内容

[1. 開発環境 3](#_Toc60059977)

[1.1. 仮想環境の構築 3](#_Toc60059978)

[1.2. Djangoなどの設定 3](#_Toc60059979)

[1.3. Setting.pyの設定 5](#_Toc60059980)

[1.3.1. INSTALLED\_APPSの設定 5](#_Toc60059981)

[1.3.2. データベースの設定 7](#_Toc60059982)

[1.3.3. TEMPLATES用の追加 7](#_Toc60059983)

[1.3.4. ロギングの設定 7](#_Toc60059984)

[1.3.5. 定数 10](#_Toc60059985)

[2. Userテーブルのカスタマイズ項目 13](#_Toc60059986)

[2.1. 全体 13](#_Toc60059987)

[2.2. 顧客情報 14](#_Toc60059988)

[2.2.1. 顧客属性 14](#_Toc60059989)

[2.2.2. 購買の履歴 16](#_Toc60059990)

[2.2.3. 収益性 17](#_Toc60059991)

[3. Web系の開発(最新) 19](#_Toc60059992)

[3.1. コンセプト 19](#_Toc60059993)

[3.2. 開発方法 19](#_Toc60059994)

[3.3. 権限について 19](#_Toc60059995)

[4. 開発方法(OLD) 21](#_Toc60059996)

[4.1. 仮想環境の作成 21](#_Toc60059997)

[5. コマンド 22](#_Toc60059998)

[6. DBについて 22](#_Toc60059999)

[7. データの作成方法 23](#_Toc60060000)

[7.1. データの作成 23](#_Toc60060001)

[8. VS-code 25](#_Toc60060002)

[9. ログイン管理について 26](#_Toc60060003)

[9.1. 総論 26](#_Toc60060004)

[10. 開発コンセプト 26](#_Toc60060005)

[10.1. バージョン0.1 26](#_Toc60060006)

[10.1.1. Admin系の開発 26](#_Toc60060007)

[10.1.2. 課題事項 26](#_Toc60060008)

[11. マスタ系テーブルデータ 29](#_Toc60060009)

[11.1. Jobテーブル 29](#_Toc60060010)

[11.1.1. テーブル構造 29](#_Toc60060011)

[11.1.2. データ設定 29](#_Toc60060012)

[11.2. 性別マスタテーブル 30](#_Toc60060013)

[11.2.1. テーブル構造 30](#_Toc60060014)

[11.2.2. データ設定 30](#_Toc60060015)

[11.3. dtb\_customer(会員テーブル) 31](#_Toc60060016)

[11.3.1. テーブル構造 31](#_Toc60060017)

[11.4. mtb\_device\_type(デバイスの種類マスタテーブル) 32](#_Toc60060018)

[11.4.1. テーブル構造 32](#_Toc60060019)

[11.4.2. データ設定 33](#_Toc60060020)

# 開発環境

## 仮想環境の構築

|  |  |
| --- | --- |
| C:\Python>mkdir aoyama  C:\Python>cd aoyama  C:\Python\aoyama >pip install virtualenv | これは1回のみの実行 |
| 以下のコマンドを実行して仮想環境にする  C:\Python\aoyama>python -m venv env  C:\Python\aoyama >.\env\Scripts\activate  (env) C:\Python\aoyama>python -m pip install --upgrade pip | 仮想環境を構築  この中でnode系をインストール  pip をアップデート |

## Djangoなどの設定

|  |  |  |
| --- | --- | --- |
| django | 本体 | python -m pip install Django |
| djangorestframework | Django rest | python -m pip install djangorestframework |
| markdown | markdown | python -m pip install markdown |
| django-filter | Filtering support | python -m　 pip install django-filter |
| django-debug-toolbar | デバッグ用 | python -m pip install django-debug-toolbar |
| django-mysql | Json型 | python -m pip install django-mysql |
| mysqlclient | MySQLクライアント | python -m pip install mysqlclient |
| django-allauth | 認証 | python -m pip install django-allauth |
| pillow | ImageField | python -m pip install pillow |
|  |  |  |
|  |  |  |
| django-bootstrap4 | Bootstrap4 | python -m pip install django-bootstrap4 |
| django-pure-pagination | ページネーション | python -m pip install django-pure-pagination |

(env) C:\Python\aoyama> python -m pip freeze

|  |
| --- |
| (env) C:\Python\aoyama>pip freeze  asgiref==3.3.1  astroid==2.4.2  autopep8==1.5.4  certifi==2020.12.5  cffi==1.14.4  chardet==4.0.0  colorama==0.4.4  cryptography==3.3.1  defusedxml==0.6.0  Django==3.1.4  django-allauth==0.44.0  django-debug-toolbar==3.2  django-mysql==3.10.0  djangorestframework==3.12.2  idna==2.10  isort==5.6.4  lazy-object-proxy==1.4.3  Markdown==3.3.3  mccabe==0.6.1  mysqlclient==2.0.2  oauthlib==3.1.0  Pillow==8.0.1  pycodestyle==2.6.0  pycparser==2.20  PyJWT==2.0.0  pylint==2.6.0  python3-openid==3.2.0  pytz==2020.4  requests==2.25.1  requests-oauthlib==1.3.0  six==1.15.0  sqlparse==0.4.1  toml==0.10.2  urllib3==1.26.2  wrapt==1.12.1 |

## Setting.pyの設定

### INSTALLED\_APPSの設定

ソースで保存

現状は　allauthはテストのみ、次のバージョンで統合

django-allauth facebookは利用しない設定

デザインは変更しない

Uncaught (in promise) TypeError: this.engines is not iterable

at E.updateSocialMedia (content.js:48)

at new E (content.js:48)

at Function.start (content.js:48)

at Function.main (content.js:48)

McAfeeChrome拡張機能が問題の原因

|  |  |  |
| --- | --- | --- |
| 追加項目 | ユーザテーブルのカスタマイズ用 | 'users.apps.UsersConfig', |
| django-allauthの追加  socialaccountの設定は必須になります。 | 'allauth',  'allauth.account',  'allauth.socialaccount',  #主要なソーシャル  allauth.socialaccount.providers.facebook',  'allauth.socialaccount.providers.github',  'allauth.socialaccount.providers.twitter', |
| サイトマップ用 | 'django.contrib.sites',  'django.contrib.sitemaps', |
| デバッグ用 | 'debug\_toolbar', |
| Bootstrap4用 | 'bootstrap4', |
| ページネーション用 | 'pure\_pagination', |

### データベースの設定

|  |  |
| --- | --- |
| ENGINE | django.db.backends.mysql |
| NAME | DB名 |
| USER | DBのユーザ名 |
| PASSWORD | DBのパスワード名 |
| OPTIONS | 'init\_command':  "SET sql\_mode='STRICT\_TRANS\_TABLES'",  ※不要なワーニングの削除 |
| HOST | ホスト名 |
| ポート番号 | 3306 |

### TEMPLATES用の追加

|  |
| --- |
| # allauthの追加  'DIRS': [  # ログインテンプレート等などを置く場所  os.path.join(BASE\_DIR, 'templates', 'allauth'),  # テンプレートを置く場所  # os.path.join(BASE\_DIR, 'templates'),  ], |

### ロギングの設定

|  |
| --- |
| LOGGING = {  'version': 1,  'disable\_existing\_loggers': False,  'formatters': {  'myformat': {  'format': '{levelname} {asctime} {pathname}.{module}.{funcName}:{lineno} {message}',  'style': '{',  },  'simple': {  'format': '%(levelname)s %(message)s'  },  },  'handlers': {  'file\_debug': {  'level': 'DEBUG',  'class': 'logging.FileHandler',  'filename': './log/debug.log',  'formatter': 'myformat',  },  'file\_info': {  'level': 'INFO',  'class': 'logging.FileHandler',  'filename': './log/info.log',  'formatter': 'myformat',  },  'file\_err': {  'level': 'ERROR',  'class': 'logging.FileHandler',  'filename': './log/error.log',  'formatter': 'myformat',  },  'console': {  'level': 'ERROR',  'class': 'logging.StreamHandler',  'formatter': 'simple'  },  },  'loggers': {  'django': {  'handlers': ['console', 'file\_debug'],  'level': 'DEBUG',  'propagate': True,  },  'infofile': {  'handlers': ['file\_info', 'file\_err'],  'level': 'INFO',  'propagate': True,  },  'myapp': {  'handlers': ['console'],  'level': 'ERROR',  'propagate': True,  },  },  } |

### 定数

|  |  |
| --- | --- |
| LANGUAGE\_CODE | ja |
| TIME\_ZONE | Asia/Tokyo |
| サイト用 | SITE\_ID = 1 |
| debug用 | INTERNAL\_IPS = [  '127.0.0.1',  ] |
| CSS、js画像置き場 | STATIC\_URL = '/static/' |
| Admin系のカスタマイズ用の  CSS,JS、画像置き場 | STATICFILES\_DIRS = (  os.path.join(BASE\_DIR, 'static'),  ) |
| django-pure-pagination用の設定 | PAGINATION\_SETTINGS = {  'PAGINATION\_DEFAULT\_WINDOW': 2,  'PAGE\_RANGE\_DISPLAYED': 2,  'MARGIN\_PAGES\_DISPLAYED': 2,  'SHOW\_FIRST\_PAGE\_WHEN\_INVALID': True,  } |
| 管理サイトの名称設定 | ADMIN\_SITE\_HEADER = '青山システムズ'  ADMIN\_SITE\_TITLE = '全体管理'  ADMIN\_INDEX\_TITLE = 'サイト管理' |
| Userテーブルのカスタマイズ | AUTH\_USER\_MODEL = 'users.User' |
| メール設定 | # メールのバックエンド処理  EMAIL\_BACKEND = 'django.core.mail.backends.console.EmailBackend'  #EMAIL\_BACKEND = 'django.core.mail.backends.smtp.EmailBackend'  # メール送信用  EMAIL\_HOST = 'smtp.gmail.com'  EMAIL\_PORT = 587  EMAIL\_USE\_TLS = True  EMAIL\_HOST\_USER = 'aoyama414@gmail.com'  EMAIL\_HOST\_PASSWORD = 'rzorycojylqfksix' # アプリパスワード |
| 画像アップ用のディレクトリ設定 | MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')  MEDIA\_URL = '/media/' |
| 認証方法 | AUTHENTICATION\_BACKENDS = [  'django.contrib.auth.backends.ModelBackend',  'allauth.account.auth\_backends.AuthenticationBackend',  ] |
| django-allauthの各種設定 | SITE\_ID = 1  LOGIN\_REDIRECT\_URL = 'admin:index' # ログイン後遷移する画面を指定  ACCOUNT\_LOGOUT\_REDIRECT\_URL = 'users:logout' # ログアウト後遷移する画面の指定  ACCOUNT\_AUTHENTICATION\_METHOD = 'email' # 認証方法をメールアドレスにする  ACCOUNT\_USER\_MODEL\_USERNAME\_FIELD = None # Userモデルにusernameは無い  ACCOUNT\_EMAIL\_REQUIRED = True # メールアドレスを要求する  ACCOUNT\_USERNAME\_REQUIRED = False # ユーザ名を要求しない  SOCIALACCOUNT\_PROVIDERS = {  # 'facebook': {  # 'SCOPE': ['email', 'publish\_stream'],  # 'AUTH\_PARAMS': {'auth\_type': 'reauthenticate'},  # 'METHOD': 'oauth2',  # 'LOCALE\_FUNC': lambda request: 'ja\_JP'  # },  'facebook': {  'METHOD': 'oauth2',  # 'SDK\_URL': '//connect.facebook.net/{locale}/sdk.js',  'SCOPE': ['email', 'public\_profile'],  'AUTH\_PARAMS': {'auth\_type': 'reauthenticate'},  'INIT\_PARAMS': {'cookie': True},  'FIELDS': [  'id',  'first\_name',  'last\_name',  'middle\_name',  'name',  'name\_format',  'picture',  'short\_name'  ],  'EXCHANGE\_TOKEN': True,  'LOCALE\_FUNC': lambda request: 'ja\_JP',  'VERIFIED\_EMAIL': False,  'VERSION': 'v7.0',  },  'github': {  'SCOPE': ['user:follow', 'gist']  }  } |

# Userテーブルのカスタマイズ項目

## 全体

|  |  |
| --- | --- |
| ID | UUIDを利用 |
| Username | 利用せず、情報は電子メールをより取得する |
| その他については | 各テーブルでユーザに紐づけて作成 |

## 顧客情報

EC-CUBEなどを参照すべし

https://www.shiro8.net/manual4/v40x/index.html

### 顧客属性

| 分類 | 項目 | 必須有無 | 内容 |
| --- | --- | --- | --- |
| 定量情報 | ID |  | システム |
| 個人：法人区分 | 必須 | 個人と法人を分類 |
| □個人情報 | | |
| 氏名 | 必須 | 50文字 |
| 氏名フリガナ | 必須 | 50文字 |
| 生年月日 | 必須 |  |
| 性別 | 選択 | 男性、女性、性別回答なし |
| 職業欄 | 選択：必須 | 会社員など |
| 趣味 | 選択：任意 |  |
| □法人情報 | | |
| 会社名 | 必須 | 150文字 |
| 会社名フリガナ | 必須 | 150文字 |
| 部署 | 必須 | 150文字 |
| 役職 | 必須 | 150文字 |
| URL | 任意 | 150文字 |
| □個人・法人共通項目 |  |  |
| 連絡先 | 必須 | 電話番号・・・ |
| 連絡方法 | 選択 | 電話、メール、郵便 |
| メール情報配信有無区分 | 選択 | 有、無 |
| 書類送付配信有無区分 | 選択 | 有、無 |
| 郵便番号 | 必須 | 7桁 |
| 郵便番号選択ボタン | 日本郵便をJSで  呼び出す | 日本郵便よりデータを取得 |
| 都道府県 | 必須 | 47都道府県 |
| 住所1(市区町村および町名) | 必須 | 150文字 |
| 住所2(番地および建物名) | 必須 | 150文字 |
| 電話番号 | 電話番号か携帯電話  が必須 | 数字のみ15文字 |
| 携帯電話 |  | 数字のみ15文字 |
| FAX | 任意 | 数字のみ15文字 |
| メールアドレス | 必須 | 150文字 |
| 画像（写真） |  |  |
| 定性情報 | 初回来店日 |  |  |
| 購買志向 |  |  |
| 購買ニーズ（理由） |  |  |
| 紹介者 |  |  |
| ライフスタイル（朝型、夜型） |  |  |
| ブランド意識 |  |  |

### 購買の履歴

| 分類 | 項目 | 必須有無 | 内容 |
| --- | --- | --- | --- |
| 購買履歴 | ID |  |  |
| ユーザID |  |  |
| 買い上げ日時 |  |  |
| 購入商品 |  |  |
| 価格 |  |  |
| 数量 |  |  |
| 金額 |  |  |
| 割引対象 |  | イベントセールス、クーポンなど |
| 割引額 |  |  |
| 支払い方法 |  | クレジット、現金など |
| 販売担当者 |  | 人またはシステムなどを想定 |

### 収益性

| 分類 | 項目 | 必須有無 | 内容 |
| --- | --- | --- | --- |
| 収益性 | ID |  |  |
| 年月 |  |  |
| 予定売上 |  |  |
| 実績売上 |  | 年月単位の売上実績 |
| 予定利益 |  |  |
| 実績利益 |  |  |
| 予定購入単価 |  |  |
| 実績購入単価 |  |  |
| 予定購買頻度 |  |  |
| 実績購買頻度 |  |  |
| ライフタイム |  |  |
| バリュー金額 |  |  |

ライフタイムバリュー（LTV)とは営業をしていると、よく「ライフタイムバリュー（LTV)」という言葉がでてきます。

ライフタイムバリューとは、英語で、Life Time Valueです。略して、LTVなどと呼びます。

日本語では、「顧客生涯価値」などと呼んでいます。ライフタイムバリューは、１人の人間、お客様に対して捉えます。つまり、１人のお客様が生まれてから死ぬまでに使用するお金のすべてです。

例えば、自動車関連のメーカーですと１人のお客様が自動車関係に使うすべてのお金を、ライフタイムバリュー（LTV)と呼びます。

大学で車の免許を取得し、中古車を買って、その後、新車を購入します。

車検を受け、タイヤの交換、バッテリー、車の備品を購入します。

また、保険に入り、車庫を作ります。

それらのお金のすべてが、ライフタイムバリューです。

以前のように、市場が拡大しているときは、販売シェア、市場占有率をあげることに、注目していましたが、現在は市場は飽和しています。

一定の市場の中では、このライフタイムバリュー（LTV)の中のシェアを獲得しなければ。生き残れない時代になりました。市場が拡大しているとき、シェアの拡大するときは、テレビや新聞、雑誌などのマスメディアで広告を行っていましたが、ライフタイムバリューを獲得するためには、一人一人のお客様に対して個別の状況を把握してアプローチする必要があります。その為には、顧客の情報、データベースが必要になります。

名前、年齢、職業、家族構成、趣味、住まいの状況、駐車場の有無、ペットの有無などいろいろな情報が必要になります。

# Web系の開発(最新)

## コンセプト

* マスタ系の更新などはDjangoまたはPHPのフレームワークで修正を行う
* 開発とデザインについて工数を使う
* シンプルでかつレスポンスが良いもので、斬新なデザインとする
* AIを利用できるようにする
* 多数言語で開発
* 確認画面は作成せず、Jsなどによるモーダル画面によって確認を行う
* 複雑な処理になることはやめよう  
  1日中考えてもわからないことはパスをする
* DjangoでできることはDjangoに任せて困難な処理を行わない

## 開発方法

|  |  |
| --- | --- |
| サーバ | Webサーバ：Apache  DB：MySQL  開発言語：Django REST framework |
| クライアント | 言語：Angularなど |
| デザイン | AngularとAdminLTEを基本としてカスタマイズ |

## 権限について

不動産管理と同じ処理

|  |  |
| --- | --- |
| システム管理者 | 全体管理 |
| システム運用者 | ロールで管理をし、システムの運用面面のみ |
| システムスタッフ | 一部の情報閲覧と修正のみ |
| ユーザ | 自己管理と、表示 |
| ユーザ(BtoB用) | 自己ユーザの管理 |

ユーザ系の変更

Ad－dj4とaoyamaとの統合

# 開発方法(OLD)

* ユーザなどのモデルを作成し、全体のテーブルを作成
* 特にユーザなどの主体となるものは先に作成しないと、削除などの繰り返しを行う必要性がある
* ただし基本となるものを作成した後に追加していくことは可能

## 仮想環境の作成

1. 仮想環境の構築(例えばad-dj)

|  |
| --- |
| C:\Python>mkdir ad-dj  C:\Python>cd ad-dj  C:\Python\ad-dj>pip install virtualenv |

1. 以下のコマンドを実行して仮想環境にする  
   pipは最新にする

|  |
| --- |
| C:\Python\ad-dj>python -m venv env01  C:\Python\ad-dj>.\env01\Scripts\activate  (env01) C:\Python\ad-dj>python -m pip install --upgrade pip |

仮想環境の中でDjangoの開発を行う

# コマンド

|  |  |
| --- | --- |
| プロジェクトの作成 | django-admin startproject プロジェクト名 作成場所 |
| カスタマイズUserモデルのマイグレーションの作成 | python manage.py makemigrations users |
| アプリケーションの作成 | python manage.py startapp　アプリ名 |
| 全体の作成 | python manage.py migrate |
| マイグレーションの確認 | python manage.py showmigrations |
| 仮想環境のスタート | activate |
| 仮想環境の終了 | deactivate |

# DBについて

XAMPで開発をすると、Django-mysqlのJSONデータ型に対応しない

MySQL57以上をインストールする必要性がある。

# データの作成方法

## データの作成

|  |
| --- |
| python manage.py shell  import datetime  import pytz  from app3.models import Person, Manager, Worker  for i in range(1, 1201):  birthday = datetime.datetime(year=1980 + i % 20, month=1 + i % 12, day=1 + i % 28, zinfo=pytz.timezone('Asia/Tokyo'))  Person.objects.create(name="person{}".format(i), birthday=birthday, sex=Person.MAN, address\_from=Person.TOKYO, current\_address=Person.TOKYO, email="person{}@gmail.com".format(i))  dep\_list = [Manager.DEP\_ACCOUNTING, Manager.DEP\_SALES, Manager.DEP\_PRODUCTION, Manager.DEP\_DEVELOPMENT, Manager.DEP \_HR, Manager.DEP\_FIN, Manager.DEP\_AFFAIRS, Manager.DEP\_PLANNING, Manager.DEP\_BUSINESS, Manager.DEP\_DISTR, Manager.DE P\_IS]  for i in range(1, 201):  p = Person.objects.get(id=i)  joined\_date = datetime.datetime(year=2005 + i % 10, month=1 + i % 12, day=1 + i % 28, tzinfo=pytz.timezone('Asia/Tokyo'))  Manager.objects.create(person=p, department=dep\_list[i % 11], joined\_at=joined\_date)  for i in range(201, 1201):  p = Person.objects.get(id=i)  m = Manager.objects.get(id=1 + i % 200)  joined\_date = datetime.datetime(year=2005 + i % 10, month=1 + i % 12, day=1 + i % 28, tzinfo=pytz.timezone('Asia/Tokyo'))  Worker.objects.create(person=p, manager=m, joined\_at=joined\_date)  >>> quit() |
|  |

# VS-code

拡張機能追加：Python,Django

* Ctrl + Shift + pでコマンドパレットを開き
* Python:Select Interpreterを選択
* 作成した仮想環境のPythonを選択する
* 左下が仮想環境のPythonを参照していたら成功

設定ファイルを作成する。

* ▷ボタンをクリック→Create a launch.json fileをさらにクリック
* Pyhonを選択
* Djangoを選択
* launch.jsonファイルが作成されるので▷をクリックして実行する

Starting development server at http://127.0.0.1:8000/

Quit the server with CTRL-BREAK.

と出力されたら成功

# ログイン管理について

## 総論

adminは通常のDjangoのログイン管理

allauthはログインについて補足するのと、管理者以外のログインを担当することも可能？

# 開発コンセプト

考えていること

* ログインで通常の画面系
* ログインで管理用の画面系に
* ログイン画面でgithubなどのユーザでログイン管理を可能とする
* Usernameがメールアドレスなので今後は姓名で表示

## バージョン0.1

### Admin系の開発

* ログインは電子メールとパスワード
* ログイン後は権限に応じてマスタ管理などが行うことが可能
* 権限に応じて登録、更新、削除(論理削除)、表示、帳票印刷が可能
* システム管理が行えること

### 課題事項

|  |  |  |
| --- | --- | --- |
| 課題 | 内容 | 解決有無 |
| Admin系 | Admin系ユーザ管理はaccessで行う  パスワードの変更及びリセットに関しては修正済 | 12/10済 |
| Admin系 | Admin系ユーザ管理はaccessで行う  ユーザの追加、メールとパスワードのみ、確認画面を追加 | 12/10済 |
| Admin系 | Admin系ユーザ管理はaccessで行う  ユーザのリストと追加画面との連携  ユーザ追加の際は、電子メールを送信し、パスワードを知らせる | 12/11済 |
| Admin系 | Admin系ユーザ管理はaccessで行う  ユーザの更新とリストとの連携 |  |
| Admin系 | ACCESSをなくし、Users系にまとめる  プロファイルのモデル、テーブルの作成 | 12/26 |

Admin系

　ログイン　⇒　ダッシュボード(マスタ管理、履歴情報の閲覧)　とりあえずパイロット版の作成　12/9

　　　　　　　⇒　システム管理用

　マスタ管理画面の作成

いずれかのアプリを一般用画面として作成

顧客管理系

　ログイン　⇒　ダッシュボード(企業向け)

　ログイン　⇒　ダッシュボード(個人向け)

ログインなし

　表示が見られるだけ

修正テンプレート

|  |  |
| --- | --- |
| \templates\admin\auth\user | add\_form.html  change\_password.html |
| \templates\admin | change\_form.html |
| \templates\admin\include | fieldset.html |
| \templates\admin\lib | \_main\_header.html |
| \templates\registration | logged\_out.html  password\_change\_done.html  password\_change\_form.html  password\_reset\_complete.html  password\_reset\_confirm.html  password\_reset\_done.html  password\_reset\_email.html  password\_reset\_form.html  password\_reset\_subject.txt |
| \adminlte3\templates\adminlte\lib | \_main\_footer.html  \_main\_sidebar.html |
| \adminlte3\_theme\templates\admin | change\_form.html  index.html |
| \adminlte3\_theme\templates\admin\includes | fieldset.html |
|  |  |

|  |  |
| --- | --- |
| \templates | 共通のテンプレート |
| \adminlte3\_theme | 管理者向け |
| \adminlte3 | 通常ユーザ(ログインなしの画面周り) |

# マスタ系テーブルデータ

## Jobテーブル

### テーブル構造

|  |  |  |
| --- | --- | --- |
| カラム名 | 属性(Doctrine) | 説明 |
| id | int | ID |
| name | string | 名称 |
| sort\_no | smallint | 並び順 |

### データ設定

|  |  |  |
| --- | --- | --- |
| id | name | sort\_no |
| 1 | 公務員 | 0 |
| 2 | コンサルタント | 1 |
| 3 | コンピューター関連技術職 | 2 |
| 4 | コンピューター関連以外の技術職 | 3 |
| 5 | 金融関係 | 4 |
| 6 | 医師 | 5 |
| 7 | 弁護士 | 6 |
| 8 | 総務・人事・事務 | 7 |
| 9 | 営業・販売 | 8 |
| 10 | 研究・開発 | 9 |
| 11 | 広報・宣伝 | 10 |
| 12 | 企画・マーケティング | 11 |
| 13 | デザイン関係 | 12 |
| 14 | 会社経営・役員 | 13 |
| 15 | 出版・マスコミ関係 | 14 |
| 16 | 学生・フリーター | 15 |
| 17 | 主婦 | 16 |
| 18 | その他 | 17 |

## 性別マスタテーブル

### テーブル構造

|  |  |  |
| --- | --- | --- |
| カラム名 | 属性(Doctrine) | 説明 |
| id | smallint | ID |
| name | string | 名称 |
| sort\_no | smallint | 並び順 |

### データ設定

|  |  |  |
| --- | --- | --- |
| id | name | sort\_no |
| 1 | 性別回答なし | 0 |
| 2 | 男性 | 1 |
| 3 | 女性 | 2 |

## dtb\_customer(会員テーブル)

### テーブル構造

| カラム名 | 属性(Doctrine) | 説明 |
| --- | --- | --- |
| id | integer | ID |
| name01 | string |  |
| name02 | string |  |
| kana01 | string |  |
| kana02 | string |  |
| company\_name | string |  |
| postal\_code | string | 郵便番号 |
| addr01 | string |  |
| addr02 | string |  |
| email | string | メールアドレス |
| phone\_number | string |  |
| birth | datetimetz |  |
| password | string |  |
| salt | string |  |
| secret\_key | string |  |
| first\_buy\_date | datetimetz |  |
| last\_buy\_date | datetimetz |  |
| buy\_times | decimal |  |
| buy\_total | decimal |  |
| note | string | 備考 |
| reset\_key | string |  |
| reset\_expire | datetimetz |  |
| point | decimal |  |
| create\_date | datetimetz | データ登録日時 |
| update\_date | datetimetz | データ更新日時 |
| customer\_status\_id | integer | 外部キー mtb\_customer\_status |
| sex\_id | integer | 外部キー mtb\_sex |
| job\_id | integer | 外部キー mtb\_job |
| country\_id | integer | 外部キー mtb\_country |
| pref\_id | integer | 外部キー mtb\_pref |

## mtb\_device\_type(デバイスの種類マスタテーブル)

### テーブル構造

|  |  |  |
| --- | --- | --- |
| カラム名 | 属性(Doctrine) | 説明 |
| id | integer | ID |
| name | string | 名称 |
| sort\_no | integer | 並び順 |

### データ設定

|  |  |  |
| --- | --- | --- |
| id | name | sort\_no |
| 2 | モバイル | 0 |
| 10 | PC | 1 |