Orbis Software GmbH

# Core Server

RESTful API Dokumentation

Osmer.Julian /ORBIS
29.3.2019

# Inhalt

# RESTful Services

## Architektur

SQL-Tabelle: Person → Entitiy: Person → Repository: PersonRepository → RESTful Service: PersonService

## Erreichbarkeit

Der *Web context path* für jede RESTful API entspricht grundsätzlich dem Entitiy-Namen in der underscore Schreibweise und ohne „Api“.

- Für PersonService → person → URL:PORT/api/person/…
- Für CustomerContactService → customer_contact → URL:PORT/api/customer_contact/…

## CRUD-Operationen

Jede Entity basierte RESTful Service bietet grundsätzlich, mindestens die CRUD-Operationen:

- **GET: URL:PORT/api/person/get/1**
  - HTML header Parameter: token: xyz
  - Parameter: Person-Id: 1
  - Gibt zurück: Person
- **GET: URL:PORT/api/person/all**
  - HTML header Parameter: token: xyz
  - Gibt zurück: Alle Personen
- **POST: URL:PORT/api/person/**
  - HTML header Parameter: token: xyz
  - Parameter: Entitiy: Person
  - Antwort erfolgreich: {"message":"1"}
  - Antwort nicht erfolgreich: {"message":"0"}
- **PUT: URL:PORT/api/person/**
  - HTML header Parameter: token: xyz
  - Parameter: Entitiy: Person **(Mit Id)**
  - Antwort erfolgreich: {"message":"1"}
  - Antwort nicht erfolgreich: {"message":"0"}
- **DELETE: URL:PORT/api/person/delete/1**
  - HTML header Parameter: token: xyz
  - Parameter: Person-Id: 1
  - Antwort erfolgreich: {"message":"1"}
  - Antwort nicht erfolgreich: {"message":"0"}
- **Gilt grundsätzlich für alle Operationen:**
  - Falls der Nutzer das entsprechende Recht nicht hat:
    - Antwort: {"message":"access denied"}
  - Falls bei einem Operationen andere Entities verändert werden müssen, wird dies vom Backend automatisch erledigt!

## Nutzerrechte

Für jede Entity basierte RESTful API und jeden Nutzer können die Rollen: GET, POST, PUT und DELETE jeweils einzeln gesetzt werden.

- Nutzer1 → Darf auf PersonApi nur GET und PUT Operationen ausführen
- Nutzer1 → Darf auf CustomerContactApi GET, POST, PUT und DELETE Operationen ausführen
- Nutzer2 → Darf auf PersonApi nur GET und POST Operationen ausführen
- Nutzer2 → Darf auf CustomerContactApi GET, POST und PUT Operationen ausführen

Falls Nutzer1 jetzt eine POST Anfrage auf PersonApi macht, bekommt er als Antwort:
→ {"message":"access denied"}

## Nutzerpasswort

Neue Nutzer können nur erfolgreich angelegt werden, wenn das Nutzer-Passwort folgenden Kriterien entspricht und der Login Name nicht vergeben würde:

- Muss mindestens aus 9 Zeichen bestehen
- Darf nicht den Loginnamen enthalten
- Muss mindestens eine Zahl haben
- Muss mindestens ein Sonderzeichen haben
- Muss Kleinbuchstaben und Großbuchstaben haben

## Rootnutzer

Ist verfügbar und darf immer alles!

- Login: root
- Passwort: OrsWin9IstGeil!

## Token

Besteht aus einem Key und der Nutzer-Id und wird verschlüsselt an das Frondend ausgeliefert.

- Key: asdf
- Nutzer-Id: 1
- → Ergibt den Token: asdf-1

## Login

Erreichbar über:

→ **GET: URL:PORT/api/login/authentication**

- HTML header Parameter: login: root, pw: OrsWin9IstGeil!
- Erfolgreich → {"token":"093b0286137b1402b06f3e942cb4e9a4"}
- Nicht Erfolgreich → {"message":"login failed"}

Der Token erneuert sich alle 24 Stunden und eine RESTful API Anfrage mit einem veraltetet Token hat eine {"message":"access denied"} Antwort vom Server zur Folge. Der Nutzer sollte dann wieder auf die Login-Seite weitergeleitet werden und muss sich erneut anmelden.

## CRUD RESTful Services

Bieten mindestens die CRUD-Operationen für eine bestimmte Entity.

### CustomerService

→ Hierüber können die Kunden verwaltet werden.

- SQL-Table: customer
- Entity: Customer
- Repository: CustomerRepository
- Web context path: customer

### CustomerAddressService

→ Hierüber können abweichenden Lieferanschriften des Kunden verwaltet werden.

- SQL-Table: customer_address (foreign_Key: customer)
- Entity: CustomerAddress
- Repository: CustomerAddressRepository
- Web context path: customer_address

Hierüber können alle Lieferanaschriften  eines bestimmten Kunden geladen werden:

- **GET: URL:PORT/api/customer_address /all/key/1**
    - HTML header Parameter: token: xyz
    - Parameter: foreign_Key: 1 (customer_id)
    - Gibt zurück: Alle Personen die zum Customer gehören

### CustomerContactService

→ Hierüber können die Ansprechpartner zum Kunden verwaltet werden.

- SQL-Table: customer_contact (foreign_Key: customer)
- Entity: CustomerContact
- Repository: CustomerContactRepository
- Web context path: customer_contact

Hierüber können alle Ansprechpartner eines bestimmten Kunden geladen werden:

- **GET: URL:PORT/api/customer_contact/all/key/1**
    - HTML header Parameter: token: xyz
    - Parameter: foreign_Key: 1 (customer_id)
    - Gibt zurück: Alle Personen die zum Customer gehören

## UserRightService

→ Hierüber können die Nutzerrechte verwaltet werden.

- SQL-Table: user_right (foreign_Key: user)
- Entity: UserRight
- Repository: UserRightRepository
- Web context path: user_right

Hierüber sind die Rechte eines Nutzers einsehbar.

- **GET: URL:PORT/api/user_right/getRights/id/1**
  - HTML header Parameter: token: xyz
  - Parameter: foreign_Key: 1 (customer_id)
  - Gibt zurück: Die Nutzerrechte (RightsBean)

Hierüber sind die Rechte eines Nutzers geändert werden.

- **PUT: URL:PORT/api/user_right/updateRights/**
  - HTML header Parameter: token: xyz
  - Parameter: RightsBean
  - Gibt zurück: Anzahl der Änderungen

## UserService

→ Hierüber können die Nutzer verwaltet werden.

- SQL-Table: user
- Entity: User
- Repository: UserRepository
- Web context path: user

Hierüber kann ein neues Passwort angefordert werden:

- **GET: URL:PORT/api/user/new_pw**
  - HTML header Parameter: token: xyz
  - Antwort erfolgreich: {"message":"pL3wCEG4X1!"}
  - Antwort nicht erfolgreich: {"message":"something went wrong"}
    - In diesem Fall würde das Passwort nicht geändert!

Hierüber kann das Passwort geändert werden:

- **PUT: URL:PORT/api/user/update_pw**
  - HTML header Parameter: token: xyz
  - Parameter: {"login":"roor","pw":"neuesPasswort!1"}
  - Antwort erfolgreich: {"message":"1"}
  - Antwort nicht erfolgreich: {"message":"0"}

## Andere RESTful Services

Bieten keine CRUD-Operationen und sind keinen Entities zugeordnet.

### ServerApi

- **Serverlogs: URL:PORT/api/server/logs.html**
- **Serverinfos: URL:PORT/api/server/index.html**

### LoginApi

Siehe Login

## Entities

Die Felder id, created, und modified werden vom Backend bzw. von der Datenbank automatisch verwaltetet und können vom Frondend aus nicht manipuliert werden.

### User

- Das Passwort ("pw": "") ist immer ein leerer String
- Bei einem Save ist isActive immer true
- lastLogin wird vom Backend verwaltet


```
{
   "created":"2019-03-29 12:33:15",
   "id":1,
   "modified":"2019-03-29 12:33:15",
   "isActive":true,
   "lastLogin":"2000-01-01 01:00:00",
   "pw":"",
   "userEmail":"user@email1"
}
```


### UserRight


```
{
   "created":"2019-03-27 11:12:59",
   "id":1,
   "modified":"2019-03-27 11:12:59",
   "service":"service1",
   "serviceRight":"right1",
   "userId":1
}
```

### Customer


```
{
   "commercial": false,
   "companyName": "Hansen & Co KG",
   "address": {
      "streetName": "Super Str.",
      "streetNr": "1",
      "zipCode": "20000",
      "place": "Oyten",
      "telephone": "05-345-34",
      "email": "info@mail.de",
      "fax": "4324-5345-435345"
   },
   "person": {
      "title": "Dr. med.",
      "salutation": "Herr",
      "firstName": "Karsten",
      "lastName": "Müller"
   },
   "id": 1,
   "created": "2019-03-01 10:08:51",
   "modified": "2019-03-01 10:08:51"
}
```


### CustomerContact


```
{
   "customerId": 1,
   "address": {
      "streetName": "Super Str.",
      "streetNr": "1",
      "zipCode": "20000",
      "place": "Oyten",
      "telephone": "05-345-34",
      "email": "info@mail.de",
      "fax": "4324-5345-435345"
   },
   "person": {
      "title": "Dr. med.",
      "salutation": "Herr",
      "firstName": "Karsten",
      "lastName": "Müller"
   },
   "id": 1,
   "created": "2019-03-01 10:11:27",
   "modified": "2019-03-01 10:11:27"
}
```

### CustomerAddress


```
{
   "customerId": 1,
   "type": "Lieferanschrift",
   "address": {
      "streetName": "Super Str.",
      "streetNr": "1",
      "zipCode": "20000",
      "place": "Oyten",
      "telephone": "05-345-34",
      "email": "info@mail.de",
      "fax": "4324-5345-435345"
   },
   "id": 1,
   "created": "2019-03-01 10:15:19",
   "modified": "2019-03-01 10:15:19"
}
```


## Beans

### AddressBean


```
{
   "streetName": "Super Str.",
   "streetNr": "1",
   "zipCode": "20000",
   "place": "Oyten",
   "telephone": "05-345-34",
   "email": "info@mail.de",
   "fax": "4324-5345-435345"
}
```


### PersonBean


```
{
   "title": "Dr. med.",
   "salutation": "Herr",
   "firstName": "Karsten",
   "lastName": "Müller"
}
```

### LoginBean

```
{
   "login": "login",
   "pw": "OrsWin9IstGeil!"
}
```

### TokenBean

```
{
   "token": "62d729beb4bd0aa69895e4f3c4b8297e"
}
```

### ResponseBean

```
{
   "message": "msg"
}
```

### RightsBean

```
{
   "login":"user@abce.de",
   "rights":[
      {
         "service":"service",
         "serviceRight":"GET"
      },
      {
         "service":"service",
         "serviceRight":"PUT"
      },
      {
         "service":"user_right",
         "serviceRight":"GET"
      },
      {
         "service":"user_right",
         "serviceRight":"PUT"
      },
      {
         "service":"user_right",
         "serviceRight":"POST"
      },
      {
         "service":"user_right",
         "serviceRight":"DELETE"
      }
   ],
   "userId":1
}
```