AMEX People REST Service Documentation

Table of Contents

[Summary 1](#_Toc530406423)

[API Endpoints 1](#_Toc530406424)

[Get All People 1](#_Toc530406425)

[Add People 2](#_Toc530406426)

[Get People by ID 2](#_Toc530406427)

[Update People 2](#_Toc530406428)

[Delete People 3](#_Toc530406429)

[Error Responses 3](#_Toc530406430)

[Exception Reponses Example 3](#_Toc530406431)

[Status Codes 4](#_Toc530406432)

[Attributes 4](#_Toc530406433)

# Summary

This document outlines the Java REST web service that provides CRUD-style API access to data stored in a relational database system.

# API Endpoints

## Get All People

* **Path**: /people
* **Method:** GET
* **Accepted Content-Type:** NA
* **Request Example:** Not Required
* **Response Example:** 200 - OK

[

{

"id": 1,

"name": "John",

"age": 2,

"birthdate": "2016-01-06",

"email": "john@yahoo.com"

},

{

"id": 2,

"name": "jay",

"age": 63,

"birthdate": "1955-01-06",

"email": "jay@gmail.com"

}

]

## Add People

* **Path:** /people
* **Method:** POST
* **Accepted Content-Type:** application/json
* **Request Example:**

{

"name": "jay",

"birthdate": "1955-01-06",

"email": "jay@gmail.com"

}

* **Response Example:** 201 - Created

{

"id": 2,

"name": "jay",

"age": 63,

"birthdate": "1955-01-06",

"email": "jay@gmail.com"

}

## Get People by ID

* **Path:** /people/{:id}
* **Method:** GET
* **Accepted Content-Type:** NA
* **Request Example:** Not Required
* **Response Example:** 200 - OK

{

"id": 1,

"name": "John",

"age": 2,

"birthdate": "2016-01-06",

"email": "john@yahoo.com"

}

## Update People

* **Path:** /people/{:id}
* **Method:** PUT
* **Accepted Content-Type:** application/json
* **Request Example:**

{

"name": "John",

"birthdate": "2011-11-30",

"email": "John@gmail.com"

}

* **Response Example:** 200 OK

{

"id": 1,

"name": "John",

"age": 6,

"birthdate": "2011-11-30",

"email": "john@gmail.com"

}

## Delete People

* **Path:** /people/{:id}
* **Method:** DELETE
* **Accepted Content-Type:** NA
* **Request Example:** No request required
* **Response Example:** 204 No Content

No response body return for success case

# Error Responses

## Exception Reponses Example

* Exceptions will return the following JSON Response (example):

{

"timestamp": 1542656458178,

"status": "422",

"error": "Unprocessable Entity",

"exception": "",

"message": "Requested Data Not Found"

}

{

"timestamp": 1542656552259,

"status": "422",

"error": "Unprocessable Entity",

"exception": "org.springframework.dao.EmptyResultDataAccessException",

"message": "No class com.javian.amex.people.domain.People entity with id 1 exists!"

}

{

"timestamp": 1542659372382,

"status": 400,

"error": "Bad Request",

"exception": "org.springframework.http.converter.HttpMessageNotReadableException",

"message": "JSON parse error: Unexpected character (',' (code 44)): expected a value; nested exception is com.fasterxml.jackson.core.JsonParseException: Unexpected character (',' (code 44)): expected a value\n at [Source: java.io.PushbackInputStream@103399cd; line: 3, column: 16]",

"path": "/people"

}

## Status Codes

* 400 – Bad Request -> Returned when the client has made a bad requested. This can mean missing or badly formed arguments.
* 404 – Resource Not Found -> Returned when the client has requested a resource that doesn’t exist. For example, asking for a user before the user has been created.
* 422 – Unprocessable Entity -> Returned when the client has made an invalid request. This can due to null or malformed attributes.
* 500 – Internal Server Error

# Attributes

|  |  |  |
| --- | --- | --- |
| Attribute Key Name | Data Type | Comment |
| id | Integer |  |
| name | String |  |
| age | Integer | Age will be calculate base on birthdate attribute |
| birthdate | String | Validation will be against below format  *YYYY-MM-DD* |
| email | String | Validation will be against email format |