# One to one

### User

one user can have one profile - @OneToOne

## Get

<http://localhost:8080/api/users>

<http://localhost:8080/api/users/1>

## Create

<http://localhost:8080/api/users>

{

    "name": "John Doe",

    "email": "john.doe@example.com",

    "profile": {

        "address": "123 Main St",

        "phoneNumber": "123-456-7890"

    }

}

## Update

<http://localhost:8080/api/users/1>

{

    "name": "John Doe",

    "email": "john.doe@example.com",

    "profile": {

        "address": "123 Main St",

        "phoneNumber": "123-456-7890"

    }

}

## Delete

<http://localhost:8080/api/users/1>

# One to many

### Author & Books

one author can have many book - @OneToMany

many book can have one author - @ManyToOne

Hibernate: create table author (id bigint not null auto\_increment, name varchar(255), primary key (id)) engine=InnoDB

Hibernate: create table book (id bigint not null auto\_increment, title varchar(255), author\_id bigint, primary key (id)) engine=InnoDB

Hibernate: alter table book add constraint FKklnrv3weler2ftkweewlky958 foreign key (author\_id) references author (id)

## Create author

<http://localhost:8080/api/authors>

{

    "name": "George R.R. Martin",

    "books": [

        {

            "title": "A Game of Thrones"

        },

        {

            "title": "A Clash of Kings"

        },

        {

            "title": "A Storm of Swords"

        }

    ]

}

## Get authors

<http://localhost:8080/api/authors>

## Get books

<http://localhost:8080/api/books>

## Create book

{

    "title": "A Music with Goats",

    "authorId": 1

}