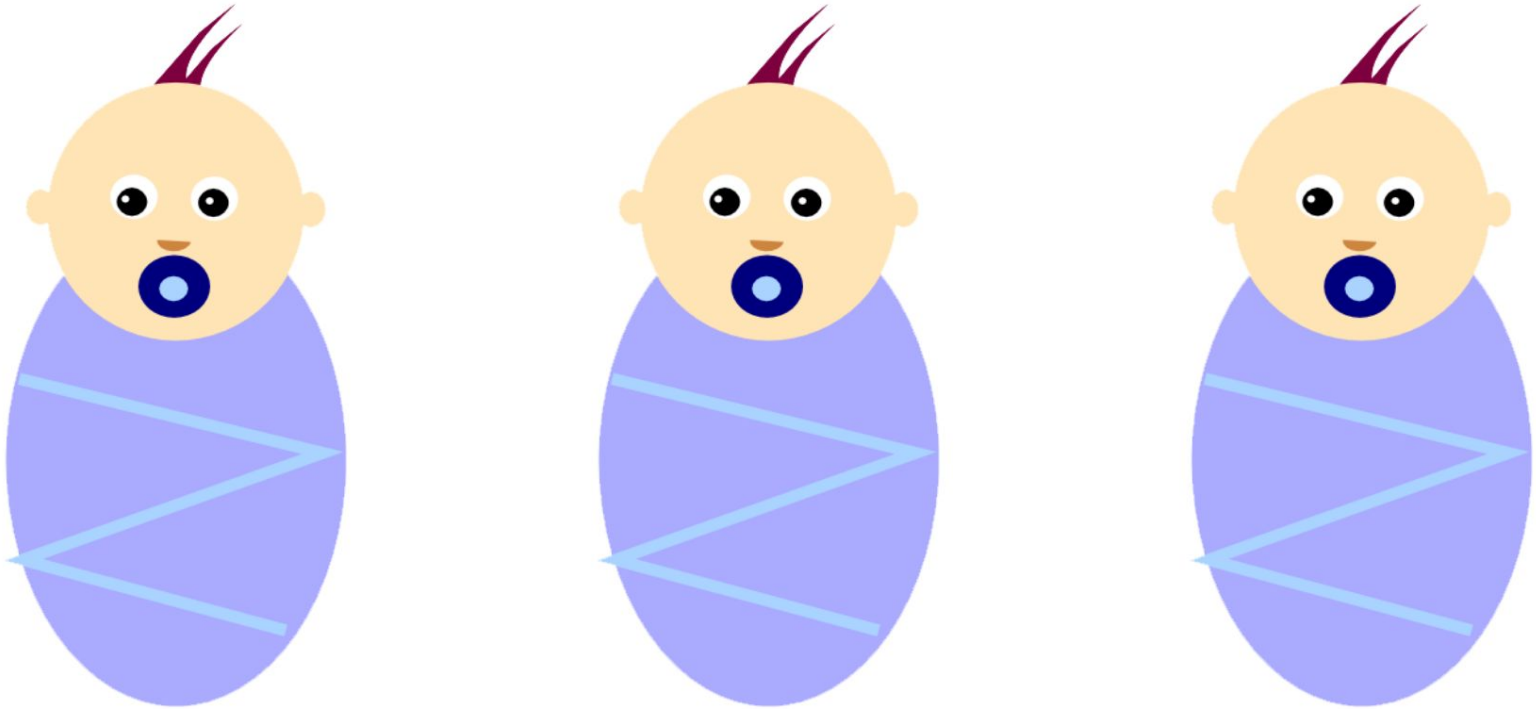
JAVA FAKTURA

JAVA FAKTURA



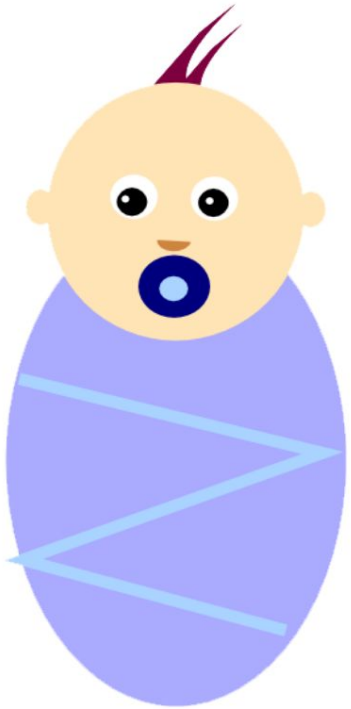
JAVA FAKTURA

JAŚ

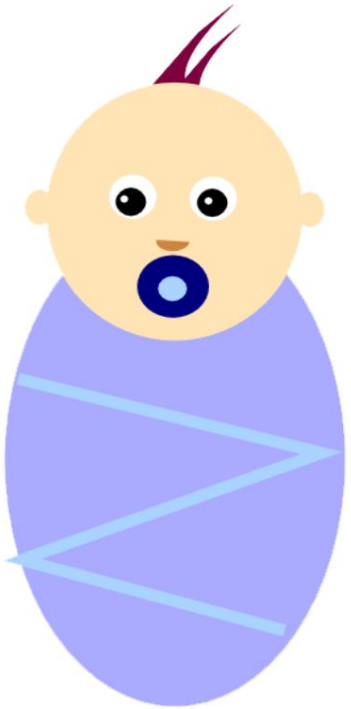


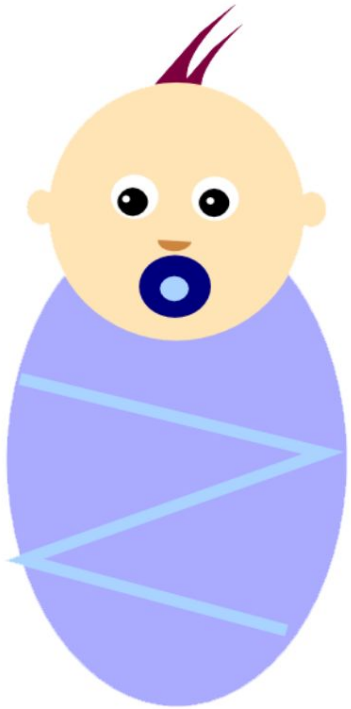
JAVA FAKTURA

JAŚ



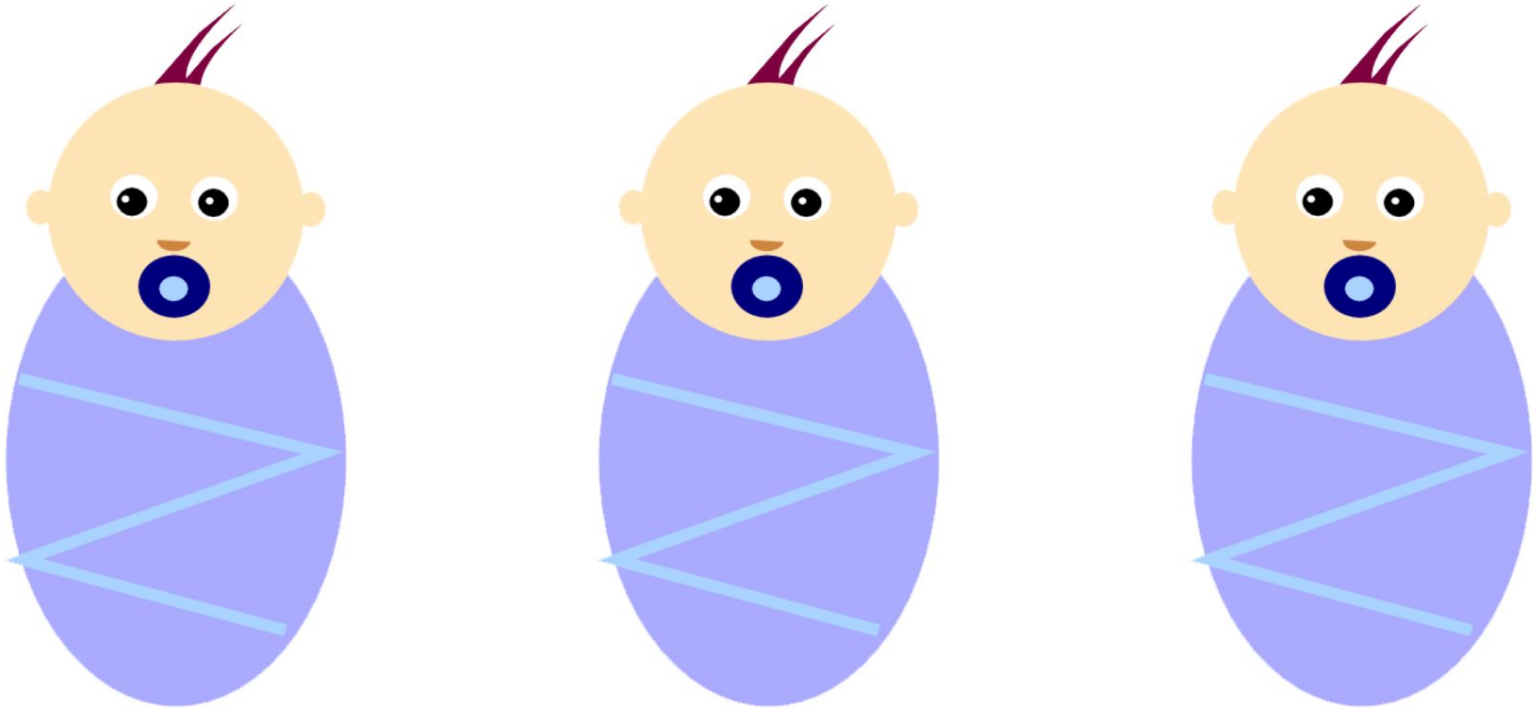
STAŚ





JAVA FAKTURA

JAŚ STAŚ ArrayIndexOutOfBoundsException



JAVA FAKTURA

https://dane.gov.pl

- Imiona nadawane dzieciom w pierwszej połowie 2019
- Częstość imion w latach 1950-2016

JAVA FAKTURA

child-core

```java
public interface ChildNameService {
   List<ChildNameStats> getAll(ParentPreferences preferences);
   int countAllOccurrences();
   ChildNameStats add(String name);
   ChildNameStats getRandom(ParentPreferences preferences);
   ChildNameStats lookFor(String name);
   Optional<ChildNameHistoricalStats> getHistoricalStats(String name);
   Integer getHistoricalOccurences(String name, Year year);
}
```

child-webapp

JAVA FAKTURA

child-webapp

child-core

```
public interface ChildNameService {
  List<ChildNameStats> getAll(ParentPreferences preferences);
  int countAllOccurrences();
  ChildNameStats add(String name);
  ChildNameStats getRandom(ParentPreferences preferences);
  ChildNameStats lookFor(String name);
  Optional<ChildNameHistoricalStats> getHistoricalStats(String name);
  Integer getHistoricalOccurences(String name, Year year);
}
```

JAVA FAKTURA

spring

child-webapp

BERENIKA

Wybór 15 rodziców w pierwszej połowie 2019 roku

Java

child-core

```
public interface ChildNameService {
  List<ChildNameStats> getAll(ParentPreferences preferences);
  int countAllOccurrences();
  ChildNameStats add(String name);
  ChildNameStats getRandom(ParentPreferences preferences);
  ChildNameStats lookFor(String name);
  Optional<ChildNameHistoricalStats> getHistoricalStats(String name);
  Integer getHistoricalOccurences(String name, Year year);
}
```

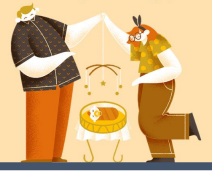
JAVA FAKTURA

spring

child-webapp

BERENIKA

Wybór 15 rodziców w pierwszej połowie 2019 roku



```
@RequestMapping("/")
        @RequestParam(required = false) Gender gender,
        @RequestParam(required = false) Popularity popularity

@RequestMapping(value = "/{name}")
        @PathVariable String name

@RequestMapping("/all")

@RequestMapping(value = "/choice", method = RequestMethod.POST)
        @ModelAttribute ParentChoice choice
```

Java

child-core

```
public interface ChildNameService {
   List<ChildNameStats> getAll(ParentPreferences preferences);
   int countAllOccurrences();
   ChildNameStats add(String name);
   ChildNameStats getRandom(ParentPreferences preferences);
   ChildNameStats lookFor(String name);
   Optional<ChildNameHistoricalStats> getHistoricalStats(String name);
   Integer getHistoricalOccurences(String name, Year year);
}
```

JAVA FAKTURA

spring

child-webapp

VIEW

CONTROLLER

```
@RequestMapp
      @Requestl                    false) Gender gender,
      @RequestPara                 ) Popularity popularity

@RequestMapping(value =
      @PathVariable String name

@RequestMapping("/all")

@RequestMapping(value = "/choice", method =                .nod.POST)
      @ModelAttribute ParentChoice choice
```

MODEL

hild-core

```
public interface ChildN
  List<ChildNameStats> ge
  int countAllOccurrences();
  ChildNameStats add(String name
  ChildNameStats getRandom(Parent
  ChildNameStats lookFor(String name);
  Optional<ChildNameHistoricalStats> getHist                , name);
  Integer getHistoricalOccurences(String name, Ye
}
```

Demo

JAVA FAKTURA

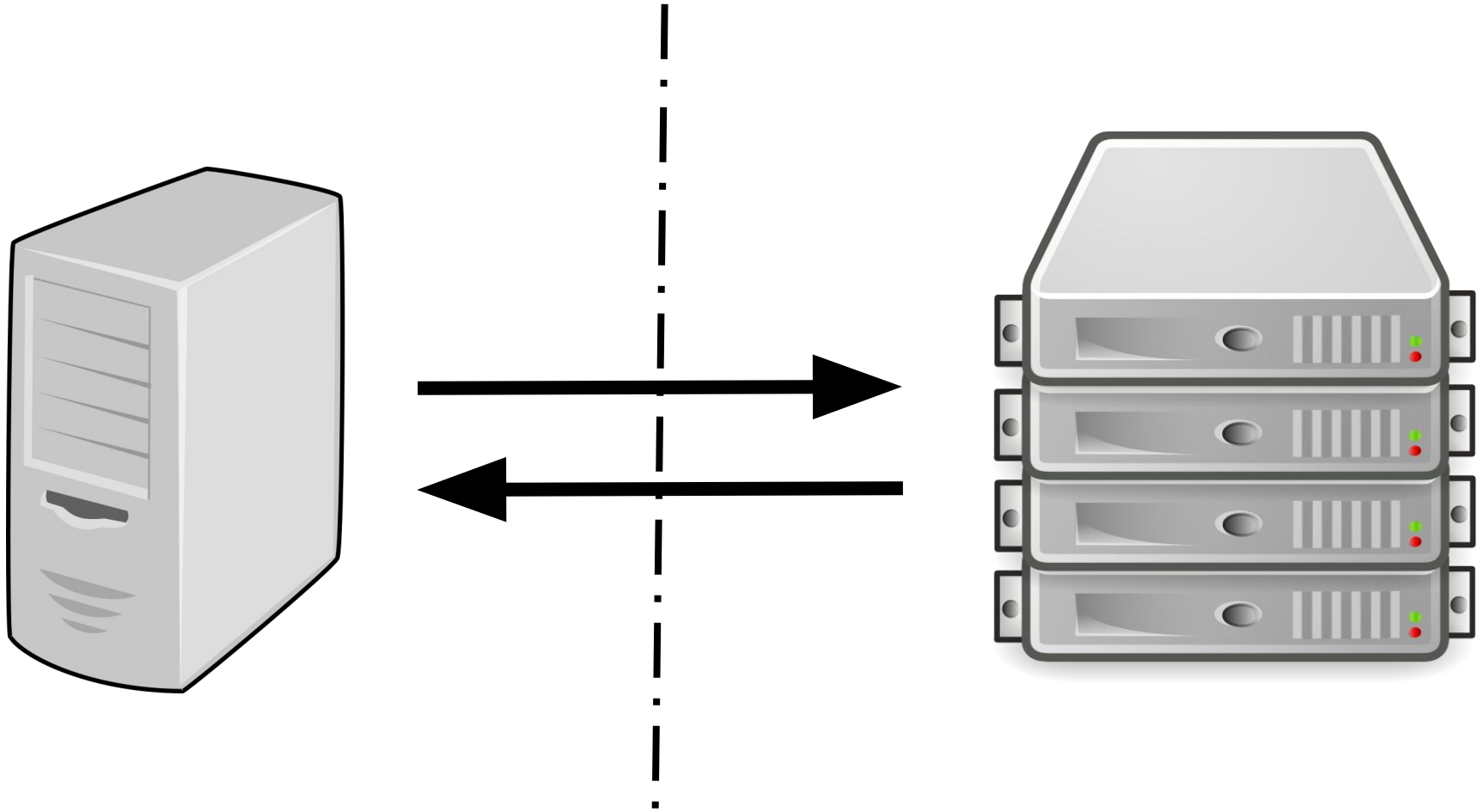
REST

Główne założenia

1. Klient i Serwer

JAVA FAKTURA

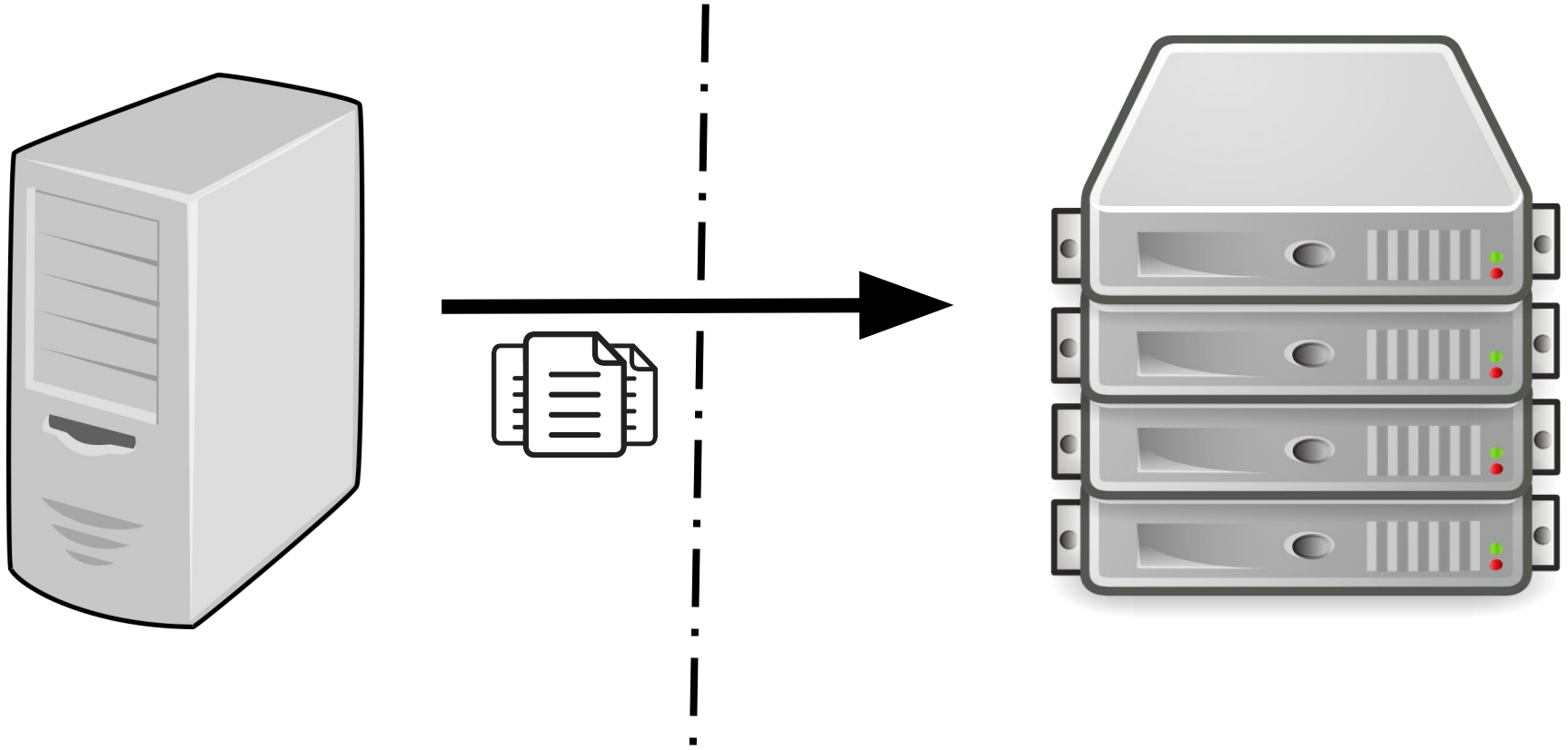
Separacja klienta i serwera



2. Bezstanowy serwer

JAVA FAKTURA

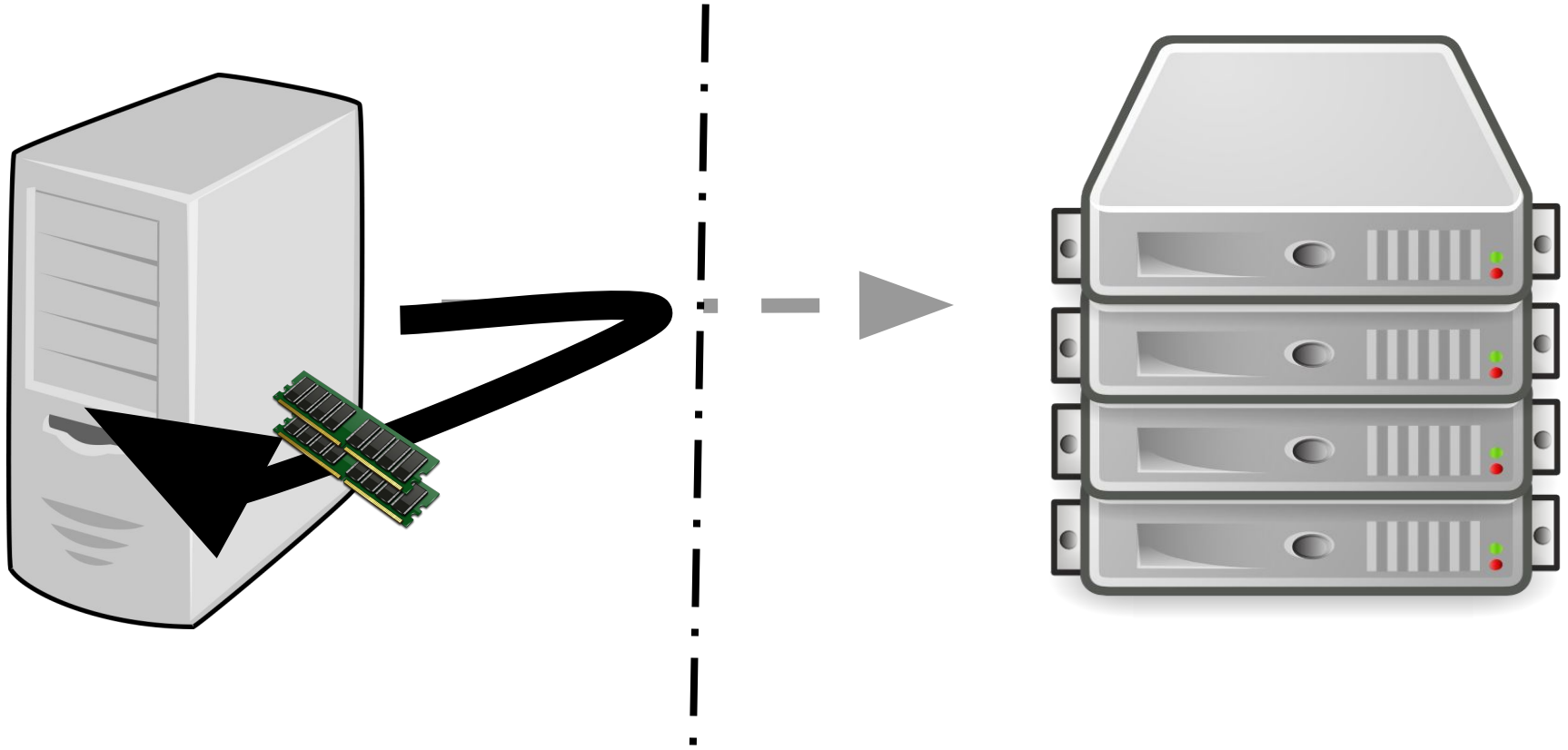
Każde pojedyncze żądanie od klienta zawiera
komplet niezbędnych informacji aby go obsłużyć po stronie serwera



3. Pamięć podręczna

JAVA FAKTURA

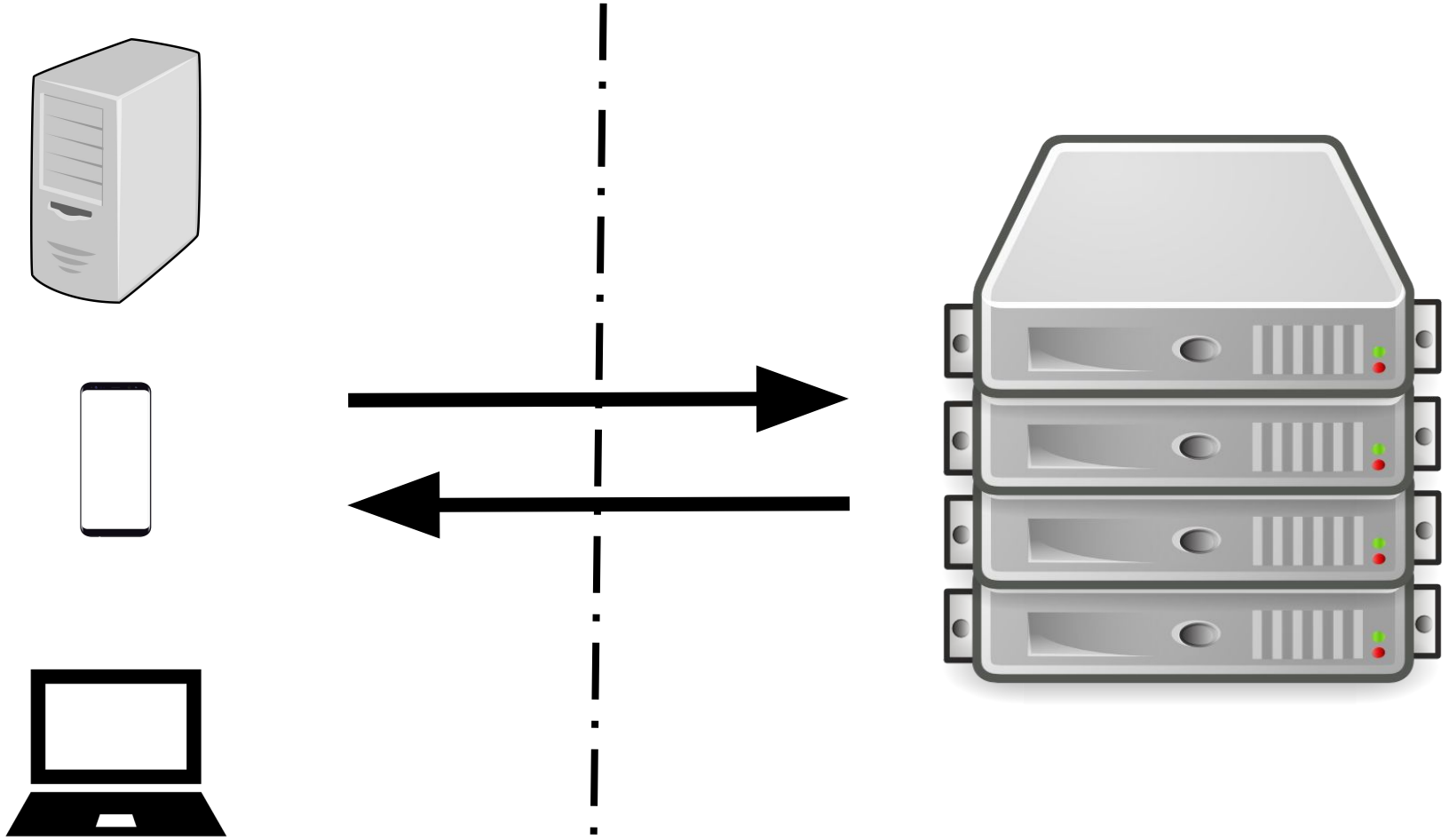
Klient może cache'ować odpowiedzi,
odpowiedzi powinny wskazywać, czy jest to możliwe



4. Ujednolicony interfejs

JAVA FAKTURA

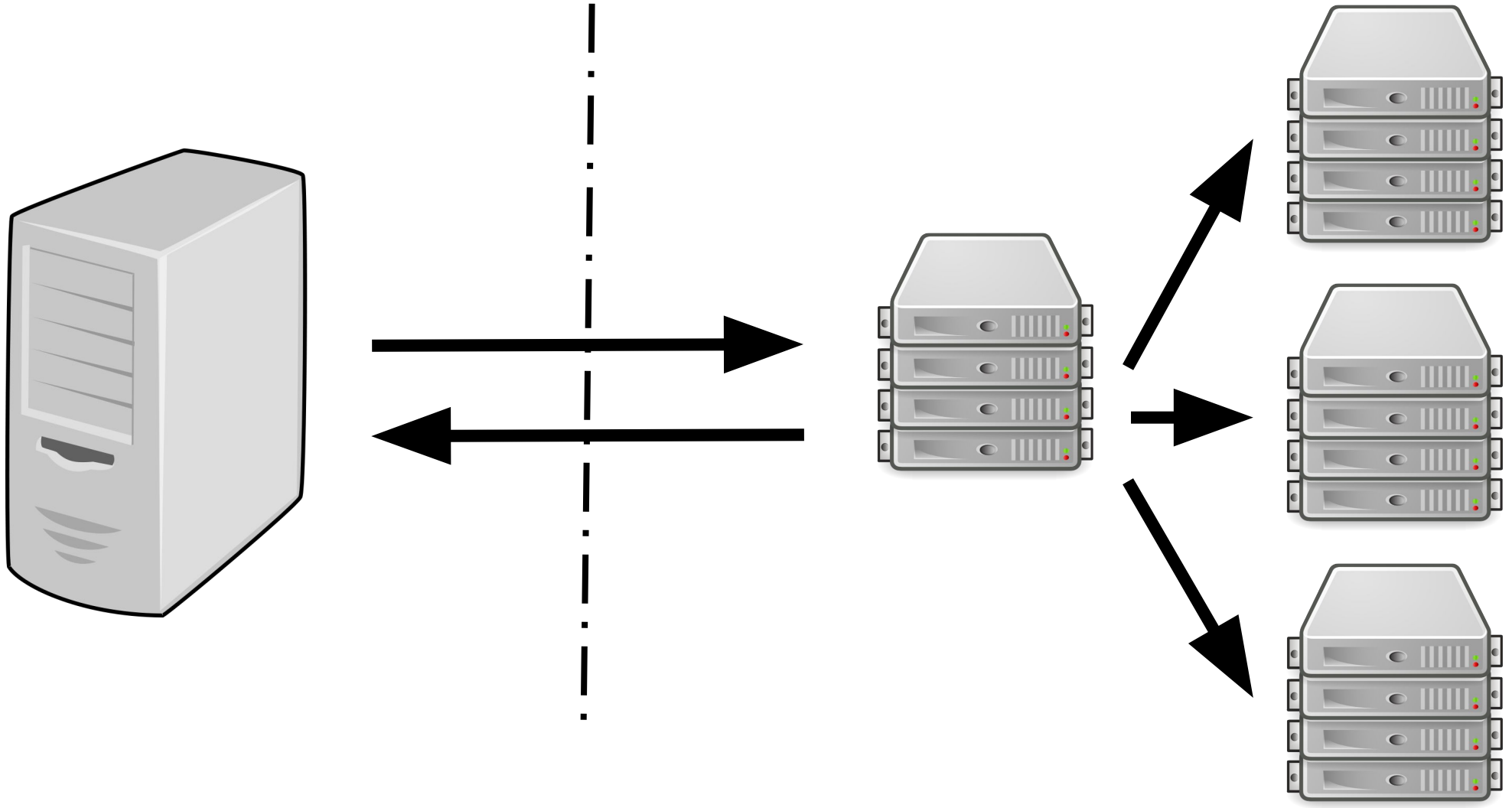
Jednolity interfejs pomiędzy klientem i serwerem



5. System warstwowy

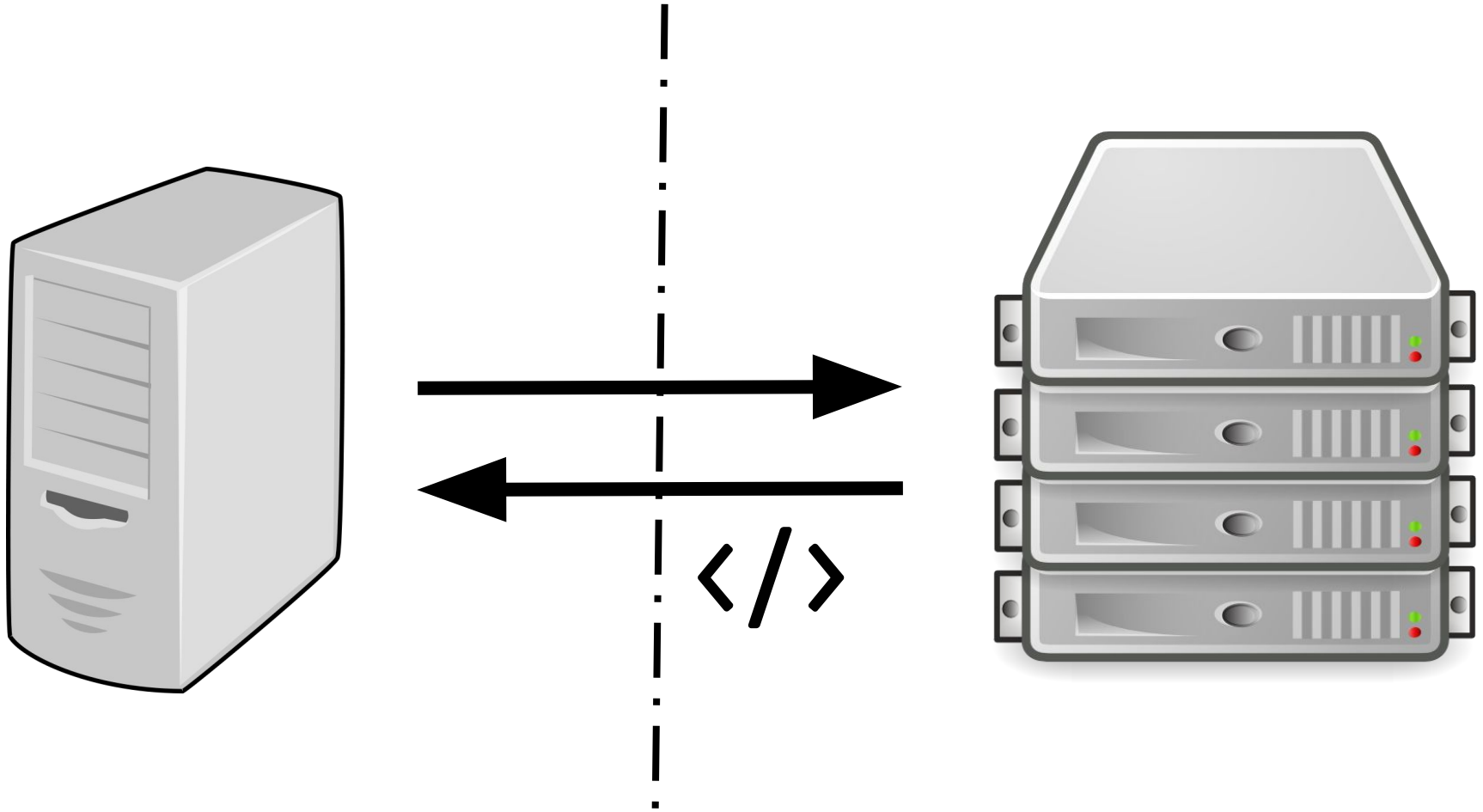
JAVA FAKTURA

powinien wspierać idee, load balancerów, proxy czy firewalli



6. Kod na żądanie (opcjonalnie)

JAVA FAKTURA



Główne założenia:

JAVA FAKTURA

1. **Klient i Serwer** - Separacja klienta i serwera

2. **Bezstanowy serwer** - każde pojedyncze żądanie od klienta zawiera komplet niezbędnych informacji aby go obsłużyć po stronie serwera

3. **Pamięć podręczna** - klient może cache'ować odpowiedzi, odpowiedzi powinny wskazywać, czy jest to możliwe

4. **Ujednolicony interfejs** - jednolity interfejs pomiędzy klientem i serwerem

5. **System warstwowy** - powinien wspierać idee, load balancerów, proxy czy firewalli

6. **Kod na żądanie** (opcjonalnie)

Główne założenia:

JAVA FAKTURA

1. **Klient i Serwer** - Separacja klienta i serwera

2. **Bezstanowy serwer** - każde pojedyncze żądanie od klienta zawiera komplet niezbędnych informacji aby go obsłużyć po stronie serwera

3. **Pamięć podręczna** - klient może cache'ować odpowiedzi, odpowiedzi powinny wskazywać, czy jest to możliwe

4. **Ujednolicony interfejs -** jednolity interfejs pomiędzy klientem i serwerem

5. **System warstwowy** - powinien wspierać idee, load balancerów, proxy czy firewalli

6. **Kod na żądanie** (opcjonalnie)

Ujednolicony interfejs

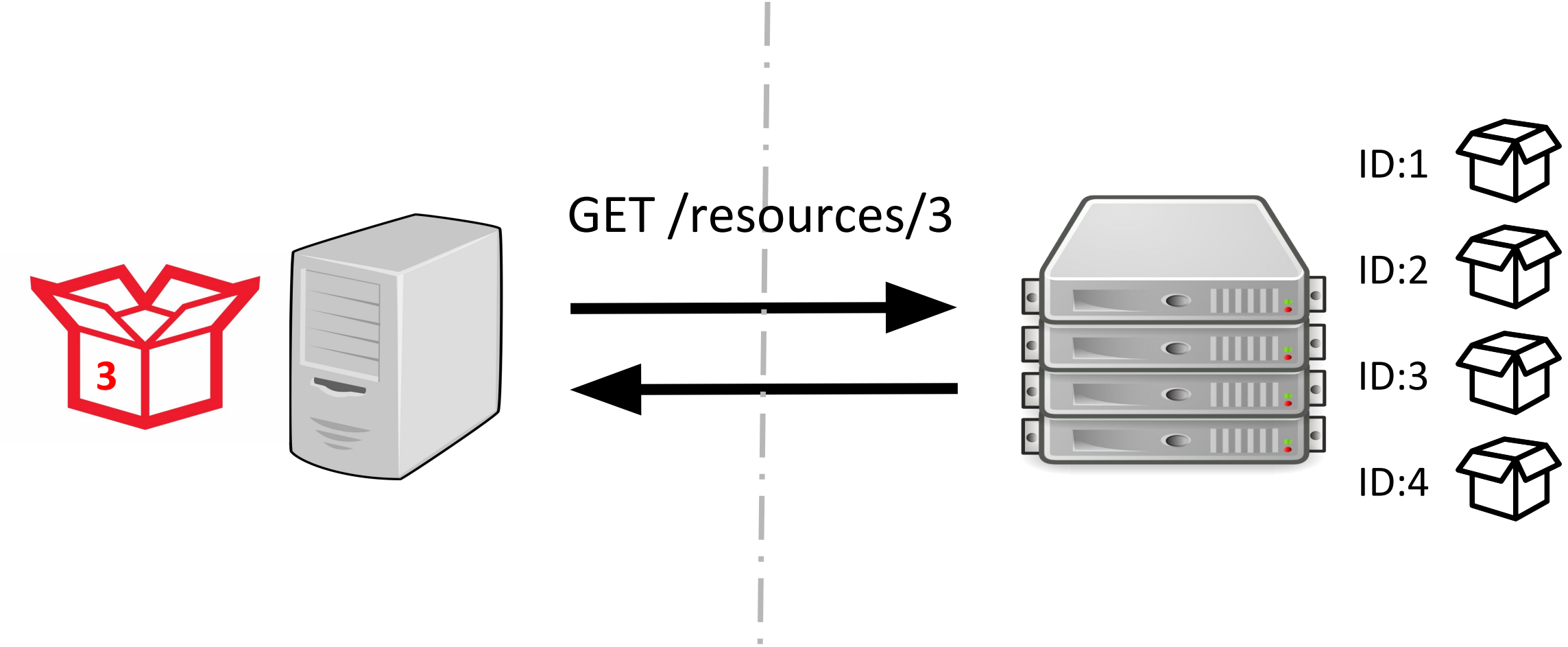
JAVA FAKTURA

4 ograniczenia:

- Identyfikacja zasobów w żądaniach
- Manipulowanie zasobami poprzez reprezentacje
- Samoopisujące się wiadomości
- Hipermedia jako silnik stanu aplikacji (HATEOAS)

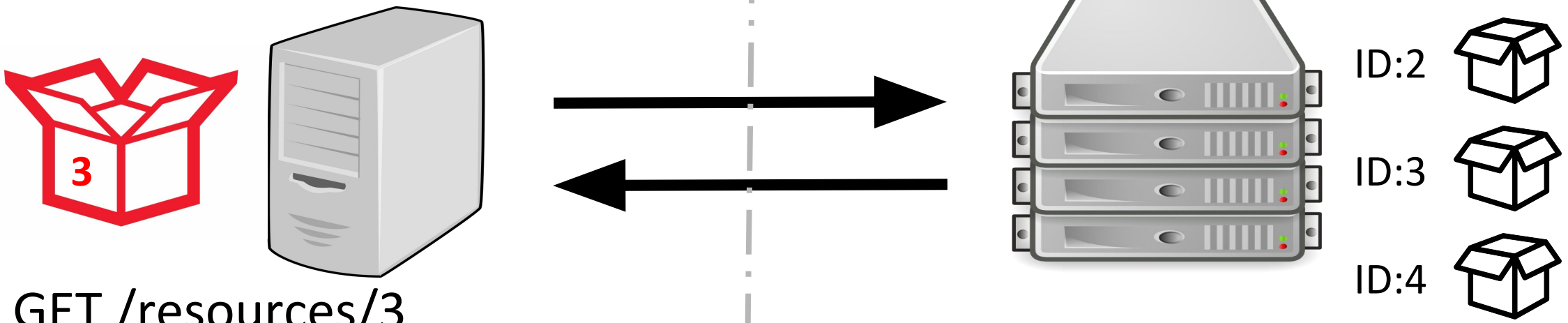
Identyfikacja zasobów w żądaniach

JAVA FAKTURA



Manipulowanie zasobami poprzez reprezentacje

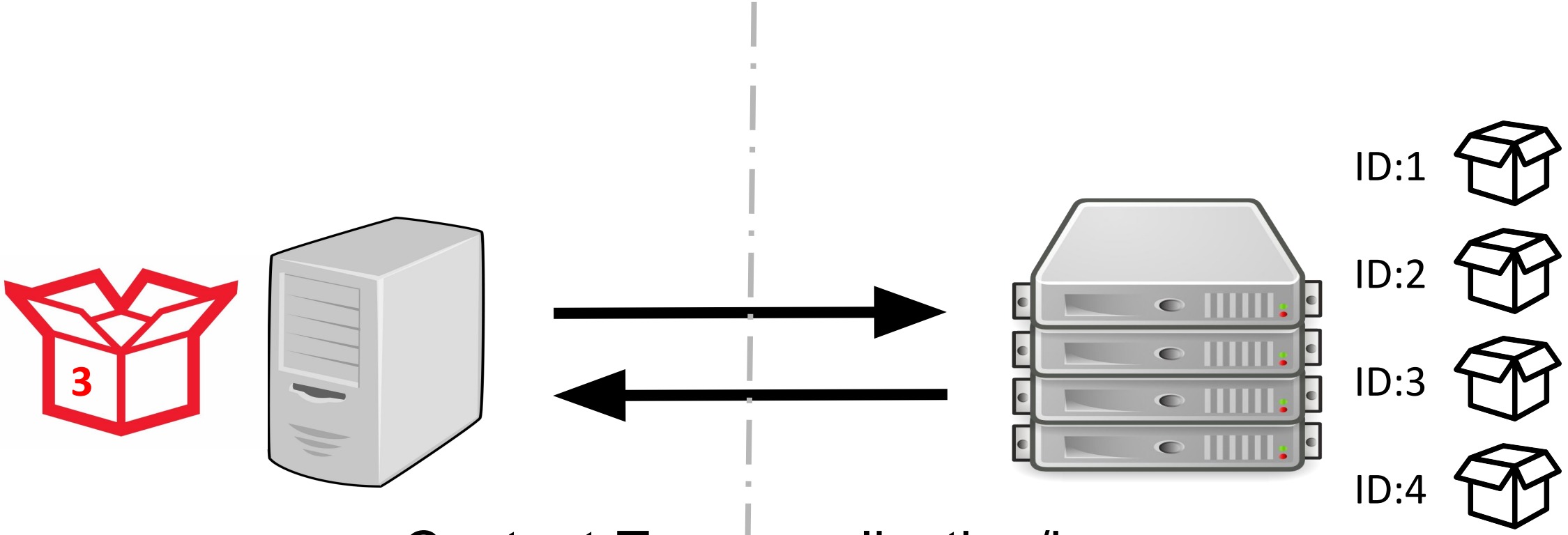
JAVA FAKTURA



GET /resources/3
POST /resources
PUT /resources/3
DELETE /resources/3

Samoopisujące się wiadomości

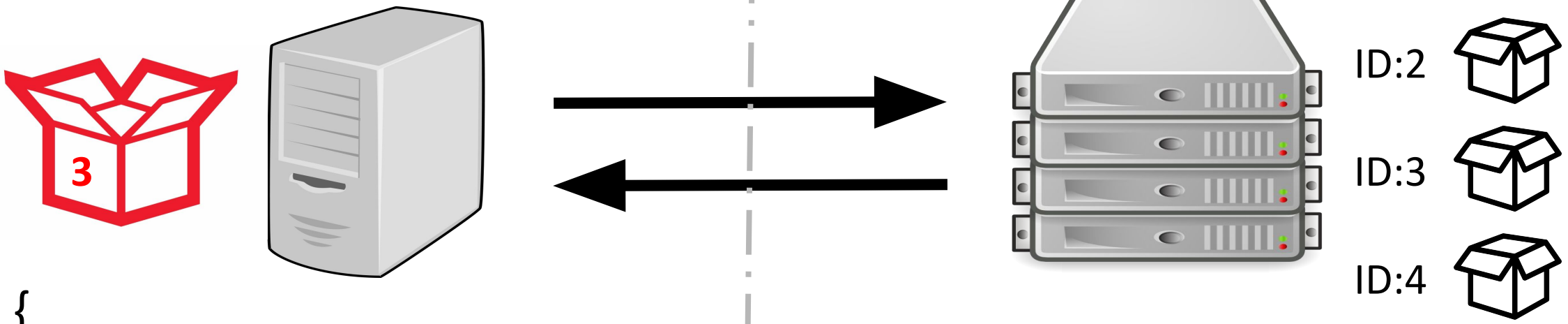
JAVA FAKTURA



Content-Type: application/json
Accept: application/json

Hipermedia jako silnik stanu aplikacji (HATEOAS)

JAVA FAKTURA




```
{
    …
    “links”: ...
}
```

JAVA FAKTURA

REST

Dobre praktyki projektowania

GET /all_stats/childName/{name}/json/

JAVA FAKTURA

GET /all_stats/childName/{name}/json/

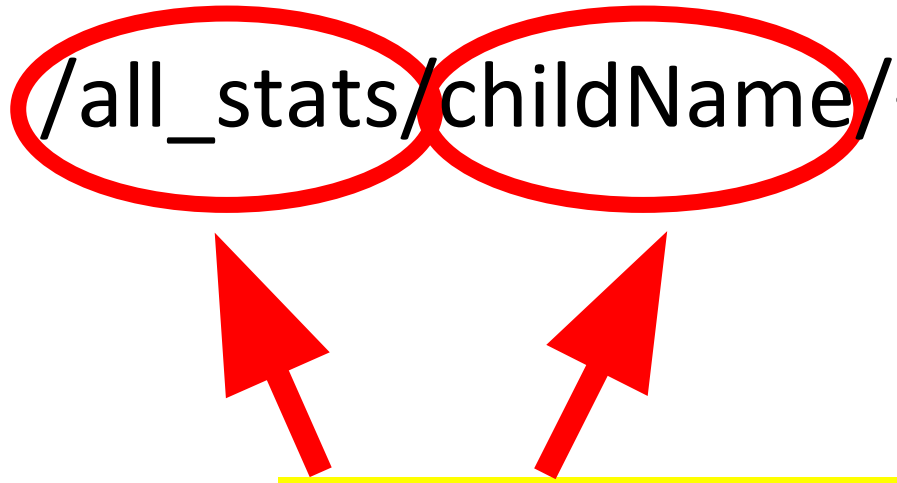
JAVA FAKTURA

GET /all_stats/childName/{name}/json/

JAVA FAKTURA

#1: “/” nie powinien być uwzględniany w URI na końcu

GET /all_stats/childName/{name}/json/

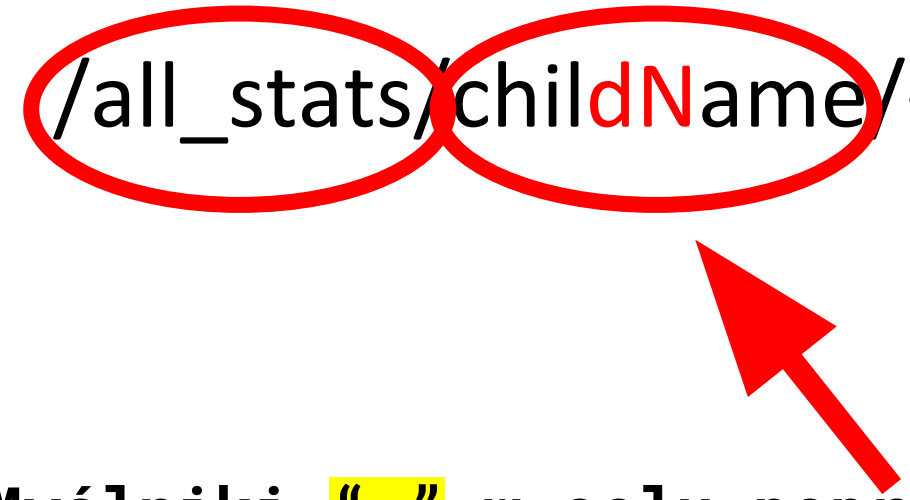


JAVA FAKTURA

#2: Modeluj relację hierarchiczną używając “/”

#1: “/” nie powinien być uwzględniany w URI na końcu

GET /all_stats/childName/{name}/json/



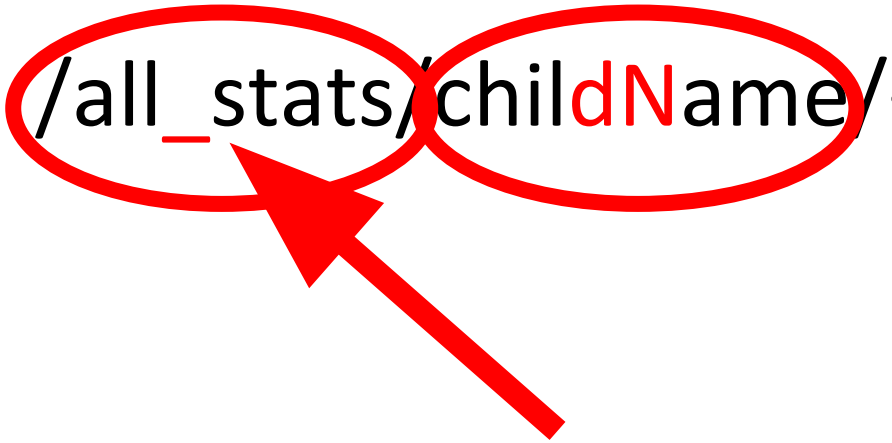
JAVA FAKTURA

#3: Myślniki “-” w celu poprawy czytelności URI

#2: Modeluj relację hierarchiczną używając “/”

#1: “/” nie powinien być uwzględniany w URI na końcu

GET /all_stats/childName/{name}/json/



JAVA FAKTURA

#4: Brak podkreślników “_” w URI

#3: Myślniki “-” w celu poprawy czytelności URI

#2: Modeluj relację hierarchiczną używając “/”

#1: “/” nie powinien być uwzględniany w URI na końcu

GET /all_stats/childName/{name}/json/

JAVA FAKTURA

#5: Stosuj tylko małe litery w ścieżkach URI

#4: Brak podkreślników “_” w URI

#3: Myślniki “-” w celu poprawy czytelności URI

#2: Modeluj relację hierarchiczną używając “/”

#1: “/” nie powinien być uwzględniany w URI na końcu

GET /all_stats/childName/{name}/json/

JAVA FAKTURA

#6: Bez rozszerzeń w URI

#5: Stosuj tylko małe litery w ścieżkach URI

#4: Brak podkreślników “_” w URI

#3: Myślniki “-” w celu poprawy czytelności URI

#2: Modeluj relację hierarchiczną używając “/”

#1: “/” nie powinien być uwzględniany w URI na końcu

GET /all_stats/childName/{name}/json/

JAVA FAKTURA

#7: Liczba mnoga w nazewnictwie zasobów

#6: Bez rozszerzeń w URI

#5: Stosuj tylko małe litery w ścieżkach URI

#4: Brak podkreślników “_” w URI

#3: Myślniki “-” w celu poprawy czytelności URI

#2: Modeluj relację hierarchiczną używając “/”

#1: “/” nie powinien być uwzględniany w URI na końcu

GET /child-names/{name}/stats

JAVA FAKTURA

#7: Liczba mnoga w nazewnictwie zasobów
#6: Bez rozszerzeń w URI
#5: Stosuj tylko małe litery w ścieżkach URI
#4: Brak podkreślników “_” w URI
#3: Myślniki “-” w celu poprawy czytelności URI
#2: Modeluj relację hierarchiczną używając “/”
#1: “/” nie powinien być uwzględniany w URI na końcu

JAVA FAKTURA

Demo?

JAVA FAKTURA

child-api

```
{
 "name": "JAVAFAKTURA",
 "occurrences": 2,
 "gender": "FEMALE"
}
```

child-core

```
public interface ChildNameService {
  List<ChildNameStats> getAll(ParentPreferences preferences);
  int countAllOccurrences();
  ChildNameStats add(String name);
  ChildNameStats getRandom(ParentPreferences preferences);
  ChildNameStats lookFor(String name);
  Optional<ChildNameHistoricalStats> getHistoricalStats(String name);
  Integer getHistoricalOccurences(String name, Year year);
}
```

```
@RequestMapping(path = "/child-names")
      @RequestParam(required = false) Gender gender,
      @RequestParam(required = false) Popularity popularity

@RequestMapping(path = "/child-names", method = RequestMethod.POST)
      @RequestBody ParentChoice choice

@RequestMapping(path = "/child-names/{name}")
      @PathVariable String name

@RequestMapping(path = "/child-names/random")
      @RequestParam(required = false) Gender gender,
      @RequestParam(required = false) Popularity popularity

@RequestMapping(path = "/child-names/{name}/history")
      @PathVariable String name
```

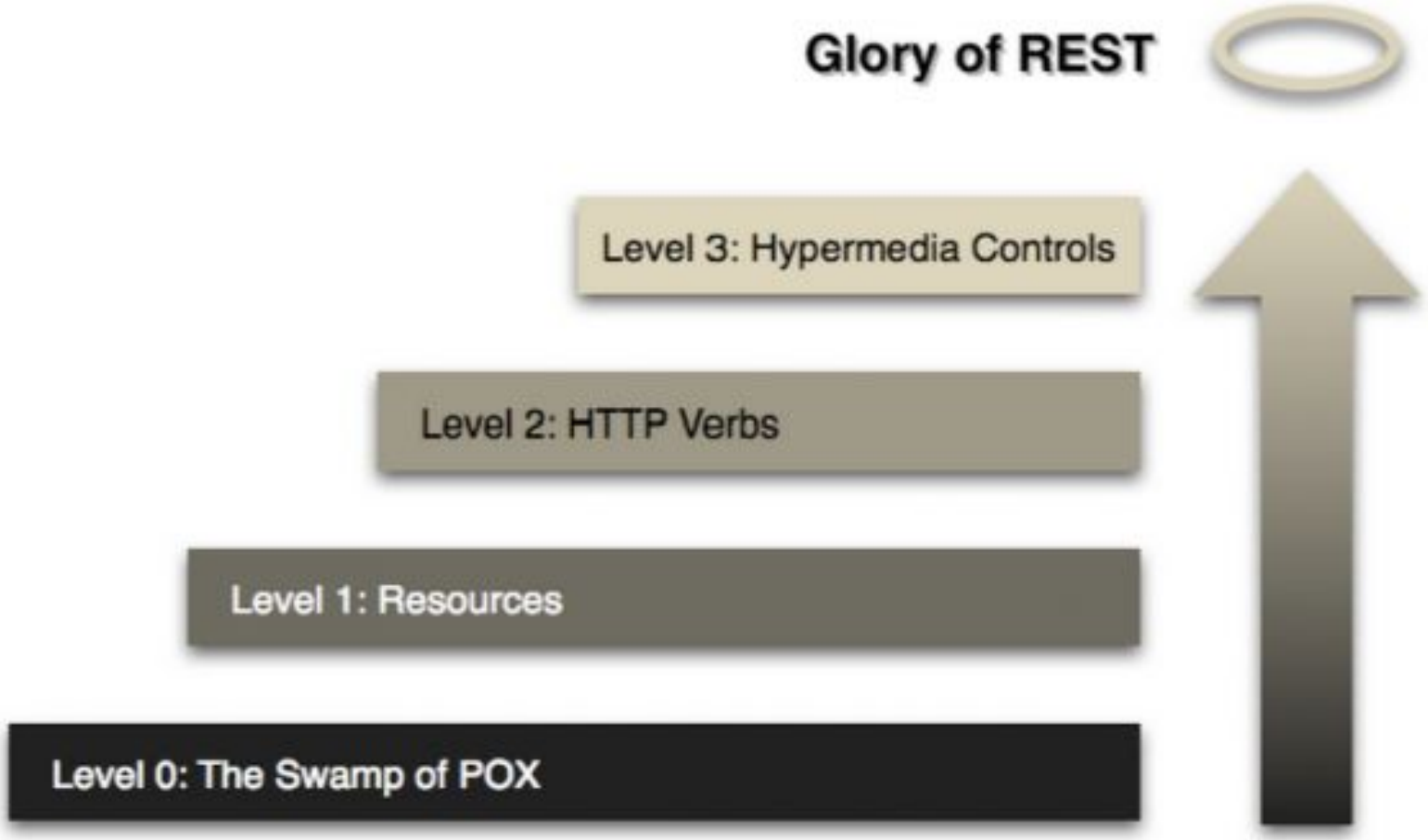
JAVA FAKTURA

REST

4 Poziomy Dojrzałości

Leonard Richardson

JAVA FAKTURA



Poziom 0:

JAVA FAKTURA

Poziom 0:

JAVA FAKTURA

Bagno POX (Plain Old XML)

POST `/child-name-service`

- Jedno URI, jedna metoda HTTP
- Wielka “czarna skrzynka”

```
"childNameServiceRequest": {
  "getAll": {
    "parentPreferences": {
      "gender": "MALE",
      "popularity": "UNIQUE"
    }
  }
}
```

Poziom 1:

JAVA FAKTURA

Poziom 1:

JAVA FAKTURA

Zasoby

- Każdy zasób ma **unikalne URI**
- Używana pojedyncza metoda HTTP (POST lub GET)
- Czasowniki nie mają znaczenia

POST `/child-names/all`

```
"parentPreferences": {
 "gender": "MALE",
 "popularity": "UNIQUE"
}
```

Poziom 2:

JAVA FAKTURA

Poziom 2:

JAVA FAKTURA

Metody HTTP

- **Dużo URI** i wiele **metod HTTP**
- Poprawne użycie kodów odpowiedzi
- Ujawniają **stan**, nie **zachowanie**
- Serwisy typu **CRUD**

GET `/child-names?gender=male&popularity=unique`


```
[
 {
  "name": "ANTONI",
  "occurrences": 4247,
  "gender": "MALE"
 },
 {
  "name": "JAKUB",
  "occurrences": 4050,
  "gender": "MALE"
 },...
]
```

Poziom 2:

JAVA FAKTURA

Metody HTTP

GET	- CZYTAJ
POST	- STWÓRZ
DELETE	- USUŃ
PUT	- ZASTĄP
PATCH	- ZAKTUALIZUJ (część)

JAVA FAKTURA


```
org.springframework.web.bind.annotation.RequestMapping
                                        GetMapping
                                        PostMapping
                                        PutMapping
                                        DeleteMapping
                                        PatchMapping
```

Poziom 2:

JAVA FAKTURA

Kody odpowiedzi HTTP

1XX	- CHWILECZKĘ
2XX	- PROSZĘ
3XX	- IDŹ SOBIE
4XX	- POPSUŁEŚ
5XX	- JA POPSUŁEM

JAVA FAKTURA

`org.springframework.http.ResponseEntity<T>`

`org.springframework.http.HttpStatus`

Poziom 3:

JAVA FAKTURA

Poziom 3:

JAVA FAKTURA

`Kontrola na bazie hipermediów`

- Zasoby są **samoopisujące się**
- **H**ypermedia **A**s **T**he **E**ngine **O**f **A**pplication **S**tate (**HATEOAS**)
- Ujawniają zarówno **stan**, jak i **zachowanie**

GET `/child-names?gender=male`


```
{
 "childNames": [
  {
   "name": "XSAVIER",
   "occurrences": 2,
   "gender": "MALE"
  },...
 ],
 "_links": {
  "self": {
   "href": "http://localhost:8080/child-names?gender=male&page=36"
  },
  "prev": {
   "href": "http://localhost:8080/child-names?gender=male&page=35"
  },
  "next": {
   "href": "http://localhost:8080/child-names?gender=male&page=37"
  }
 }
}
```

JAVA FAKTURA

Czy poziomy 0, 1, 2 są już RESTful?

JAVA FAKTURA

"Co należy zrobić, aby stało się jasne, że ***hiperłącza są wymogiem*** *przy projektowaniu* ***API REST****owych? Innymi słowy, jeżeli* ***silnik stanu aplikacji nie jest sterowany przez hiperłącza****,* ***nie może być nazywany RESTful*** *i nie może być API RESTowym.* ***Kropka****. Czy istnieje gdzieś jakaś błędna dokumentacja, która powinna być naprawiona?"*

Roy Thomas Fielding

JAVA FAKTURA

REST

Konsumowanie API

JAVA FAKTURA

`org.springframework.web.client.RestTemplate`

JAVA FAKTURA

spring

child-client


```
@RequestMapping("/")
        @RequestParam(required = false) Gender gender,
        @RequestParam(required = false) Popularity popularity

@RequestMapping(value = "/{name}")
        @PathVariable String name

@RequestMapping("/all")

@RequestMapping(value = "/choice", method = RequestMethod.POST)
        @ModelAttribute ParentChoice choice
```

```
public class ChildNameClientService {
   private final RestTemplate restTemplate;
}
```

spring

child-api

```
{
 "name":
"JAVAFAKTURA",
 "occurrences": 2,
 "gender": "FEMALE"
}
```

Java

child-core

```
public interface ChildNameService {
  List<ChildNameStats> getAll(ParentPreferences preferences);
  int countAllOccurrences();
  ChildNameStats add(String name);
  ChildNameStats getRandom(ParentPreferences preferences);
  ChildNameStats lookFor(String name);
  Optional<ChildNameHistoricalStats> getHistoricalStats(String
name);
  Integer getHistoricalOccurences(String name, Year year);
}
```

```
@RequestMapping(path = "/child-names")
        @RequestParam(required = false) Gender gender,
        @RequestParam(required = false) Popularity
        popularity

@RequestMapping(path = "/child-names",
method = RequestMethod.POST)
        @RequestBody ParentChoice choice

@RequestMapping(path = "/child-names/{name}")
        @PathVariable String name

@RequestMapping(path = "/child-names/random")
        @RequestParam(required = false) Gender gender,
        @RequestParam(required = false) Popularity
        popularity

@RequestMapping(path = "/child-names/{name}/history")
        @PathVariable String name
```

JAVA FAKTURA

spring

child-client

6

```
@RequestMapping("/")
        @RequestParam(required = false) Gender gender,
        @RequestParam(required = false) Popularity popularity

@RequestMapping(value = "/{name}")
        @PathVariable String name

@RequestMapping("/all")

@RequestMapping(value = "/choice", method = RequestMethod.POST)
        @ModelAttribute ParentChoice choice
```

5

```
public class ChildNameClientService {
   private final RestTemplate restTemplate;
}
```

4

spring

child-api

```
{
 "name":
"JAVAFAKTURA",
 "occurrences": 2,
 "gender": "FEMALE"
}
```

3

Java

child-core

```
public interface ChildNameService {
  List<ChildNameStats> getAll(ParentPreferences preferences);
  int countAllOccurrences();
  ChildNameStats add(String name);
  ChildNameStats getRandom(ParentPreferences preferences);
  ChildNameStats lookFor(String name);
  Optional<ChildNameHistoricalStats> getHistoricalStats(String name);
  Integer getHistoricalOccurences(String name, Year year);
}
```

1

```
@RequestMapping(path = "/child-names")
        @RequestParam(required = false) Gender gender,
        @RequestParam(required = false) Popularity popularity

@RequestMapping(path = "/child-names",
method = RequestMethod.POST)
        @RequestBody ParentChoice choice

@RequestMapping(path = "/child-names/{name}")
        @PathVariable String name

@RequestMapping(path = "/child-names/random")
        @RequestParam(required = false) Gender gender,
        @RequestParam(required = false) Popularity popularity

@RequestMapping(path = "/child-names/{name}/history")
        @PathVariable String name
```

2

JAVA FAKTURA

spring child-client

6

```
@RequestMapping("/")
        @RequestParam(required = false) Gender gender,
        @RequestParam(required = false) Popularity popularity

@RequestMapping(value = "/{name}")
        @PathVariable String name

@RequestMapping("/all")

@RequestMapping(value = "/choice", method = RequestMethod.POST)
        @ModelAttribute ParentChoice choice
```

5

```
public class ChildNameClientService {
   private final RestTemplate restTemplate;
}
```

4

spring child-api

```
{
 "name":
"JAVAFAKTURA",
 "occurrences": 2,
 "gender": "FEMALE"
}
```

3

Java child-core

```
public interface ChildNameService {
  List<ChildNameStats> getAll(ParentPreferences preferences);
  int countAllOccurrences();
  ChildNameStats add(String name);
  ChildNameStats getRandom(ParentPreferences preferences);
  ChildNameStats lookFor(String name);
  Optional<ChildNameHistoricalStats> getHistoricalStats(String name);
  Integer getHistoricalOccurences(String name, Year year);
}
```

1

```
@RequestMapping(path = "/child-names")
        @RequestParam(required = false) Gender gender,
        @RequestParam(required = false) Popularity
        popularity

@RequestMapping(path = "/child-names",
method = RequestMethod.POST)
        @RequestBody ParentChoice choice

@RequestMapping(path = "/child-names/{name}")
        @PathVariable String name

@RequestMapping(path = "/child-names/random")
        @RequestParam(required = false) Gender gender,
        @RequestParam(required = false) Popularity
        popularity

@RequestMapping(path = "/child-names/{name}/history")
        @PathVariable String name
```

2


JAVA FAKTURA
spring
child-client
BERENIKA
Wybór 15 rodziców w pierwszej połowie 2019 roku
6
@RequestMapping("/")
@RequestParam(required = fa
@RequestParam(required = fa
@RequestMapping(value = "/{name}")
@PathVariable String name
@RequestMapping("/all")
@RequestMapping(value = "/choice", m
@ModelAttribute ParentChoice
public class ChildNameCl
private final RestTempla
}
4
spring
child-api
ces": 2,
"FEMALE
3
child-core
public interface ChildNameService {
List<ChildNameStats> getAll(ParentPreferences preferences);
int countAllOccurrences();
ChildNameStats add(String name);
ChildNameStats getRandom(ParentPreferences preferen
ChildNameStats lookFor(String name);
Optional<ChildNameHistoricalStats> getHistoricalStats(String name);
Integer getHistoricalOccurences(String name, Year year);
}
1
@RequestMapping(path = "/child-names")
@RequestParam(required = false) Gender gender,
@RequestParam(required = false) Popularity
popularity
@RequestMapping(path = "/child-names",
method = RequestMethod.POST)
@RequestBody ParentChoice choice
@RequestMapping(path = "/child-names/{name}")
@PathVariable String name
@RequestMapping(path = "/child-names/random")
@RequestParam(required = false) Gender gender,
@RequestParam(required = false) Popularity
popularity
@RequestMapping(path = "/child-names/{name}/history")
@PathVariable String nam
2

Demo

JAVA FAKTURA

Message More...

Marcin Grzejszczak · 1st

Principal Software Engineer at Pivotal

Warsaw, Masovian District, Poland · 500+ connections · Contact info

Pivotal Software, Inc.

Lodz University of Technology

Dziękuję za uwagę

JAVA FAKTURA

tinyurl.com/tomekprosi