{ REST } GOODBYE

WELCOME GRPC

PAWEŁ REKOWSKI

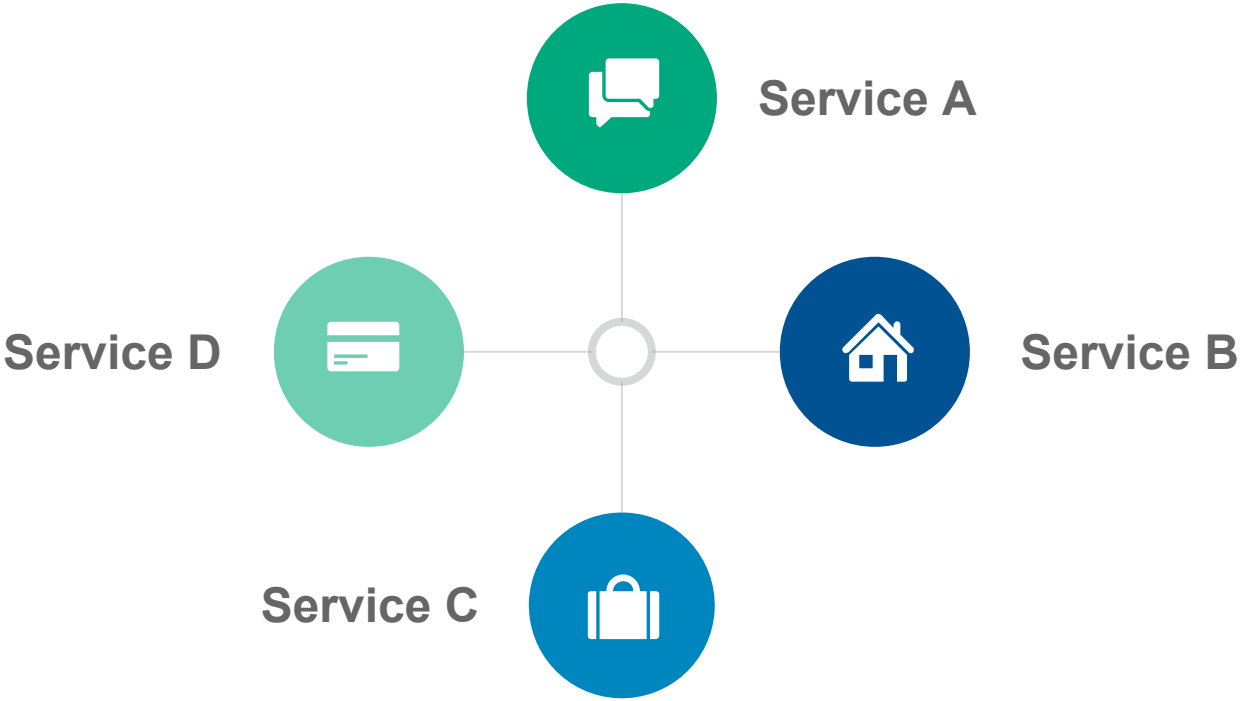
BlaBlaCar

pawel.rekowski@blablacar.com

Over 45 million members and growing

Komunikacja między serwisami

Przykładowe wyzwanie



Komunikacja przy użyciu REST API

Request

Resource: Trips

Action: Search

GET /api/v2/trips

Parameters | Headers | Body

fn: Warszawa	tn: Poznań
locale: pl_PL	_format: json
cur: PLN	fc: 48.756|7.268
tc: 48.756|7.268	db: 2017-09-02
de: 2017-09-02	hb: 7
he: 14	page: 1
seats: 1	photo:
fields: user,trips	sort: trip_price

Response

Status: 200 OK Time: 426 ms

JSON | Raw | Headers


```
{
  "links": {
    "_self": "https://api.blablacar.pl/api/v2/trips?_locale=pl_
    "_front": "https://www.blablacar.pl/search?fn=Warszawa&tn=P
  },
  "pager": {
    "page": 1,
    "pages": 1,
    "total": 5,
    "limit": 10
  },
  "trips": [
    {
      "links": {
        "_self": "https://api.blablacar.pl/api/v2/trips/8748738
        "_front": "https://www.blablacar.pl/podroz-warszawa-poz
        "_threads": "https://api.blablacar.pl/api/v2/trips/8748
      },
      "departure_date": "02/09/2017 08:00:00",
      "departure_place": {
        "city_name": "Warszawa",
        "address": "Plac Wilsona, Warszawa, Polska",
        "latitude": 52.2695,
        "longitude": 20.9854,
        "country_code": "PL"
      },
      "arrival_place": {
        "city_name": "Poznań",
        "address": "Poznań, Polska",
        "latitude": 52.4064,
        "longitude": 16.9252,
        "country_code": "PL"
      },
      "price": {
        "value": 60,
        "currency": "PLN",
        "symbol": "zł",
        "string_value": "60 zł",
        "price_color": "BLACK"
      },
```


https://dev.blablacar.com/

JSON jak Tablica

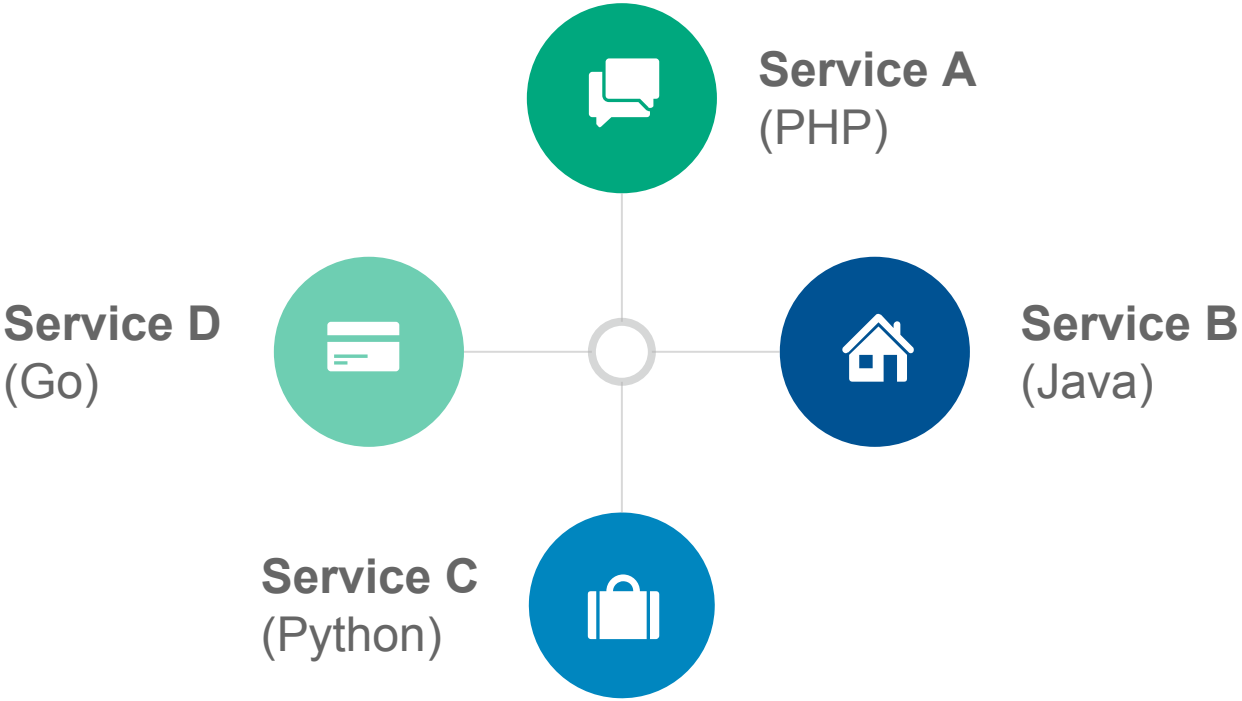
```
$array = [
    "city_name" => "Warszawa",
    "address" => "Plac Wilsona, Warszawa, Polska",
    "latitude" => 52.2695,
    "longitude" => 20.9854,
    "country_code" => "PL",

    ...

    "need_more" => "add_new_row",
];
```

Może lepiej ValueObject?

```php
final class Object {
    private
        $city,
        $address,
        $latitude,
        $longitude;

    public function __construct($city, $address, $latitude, $longitude)
    {
        $this->city = $city;
        $this->address = $address;
        $this->latitude = $latitude;
        $this->longitude = $longitude;
    }

    public function getCity() {
        return $this->city;
    }

    public function getAddress() {
        return $this->address;
    }

    public function getLatitude() {
        return $this->latitude;
    }

    public function getLongitude() {
        return $this->longitude;
    }
}
```

NEED MORE?

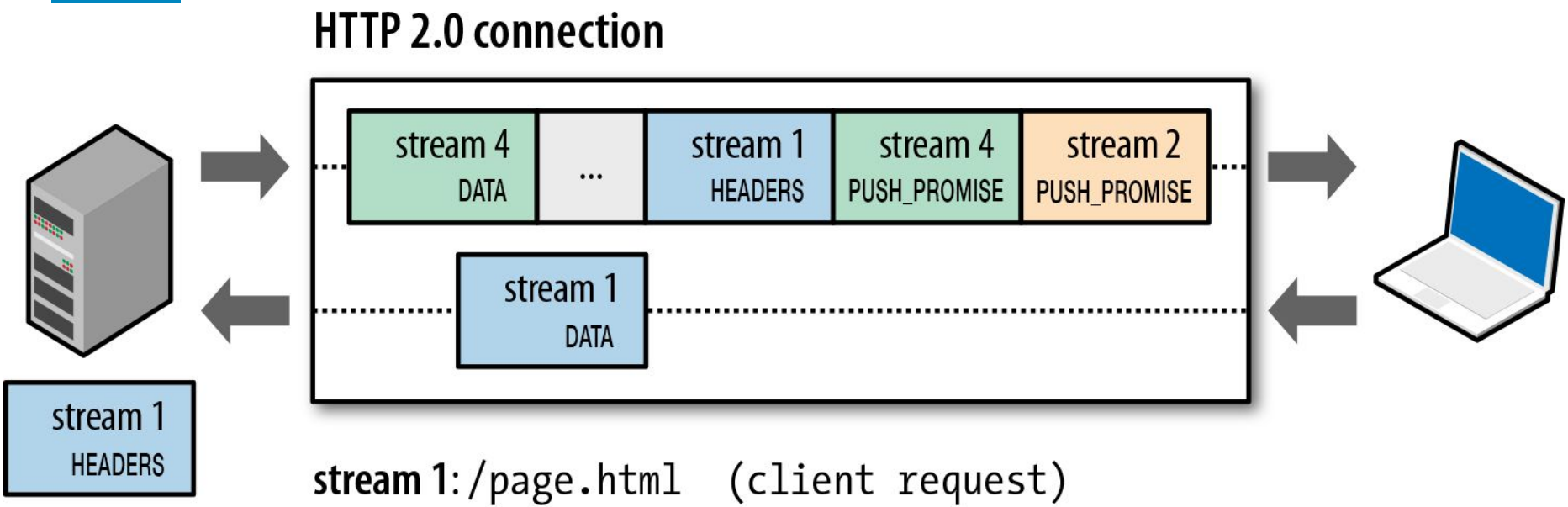
Multi Language Challenge



Kontrakt? Jaki Kontrakt?

HTTP/1.1 do kilku milionów Requestów

HTTP/2.0 do kilku miliardów Requestów



stream 1: /page.html (client request)
stream 2: /script.js (push promise)
stream 4: /style.css (push promise)

Źródło: https://www.oreilly.com/learning/http2-a-new-excerpt

Czy REST Ci wystarczy?

Czy REST Ci wystarczy?
Jeśli nie to co innego?

{ REST } GOODBYE

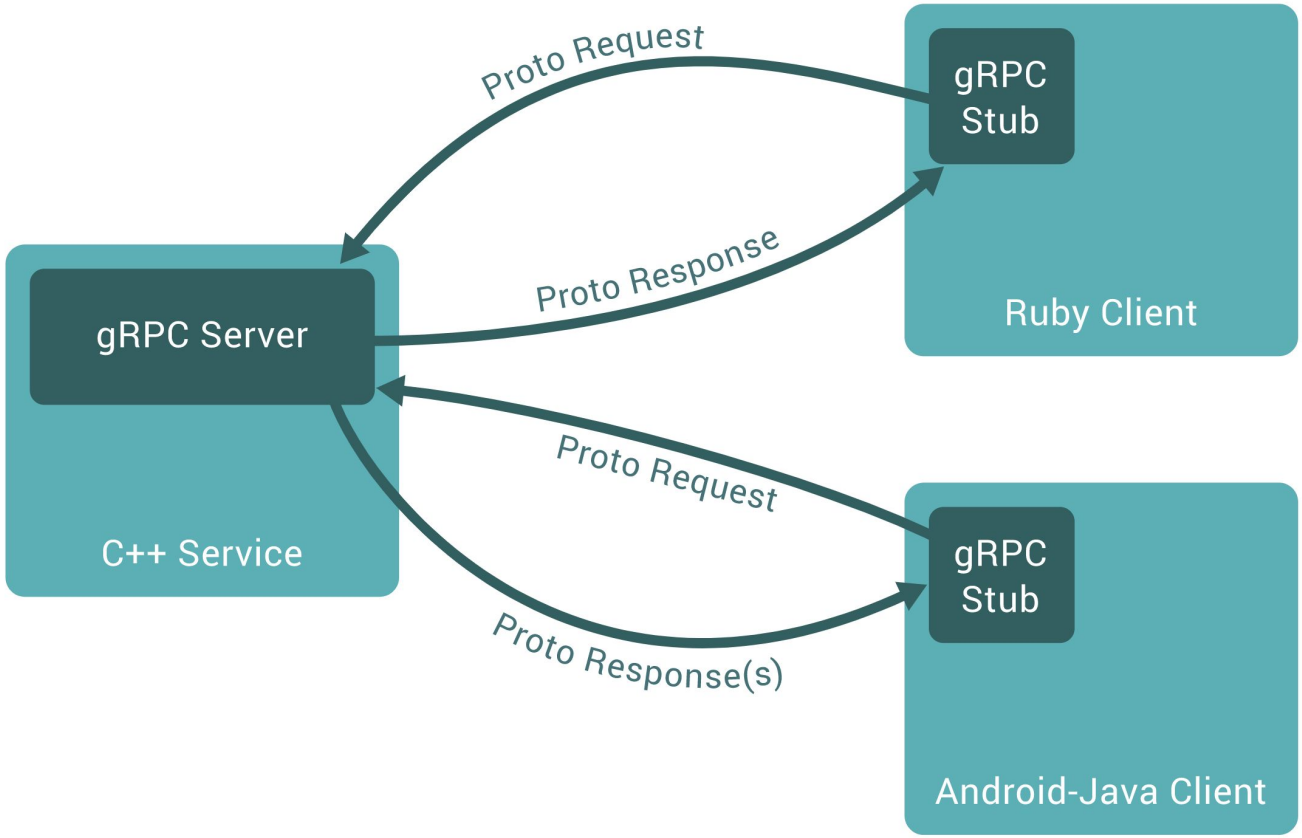
WELCOME GRPC

Co to jest GRPC?

A high performance, open-source universal RPC framework

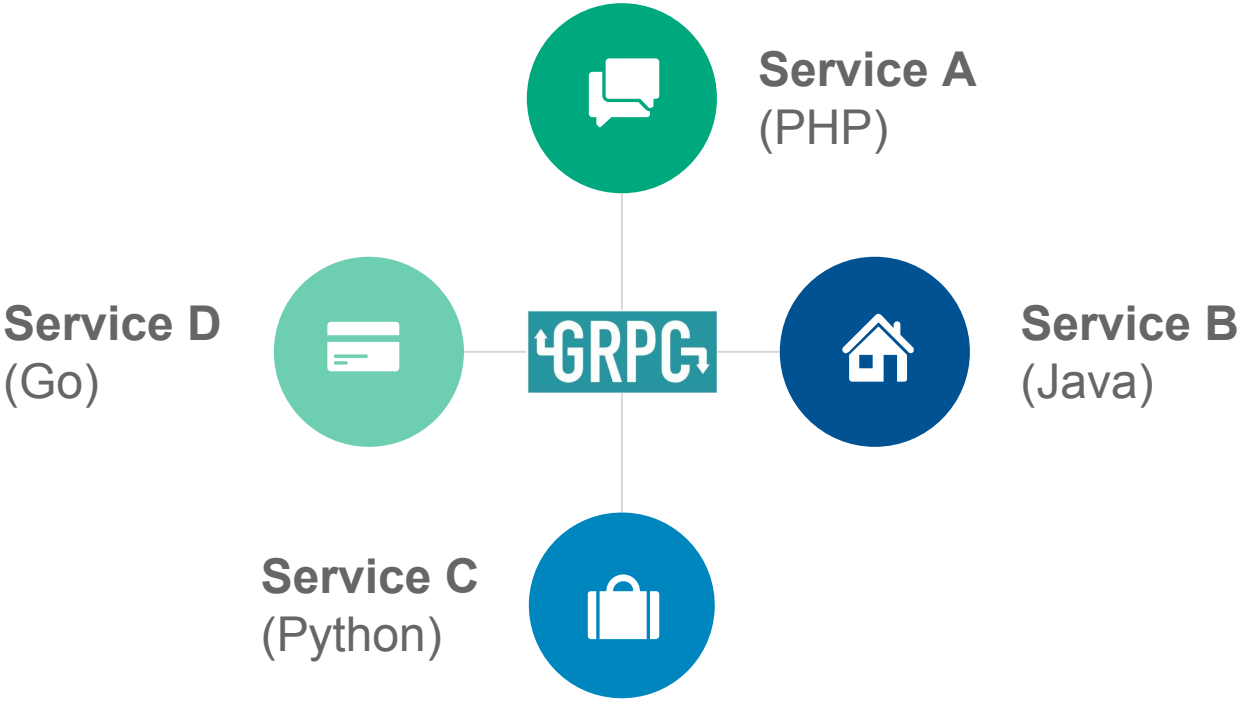
Źródło: https://grpc.io/

I co w tym takiego fajnego?



Źródło: https://grpc.io/

I co w tym takiego fajnego?



Kontrakt i komunikacja

protobuf
Protocol Buffers

```
syntax = "proto3";

package Grpc.Project;

message Person {
  string name = 1;
  int32 id = 2;
  string email = 3;
}
```

Kontrakt i komunikacja

Project

- prezentacja-grpc ~/Workspac
  - target
    - GPBMetadata
      - User.php
    - Grpc
      - Project
        - Person.php
  - user.proto
- External Libraries

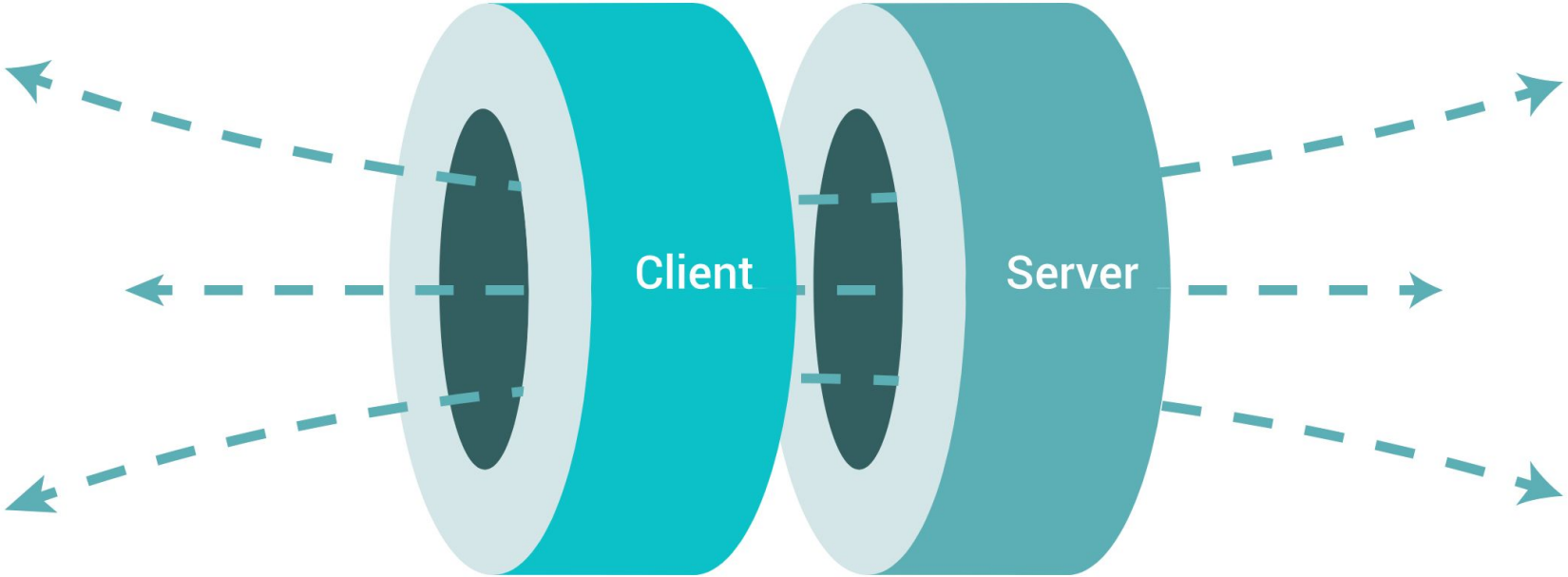
user.proto | Person.php

```php
<?php
# Generated by the protocol buffer compiler.  DO NOT EDIT!
# source: user.proto

namespace Grpc\Project;

use Google\Protobuf\Internal\GPBType;
use Google\Protobuf\Internal\RepeatedField;
use Google\Protobuf\Internal\GPBUtil;

/**
 * Generated from protobuf message <code>Grpc.Project.Person</code>
 */
class Person extends \Google\Protobuf\Internal\Message
{
    /**
     * Generated from protobuf field <code>string name = 1;</code>
     */
    private $name = '';
    /**
     * Generated from protobuf field <code>int32 id = 2;</code>
     */
    private $id = 0;
    /**
     * Generated from protobuf field <code>string email = 3;</code>
     */
    private $email = '';

    public function __construct() {
        \GPBMetadata\User::initOnce();
        parent::__construct();
    }
```

Kontrakt i komunikacja

```
4   package Grpc.Project;
5
6   public final class User {
7     private User() {}
8     public static void registerAllExtensions(
9         com.google.protobuf.ExtensionRegistryLite registry) {
10    }
11
12    public static void registerAllExtensions(
13        com.google.protobuf.ExtensionRegistry registry) {
14      registerAllExtensions(
15          (com.google.protobuf.ExtensionRegistryLite) registry);
16    }
17    public interface PersonOrBuilder extends
18        // @@protoc_insertion_point(interface_extends:Grpc.Project.Person)
19        com.google.protobuf.MessageOrBuilder {
20
21      /**
22       * <code>string name = 1;</code>
23       */
24      java.lang.String getName();
25      /**
26       * <code>string name = 1;</code>
27       */
```

I co w tym takiego fajnego?



Źródło: https://grpc.io/

Jak zacząć?

PHP Version 7.1.7

grpc

grpc support	enabled

```
➜  ~ brew install php71-grpc
➜  ~ php -m | grep grpc
grpc
```

GitHub, Inc. [US] https://github.com/google/protobuf/tree/master/php

This repository Search Pull requests Issues Marketplace Explore

google / protobuf

Watch 1,533 Star 19,712 Fork 5,863

Code Issues 463 Pull requests 177 Projects 0 Wiki Insights

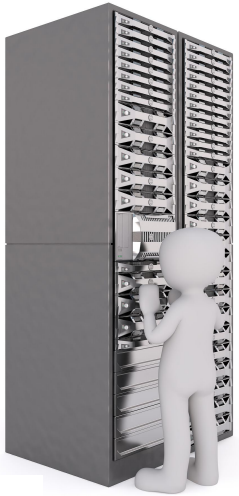
Branch: master protobuf / php / Create new file Upload files Find file History

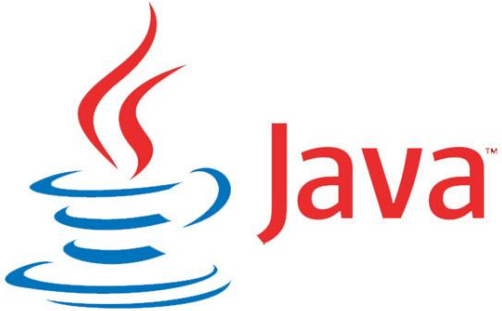
TeBoring committed on GitHub Add php support for Timestamp. (#3575) ... Latest commit b70e0fd 14 minutes ago

..

ext/google/protobuf	Add php support for Timestamp. (#3575)	14 minutes ago
src	Add php support for Timestamp. (#3575)	14 minutes ago
tests	Add php support for Timestamp. (#3575)	14 minutes ago
README.md	Merge 3.2.x branch into master (#2648)	7 months ago
composer.json	Update PHP descriptors (#3391)	27 days ago
generate_descriptor_protos.sh	Add file option php_class_prefix (#2849)	6 months ago
phpunit.xml	Update PHP descriptors (#3391)	27 days ago

C++





```
syntax = "proto3";

package Proto.Api;

import "common.proto";

service UserService {
    rpc GetUser(GetUserRequest) returns (GetUserResponse) {}
}

message GetUserRequest {
    bytes uuid = 1;
}

message GetUserResponse {
    User user = 1;
    Status = 2;
}

enum Status {
    STATUS_UNKNOWN = 0;
    OK = 1;
}
```

Generujemy klasy PHP

```
$ protoc \
    -I=./common/proto \
    --php_out=${PATH_GENERATED}
    --grpc_out=${PATH_GENERATED}
    --plugin=protoc-gen-grpc=/grpc_php_plugin ${PROTO_FILES}
```

```php
use Grpc\ChannelCredentials;

class Client
{
    private $userService;

    public function __construct($endpoint, $userAgent)
    {
        $credentials = ChannelCredentials::createInsecure();

        $this->userService = new UserServiceClient($endpoint, [
            'credentials' => $credentials,
            'grpc.primary_user_agent' => $userAgent,
        ]);
    }

    public function getUser(GetUserRequest $request)
    {
        list($response, $status) = $this->userService->GetUser($request)->wait();
        $this->handleStatus($status);

        return $response;
    }

    private function handleStatus($status)
    {
        switch ($status->code) {
            case 14:
                throw new ServerUnavailableException();
            case 5:
                throw new UserNotFoundException();
            default:
                break;
        }
    }
```

Tworzymy Klienta

```php
<?php

    use Proto\Api\Client;
    use Proto\Api\GetUserRequest;
    use Ramsey\Uuid\Uuid;

    include_once __DIR__.'/../vendor/autoload.php';

    $client = new Client( endpoint: "localhost:6565",  userAgent: 'user-client');

    $userId = Uuid::fromString( name: 'f2994de0-4b93-11e7-b31b-a23eb0749c31');

    $response = $client->getUser(new GetUserRequest($userId));

    var_dump($response);
```

Wyzwania dnia codziennego

Breaking names - UNKNOWN

```
enum Country {
  UNKNOWN = 0;
  PL = 1;
  // ...
}
```

```
enum Locale {
  UNKNOWN = 0;
  PL_PL = 1;
  // ...
}
```

```
enum Gender {
  UNKNOWN = 0;
  MAN = 1;
  WOMAN = 2;
}
```

Breaking names - UNKNOWN

```
enum Country {
  UNKNOWN = 0;
  PL = 1;
  // ...
}

enum Locale {
  UNKNOWN = 0;
  PL_PL = 1;
  // ...
}

enum Gender {
  UNKNOWN = 0;
  MAN = 1;
  WOMAN = 2;
}
```

Breaking names - UNKNOWN

```
enum Country {
  UNKNOWN = 0;
  PL = 1;
  // ...
}

enum Locale {
  UNKNOWN = 0;
  PL_PL = 1;
  // ...
}

enum Gender {
  UNKNOWN = 0;
  MAN = 1;
  WOMAN = 2;
}
```

```
enum Country {
  COUNTRY_UNKNOWN = 0;
  PL = 1;
  // ...
}

enum Locale {
  LOCALE_UNKNOWN = 0;
  PL_PL = 1;
  // ...
}

enum Gender {
  GENDER_UNKNOWN = 0;
  MAN = 1;
  WOMAN = 2;
}
```

UUID

32e19097-550a-11e7-b77a-5c0855b93c21

UUID między językami

```
message UUID {
  int64 most = 1;
  int64 least = 2;
}
```

VS

```
message UUID {
  bytes value = 1;
}
```

Quiz

@TODO

Dziękuje!