



Claim Your Free DZone Job Seeker Profile, Take Your Career t...

[Claim Now▶](#)

# Developing RESTful APIs With Loopback

by Prosper Otemuyiwa MVB · Sep. 08, 17 · Web Dev Zone

Prove impact and reduce risk when rolling out new features. Optimizely Full Stack helps you experiment in any application.

---

Note: Check out the repo to get the code.

## RESTful API Overview

API (Application Programming Interface) endpoints are the connections between your application and the rest of the developer community. You can decide to open up your data source to the world by crafting API endpoints that developers can consume. Furthermore, developing an application that will require clients for different platforms such as Desktop, Android, iOS and Windows platform will most likely need a RESTful API for all the clients to access data seamlessly. Unless of course, you are engaging GraphQL, the alternate option to RESTful APIs.

If you have built a RESTful API from scratch before now, you'll understand that there are a lot of processes you have to follow, a lot of code to write, and a lot of testing to do to ensure your APIs work correctly. Loopback was designed to fast-track those processes and take away the pain of developing RESTful APIs from scratch.

## What Is Loopback

**Loopback** is a highly extensible open-source Node.js framework that can be used to build dynamic end-to-end REST APIs. With little or no code, Loopback hands you the power to:

- Quickly create APIs.
- Connect your APIs to data sources such as relational databases, MongoDB, REST APIs, etc.
- Incorporate model relationships and access controls for complex APIs.
- Extend your APIs.

**Loopback** runs on-premises or in the cloud. It also provides several add-on components for file management, 3rd-party login, OAuth2, Storage providers, etc. Simply put, Loopback allows you to create REST APIs in minutes. It does that by:

- Providing a CLI tool for use.
- Providing a built-in API explorer.
- Allowing a developer to create models based on a particular schema.
- Allowing a developer to create dynamic models in the absence of a schema.
- Providing SDKs for iOS, Android, and AngularJS to easily create client apps.

## Application Overview

In this tutorial, we'll build a RESTful API for Star Wars. A Star Wars API, **SWAPI** already exists. Rather than crafting an entirely different API, we'll build a clone of the Star Wars API. The objective of building the API clone is to learn how to craft APIs quickly with Loopback. Consider a case where you are assigned the task of designing an API for a new interesting HBO show, e.g Game of Thrones, that has caught the attention of Google and Facebook.

## Explore SWAPI (Star Wars API) Design

We will explore the API endpoints for SWAPI(Star Wars API). There are six endpoints for the Star Wars API:

- Planets - `/planets`
- People - `/people`
- Starships - `/starships`
- Species - `/species`

- **Vehicles** - `/vehicles`
- **Films** - `/films`
- **Planets** - This resource is about the planetary bodies in the Star Wars Universe.
  - GET - `/planets` returns a list of all the planets.
  - GET - `/planets/<id>` returns the details of a specific planet.
- **People** - This resource deals with characters in the Star Wars Universe.
  - GET - `/people` returns a list of all the characters.
  - GET - `/planets/<id>` returns the details of a specific character.
- **Starships** - This resource is about the transport crafts that have hyperdrive capability in the Star Wars Universe.
  - GET - `/starships` returns a list of all the starships.
  - GET - `/starships/<id>` returns the details of a specific starship.
- **Vehicles** - This resource is about the transport crafts that do not have hyperdrive capability.
  - GET - `/vehicles` returns a list of all the vehicles.
  - GET - `/vehicles/<id>` returns the details of the vehicle.
- **Species** - This resource is about the type of characters in the Star Wars Universe.
  - GET - `/species` returns a list of all the species.
  - GET - `/species/<id>` returns the details of a specific species.
- **Films** - This resource is about the episodes of the Star Wars film.
  - GET - `/films` returns a list of all the episodes.
  - GET - `/films/<id>` returns the details of a specific film.

In the analysis above, you'll observe that all the calls for

In the analysis above, you can observe that all the calls for the endpoints are GET requests. We'll ensure that the other HTTP verbs, POST, PUT, and DELETE will be active for these endpoints when building the API.

## Getting Started: Building Star Wars API

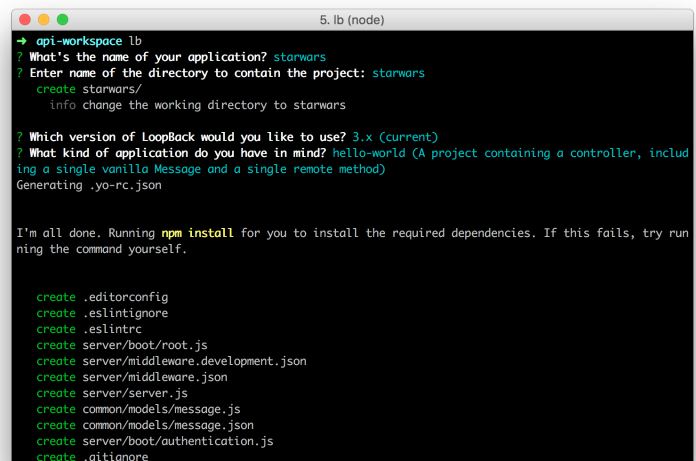
Let's get started. Go ahead and install Loopback:

```
1 npm install -g loopback-cli
```

Once you are done installing, go ahead and run the loopback command:

```
1 lb
```

The Loopback CLI is an interactive wizard. A series of questions will be asked such as the name of the application, directory to store the project, the version of loopback you prefer, and the kind of application you want loopback to provision. Answer the questions:



```
5. lb (node)
api-workspace lb
? What's the name of your application? starwars
? Enter name of the directory to contain the project: starwars
  create starwars/
    info change the working directory to starwars

? Which version of LoopBack would you like to use? 3.x (current)
? What kind of application do you have in mind? hello-world (A project containing a controller, including a single vanilla Message and a single remote method)
Generating .yo-rc.json

I'm all done. Running npm install for you to install the required dependencies. If this fails, try running the command yourself.

create .editorconfig
create .eslintrc
create .eslintrc
create server/boot/root.js
create server/middleware.development.json
create server/middleware.json
create server/server.js
create common/models/message.js
create common/models/message.json
create server/boot/authentication.js
create .gitignore
```

Next, move into the `starwars` directory from the terminal.

## Create Models

We'll create all the models needed for this API. As mentioned earlier, we have 6 resources, People, Films, Starships, Species, Vehicles and Planets. In our API, we'll deal with just 5.

```
1 lb model
```

It will prompt you for a model name: enter `People`. Next, select `db(memory)` as the data source. Go ahead and select `Person` as the model's base class. Hit `Enter` to

`persistedModel` as the `models` base class. Hit enter to expose `People` via the REST API.

Next, select `common` model. Selecting this ensures that the models are stored in the `common` directory. You can also decide to store them in the `server` directory only. The `server` directory is used to store server-side models while the `common` directory stores models that can potentially be used by both server and client Loopback APIs.

Now, you are going to define properties for the `People` model. The properties are:

- Property name: `name`, Property type: `string`, Required? `Yes`
- Property name: `height`, Property type: `number`, Required? `Yes`
- Property name: `mass`, Property type: `number`, Required? `Yes`
- Property name: `gender`, Property type: `string`, Required? `Yes`

Repeat the same process for the `Films`, `Starships`, `Species` and `Planets` models with these properties:

**Note:** You'll have to run `lb model` on your terminal everytime you need to create a new model.

## Film

- Property name: `title`, Property type: `string`, Required? `Yes`
- Property name: `opening_crawl`, Property type: `string`, Required? `Yes`
- Property name: `director`, Property type: `string`, Required? `Yes`
- Property name: `producer`, Property type: `string`, Required? `Yes`

## Starship

- Property name: `name`, Property type: `string`, Required? `Yes`
- Property name: `model`, Property type: `string`, Required? `Yes`
- Property name: `manufacturer`, Property type: `string`, Required? `Yes`
- Property name: `passengers`, Property type: `number`, Required? `Yes`

- Property name: passengers, Property type: number, Required? Yes
- Property name: class, Property type: string, Required? Yes

## Species

- Property name: name, Property type: string, Required? Yes
- Property name: classification, Property type: string, Required? Yes
- Property name: designation, Property type: string, Required? Yes
- Property name: average\_height, Property type: number, Required? Yes
- Property name: skin\_color, Property type: string, Required? Yes
- Property name: language, Property type: string, Required? Yes

## Planet

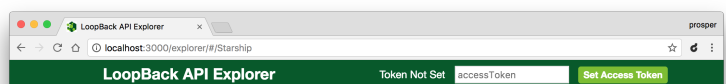
- Property name: name, Property type: string, Required? Yes
- Property name: rotation\_period, Property type: number, Required? Yes
- Property name: orbital\_period, Property type: number, Required? Yes
- Property name: diameter, Property type: number, Required? Yes
- Property name: population, Property type: number, Required? Yes

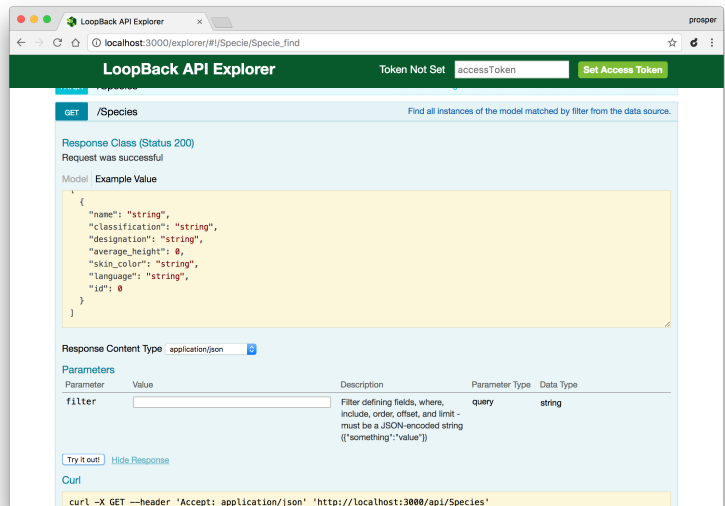
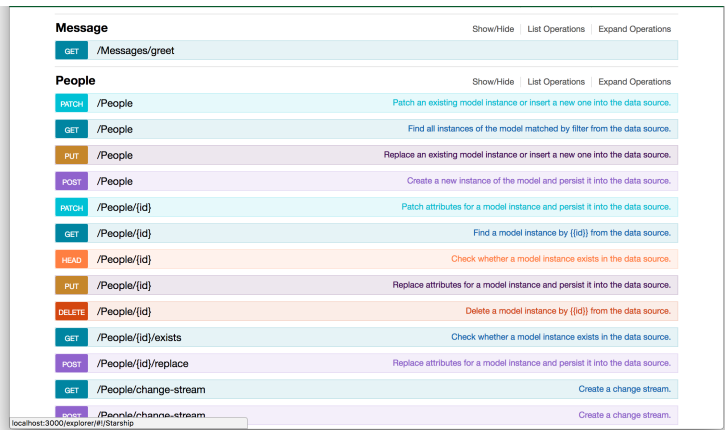
Once you are done provisioning all the models, head over to the `common/models` directory. In this directory, you'll see all the models in `.js` and `.json` files. For example, `people.js` and `people.json` file.

Next, test the API. Run the command below in the root path of the project:

```
1 node .
```

Check out your API explorer.

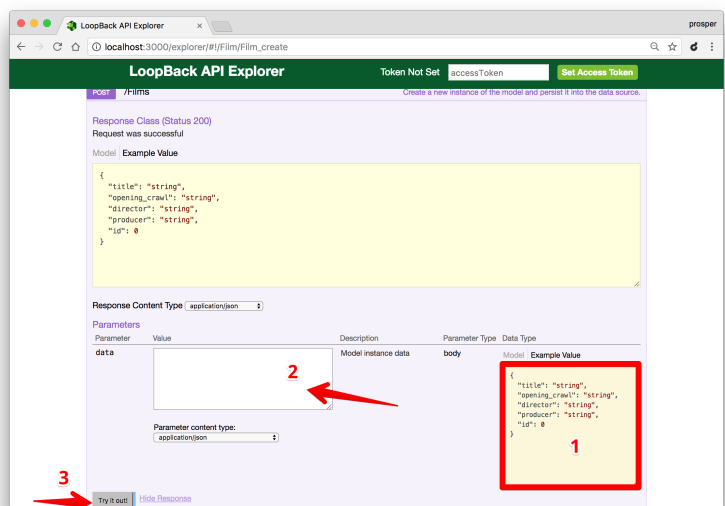




Here, you can see the structure of the Species resource. Clicking the **Try it out** button hits the `/api/species` endpoint and returns a result. Right now, there is no data so it returns an empty array.

**Note:** The API explorer allows you test the REST API operations during development.

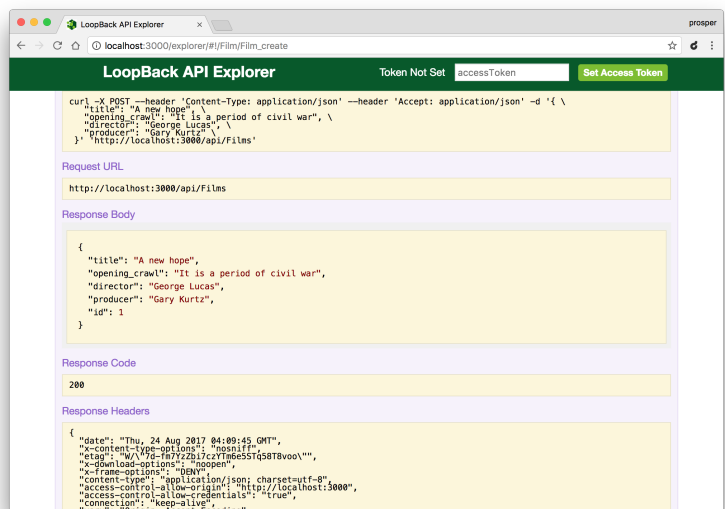
Let's make a POST request to the `/api/film` endpoint using the explorer. Head over to [http://localhost:3000/explorer/#!/Films/Films\\_create](http://localhost:3000/explorer/#!/Films/Films_create).



In the diagram above, I highlighted three steps:

1. Click on the text-area box that contains the parameter type. Once clicked, it immediately appears in box 2.
2. Edit the JSON in the box.
3. Click the `Try it out` button to execute the POST operation.

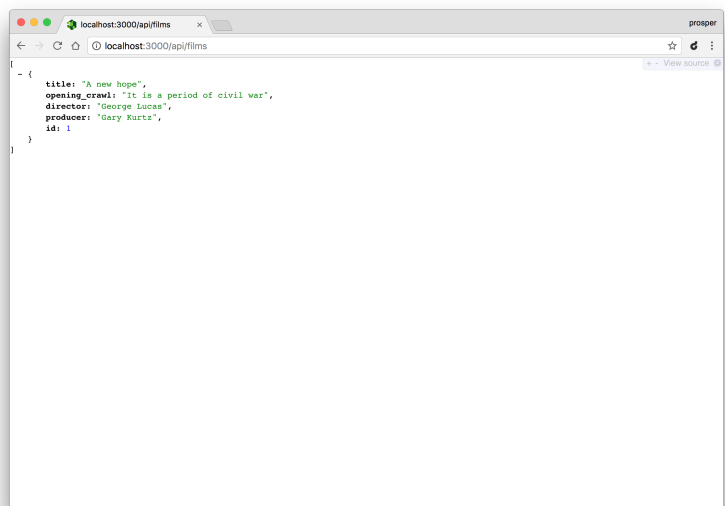
**Note:** If you are making a POST request to create a new record, remove the `id` attribute and value from the JSON in the box. `id` is automatically assigned.



### *Result of POST operation*

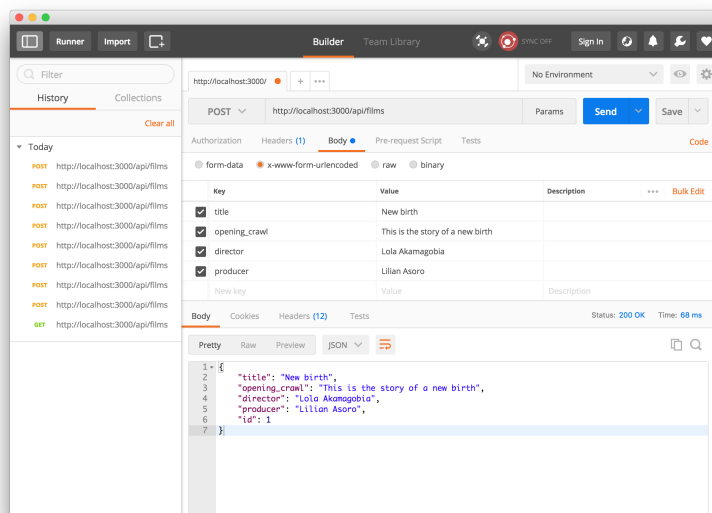
Now check out the URL,

<http://localhost:3000/api/films>.



Try making a POST request to the films endpoint with Postman.





Voila! Be careful, our API is not connected to any data source yet.

Next, let's connect our API to a data source.

## Connect API to Data Source

Let's connect our API to a database. Luckily for us, Loopback supports a lot of data sources. Head over to your terminal and run the following command:

```
1 lb datasource
```

It will prompt you for a name. Enter `mysqlDs` .

`mysqlDs` is MySQL Datasource for short. Yes, we'll make use of MySQL. So, make sure you have MySQL database installed on your machine.

Next, select `mysql` as the connector for `mysqlDs` and hit Enter. Go ahead and supply the `host` , `port` , `user` and `password` values for the connection string. And enter `Yes` to install the **loopback-connector-mysql** tool.

The connection details can be found at `server/datasources.json` . Feel free to edit the file anytime.

## Connect API to MySQL

Open up `server/model-config.json` and change the value of the `dataSource` property from `db` to `mysqlDs` for each of the models.

*server/model-config.json*

```
1  ...
2  "People": {
3    "dataSource": "db",
4    "public": true
5  },
6  "Film": {
7    "dataSource": "db",
8    "public": true
9  },
10 "Starship": {
11   "dataSource": "db",
12   "public": true
13 },
14 "Specie": {
15   "dataSource": "db",
16   "public": true
17 },
18 "Planet": {
19   "dataSource": "db",
20   "public": true
21 }
```

**After changing to `mysqlDs`server/`model-config.json`**

```
1  ...
2  "People": {
3    "dataSource": "mysqlDs",
4    "public": true
5  },
6  "Film": {
7    "dataSource": "mysqlDs",
8    "public": true
9  },
10 "Starship": {
11   "dataSource": "mysqlDs",
12   "public": true
13 },
14 "Specie": {
15   "dataSource": "mysqlDs",
16   "public": true
17 },
18 "Planet": {
19   "dataSource": "mysqlDs",
20   "public": true
21 }
```

## Create Model Tables in

MySQL

We'll go ahead and create tables for each of the models in MySQL. There are two approaches:

- Let's go with the second option. It's more effective. And we need test data. We'll write an automigration script to programmatically create the tables.

```

1 module.exports = function(app) {
2     app.dataSources.mysqlDs.automigrate('People', function(err) {
3         if (err) throw err;
4
5         app.models.People.create([
6             {
7                 name: 'Luke Skywalker',
8                 height: 172,
9                 mass: 77,
10                gender: 'Male'
11            }, {
12                name: 'C-3PO',
13                height: 167,
14                mass: 75,
15                gender: 'Undetermined'
16            }], function(err, People) {
17                if (err) throw err;
18
19                console.log('Models created: \n', People);
20            });
21        });
22
23    app.dataSources.mysqlDs.automigrate('Film', function(err) {
24        if (err) throw err;
25
26        app.models.Film.create([
27            {
28                title: 'A New Hope',
29                opening_crawl: 'It is a period of civil war'
30            }, {
31                title: 'The Empire Strikes Back',

```

```
--
    opening_crawl: 'It is a dark time for the re
32  ◀────────────────────────────────────────▶
    director: 'Irvin Kershner',
33
    producer: 'Rick McCallum'
34
    }], function(err, People) {
35
    if (err) throw err;
36
37
    console.log('Models created: \n', People);
38  ◀────────────────────────────────────────▶
    });
39
    });
40
41
    app.dataSources.mysqlDs.automigrate('Starship',
42  ◀────────────────────────────────────────▶
    if (err) throw err;
43
44
    app.models.Starship.create([
45
    name: 'Death Star',
46
    model: 'DS-1 Orbital Battle Station',
47
    manufacturer: 'Imperial Department of Milita
48  ◀────────────────────────────────────────▶
    passengers: 843342,
    class: 'Deep Space Mobile Battlestation'
49
50  ◀────────────────────────────────────────▶
    }, {
51
    name: 'Sentinel-class landing craft',
52
    model: 'Sentinel-class landing craft',
53
    manufacturer: 'Sienar Fleet Systems, Cyngus
54  ◀────────────────────────────────────────▶
    passengers: 75,
    class: 'Landing Craft'
55
    }], function(err, Starship) {
56
    if (err) throw err;
57
58
    console.log('Models created: \n', Starship);
59
60  ◀────────────────────────────────────────▶
    });
61
    });
62
63
    app.dataSources.mysqlDs.automigrate('Specie', fu
64  ◀────────────────────────────────────────▶
    if (err) throw err;
65
66
    app.models.Specie.create([
67
    name: 'Droid',
68
    classification: 'artificial',
69
    designation: 'sentient',
70
    average_height: 34,
71
    skin_color: "brown",
72
```

```

73     language: "Galacticus"
74   }, {
75     name: 'Human',
76     classification: 'Mammal',
77     designation: 'sentient',
78     average_height: 180,
79     skin_color: 'black',
80     language: 'Galactic Basic'
81   }], function(err, Specie) {
82     if (err) throw err;
83
84     console.log('Models created: \n', Specie);
85   });
86 });
87
88 app.dataSources.mysqlDs.automigrate('Planet', fu
89   if (err) throw err;
90
91   app.models.Planet.create([
92     {
93       name: 'Yavin IV',
94       rotation_period: 24,
95       orbital_period: 4818,
96       diameter: 10200,
97       population: 1000
98     }, {
99       name: 'Hoth',
100      rotation_period: 23,
101      orbital_period: 549,
102      diameter: 7200,
103      population: 12500
104    }], function(err, Planet) {
105      if (err) throw err;
106
107      console.log('Models created: \n', Planet);
108    });
109  });
110
111  });
112  };

```

The bunch of code above simply creates the tables and seeds them with the data provided. Whenever your app is

initialized, the script will run and ensure the tables exist and are seeded with the right data.

Head over to your console and start your app again:

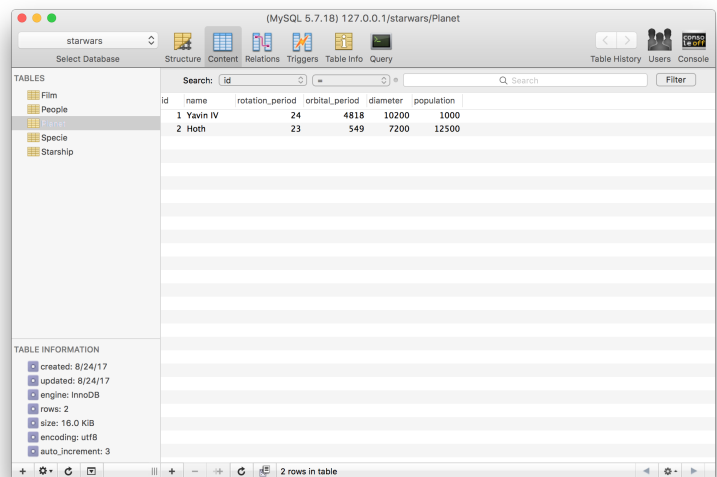
```

→ starwars node .
Web server listening at: http://localhost:3000
Browse your REST API at http://localhost:3000/explorer
Models created:
[ { title: 'A New Hope',
  opening_crawl: 'It is a period of civil war',
  director: 'George Lucas',
  producer: 'Gary Kurtz',
  id: 1 },
  { title: 'The Empire Strikes Back',
    opening_crawl: 'It is a dark time for the rebellion',
    director: 'Irvin Kershner',
    producer: 'Rick McCallum',
    id: 2 } ]
Models created:
[ { name: 'Luke Skywalker',
  height: 172,
  mass: 77,
  gender: 'Male',
  id: 1 },
  { name: 'C-3PO',

```

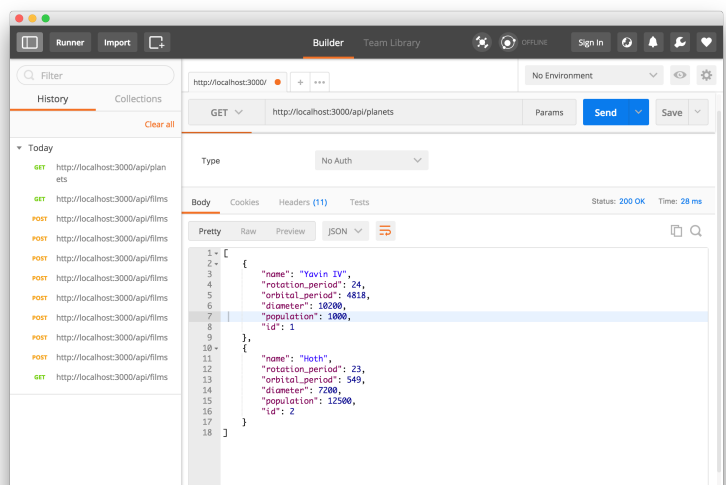
*Create Tables and seed data*

Check your database. The new tables and data should reflect there.



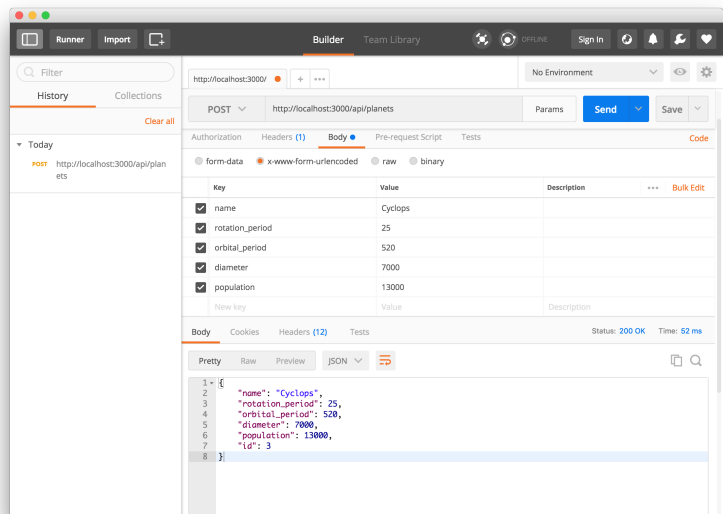
## Test the API

Now that our API is connected to MySQL. Let's test our API for persistence. Make a GET request to `http://localhost:3000/api/planets`.



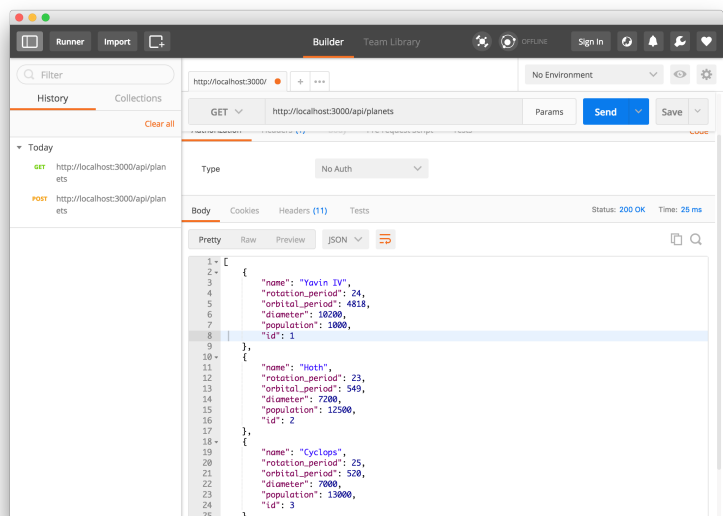
## Initial GET Request

Go ahead and make a `POST` request to the URL.



## POST Request

Now, make another `GET` request to the URL.



## Another GET Request

Yes, we see three records. Our data was persisted successfully. You can test out the `PUT`, `PATCH`, and `DELETE` operations for the `planets` and other API endpoints.

With SDKs for all major client and server side platforms, you can experiment on any platform with Optimizely Full Stack.

Published at DZone with permission of Prosper Otemuyiwa, DZone MVB. [See the original article here.](#)



Opinions expressed by DZone contributors are their own.

## Get the best of Web Dev in your inbox.

Stay updated with DZone's bi-weekly Web Dev Newsletter.

[SEE AN EXAMPLE](#)

[SUBSCRIBE](#)

## Web Dev Partner Resources

---

A Guide to Modern Java Web Development with Crafter CMS  
Crafter Software



Learn how to experiment server-side with Optimizely Full Stack.  
Watch a six-minute demo today.

Optimizely



See how top engineering teams are experimenting in any  
application. Watch a webinar on Optimizely Full Stack.

Optimizely



Getting Started with Node.js, Docker, and Kubernetes

NodeSource



## 10 Books to Boost Team Work During Software Development

by Andrew Stepanov · Sep 08, 17 · Agile Zone

See how three solutions work together to help your teams have the tools they need to deliver quality software quickly. Brought to you in partnership with CA Technologies.



***'Talent wins games, but teamwork and intelligence wins championships.' - Michael Jordan***

---

I will not surprise anyone – better results in any project (and software development is no exception here) are achieved thanks to teamwork. Communication, ideas, and thoughts sharing as well as outside points of view in combination with project management software open great perspectives for efficient and successful work on a project or anything else.

But to make a team out of a group of developers is not an easy task. Obstacles did, do and will occur. Sometimes those obstacles are expressed in team members' ego, different approaches to problem-solving, or a lack of motivation, and so on and so forth. That's why managing a group of developers (and people in general) implies the importance of teamwork.

To help you to create a better team environment and enhance its efficiency, I prepared a list of 10 books about teamwork that are relevant in 2017. By the way, you may combine this list with the best project management books for every manager.

## **1. The 7 Habits of Highly Effective People: Powerful Lessons in Personal Change**

**Author:** Stephen Covey.

**Year:** 1989

I could not help but include this book in my list. It was first published in 1989, and a few years ago its celebrated 25th edition, which says a lot about its fabulous popularity.

Stephen Covey was an educator, consultant, speaker, and management expert. In the book, he presents the approaches of being effective and the ways to achieve effectiveness. He underlines that before someone is going to include these approaches in their life, he or she has to

re-evaluate their whole system of views and ideas. Covey states that in order to change something, we should pay attention to our personal behavior and attitude.

What does it have to do with teamwork? Under habits 4, 5 and 6 the author discusses communication and collaboration.

### **What Is So Special About This Book?**

Actually, it is recommended by many people. And all reviews you'll find, regardless of the source, will say something like, 'this is one of the most inspiring books I have ever read.'

## **2. The 17 Indisputable Laws of Teamwork: Embrace Them and Empower Your Team**

**Author:** John C. Maxwell.

**Year:** 2000

In this book, the author elaborates on a theme of how a human being can become a more productive team member, a leader, or even an example to follow.

In fact, most of Maxwell's ideas and statements are based on common sense. It is a practical guide-book for those who want to build successful teams and get to know how to be a part of them.

### **What Is So Special About This Book?**

The expressed ideas are presented in understandable terms and language. Managers and leaders will find plenty of useful information in it.

## **3. The Five Dysfunctions of a Team: A Leadership Fable**

**Author:** Patrick Lencioni.

**Year:** 2000

This business-book by Patrick Lencioni, the famous consultant and speaker, was first published in 2000. In it, he describes all the difficulties and impediments that many teams face as well as the basic reasons for these

difficulties.

The book was included in the bestseller list by many leading media resources such as The New-York Times, The Wall-Street Journal, Bloomberg Business Week, and USA Today.

### **What Is So Special About This Book?**

The information is given in a light manner. After you are done with the book, you will look at your colleagues and team members in a different way. The described approaches can be applied not only in the business environment, but in other spheres as well.

## **4. Drive: The Surprising Truth About What Motivates Us**

**Author:** Daniel H. Pink.

**Year:** 2009

The author of this book claims that the carrot and the stick approach no longer works as a motivational technique. People are motivated by their personal desires to direct their lives, by striving to develop skills and bettering themselves. This is what drives them to show the best results and get pleasure from their work and study.

*Drive: The Surprising Truth About What Motivates Us* was included in New-York Times' bestseller list.

### **What Is So Special About This Book?**

The book is greatly focused on motivation. All the ideas and assertions within this book are justified by psychological research on people's behavior. The author also offers up ideas of creating your own motivational systems and tools.

## **5. Extraordinary Groups: How Ordinary Teams Achieve Amazing Results**

**Authors:** Geoffrey M. Bellman, Kathleen D. Ryan.

**Year:** 2009

Two experts elaborate upon a new approach that helps teams to gain extraordinary experience and succeed.

Sometimes people work in groups and describe their experience as wonderful. But, there are just as many cases where working within teams is described in a rather negative way. Why do some people have great teamwork experiences and others do not?

In the book, Bellman and Ryan state that extraordinary groups appear when two or more core values, subconsciously carried by all team members, are satisfied in teamwork.

### **What Is So Special About This Book?**

The information is well-presented and backed by extensive research.

## **6. Teaming: How Organizations Learn, Innovate, and Compete in the Knowledge Economy**

**Author:** Amy C. Edmondson.

**Year:** 2009

Amy Edmondson is a professor of leadership and management at Harvard Business School. In the book that was published in 2009, she acknowledges that people are different. But, regardless of all those differentiations, Edmondson welcomes and supports team collaboration. For that reason, she offers innovative ideas and argues for the importance of teamwork.

It should be noted that her ideas are not somehow intuitive, but pretty easy to understand and learn. There is no step-by-step ready-made plan in the book. Quite the opposite, you will find ideas and methodologies that will help to spark team spirit.

### **What Is So Special About This Book?**

This work is based on the author's many years of experience. She clearly defines what a team is and what it is not. The book sparks the desire to work in a team and not to dictate your terms.

## 7. Help the Helper: Building a Culture of Extreme Teamwork

**Authors:** Kevin Pritchard, John Eliot.

**Year:** 2012

The authors of this book are John Eliot – a consultant in professional sports leagues such as Major League Baseball, the National Football League, and the National Basketball Association, as well Fortune 500 companies – and Kevin Pritchard – the general manager for the NBA's Indiana Pacers. Here you will find tons of unique, behind closed doors information about teamwork that is based on sports experience.

### What Is So Special About This Book?

The book describes in a great way what is so dangerous about disbalance in teams. The authors explain why organizations that value their players succeed. Each chapter contains examples that can shed light on new ideas for you.

## 8. The Alliance: Managing Talent in the Networked Age

**Authors:** Reid Hoffman, Ben Casnocha, Chris Yeh.

**Year:** 2014

Just look at the names of those who had a hand in creating this book – Reid Hoffman, co-founder of LinkedIn, and entrepreneurs Ben Casnocha and Chris Yeh. They claim that the former approach where employer and employee established long-term relations doesn't exist anymore. They studied the modern relationship between employer and employee and show how collaboration can transform work into a pleasant experience.

### What Is So Special About This Book?

The book contains plenty of unique information. The authors intentionally pay attention to the notion that any company should not be a family with its values. On the contrary, it should be a sports team that has definite aims.

## 9. A Team of Leaders: Empowering Every Member to Take Ownership, Demonstrate Initiative, and Deliver Results

**Authors:** Paul Gustavson, Stewart Liff.

**Year:** 2014

The book contains stories from the authors' personal experience about the ways different teams implemented methods of work and how they helped them to move toward success.

### What Is So Special About This Book?

It will help not only managers but all other team members. Thanks to the described ideas, teams are able to determine the ways to achieve success regardless of how effectively they are working now.

## 10. Debugging Teams: Better Productivity Through Collaboration

**Authors:** Brian W. Fitzpatrick, Ben Collins-Sussman.

**Year:** 2015

Having many years of experience in software development, and then as team leaders and managers, the authors share their thoughts and ideas in this book. They describe, in detail, behavioral patterns and the impossibility of predicting or foreseeing them. Fitzpatrick and Collins-Sussman give their advice on how to solve such problems and come out victorious.

The authors also claim that people tend to ignore human factors during work processes. But it is of high importance to learn how to collaborate and work in a team. It raises your chances of successfully accomplishing the goals of a software development project.

### What Is So Special About This Book?

It contains tons of real examples. It is written in conversational style mixed with good humor. Many pieces of advice can be successfully applied in everyday life. But software developers will benefit the most from reading it.

---

Discover how TDM Is Essential To Achieving Quality At Speed For Agile, DevOps, And Continuous Delivery. Brought to you in partnership with CA Technologies.

---

Topics: TEAM MANAGEMENT, PROJECT MANAGEMENT, TEAM COLLABORATION, AGILE

Opinions expressed by DZone contributors are their own.

