用电量项目接口文档_v1.0

1. 更新记录

1.1 第一次更新(2017/10/26)

> 4.4 添加用户的更新
>
> 修改返回参数 "data"，由 null 改为 刚添加的用户的信息
>
> 4.5 删除用户的更新
>
> 修改返回参数 "data"，由 null 改为 刚删除的用户的id

> 4.6 修改单个用户信息的更新
>
> 修改返回参数 "data"，由 null 改为 刚修改的用户的信息
>
> 5.1 导入原始数据的更新
>
> 修改参数提交方式，由 application/json 改为 formData
>
> 5.4 删除单个数据集的更新
>
> 修改返回参数 "data"，由 数据集内容的二维数组 改为 刚删除的数据集的id
>
> 5.5 设置预测的默认值的更新
>
> 修改返回参数 "data"，由 null 改为 设置预测的默认值

1.2 第二次更新(2017/10/26)

> 5.6 查询预测的默认值的更新
>
> 修改返回参数 "algorithm"，由 string数组 改为 组合模型和单模型分开
>
> 6.4 添加接口 查询所有算法
>
> 7.2 查询算法列表的更新
>
> 修改返回参数 "algorithm"，由 string数组 改为 组合模型和单模型分开

1.3 第三次更新(2017/11/1)

> 6.5 添加接口 单独设置默认算法
>
> 7.1 查询用电客户列表的更新
>
> 修改返回参数 "area", "industry"
>
> 7.3 用电量可视化的更新
>
> 添加返回参数 “max” 和 "min" 表示阈值
>
> 7.4 误差可视化-误差柱状图的更新
>
> 修改返回参数 "y" , 添加返回参数 “max” 表示阈值

2. 数据格式统一标准

- 后台返回统一格式：

```
{
  "state":"状态码", // number类型
  "stateInfo":"状态信息",
  "data": "返回数据; 若非查询类操作, 则返回null;"
}
```

3. 登录模块

3.1 请求登录验证码

- 请求 URL：http://ip:8000/utils/valcode
- 请求方式：GET
- 参数提交方式：application/json
- 备注：登录请求验证码
- 请求参数：无
- 返回参数：

```
{
  "state":"状态码", // 0表示成功，1表示失败
  "stateInfo":"状态信息",
  "data": {
    "valcodeUrl":"验证码图片Url" // string类型
  }
}
```

3.2 登录

- 请求 URL：http://ip:8000/user/login
- 请求方式：POST
- 参数提交方式：application/json
- 备注：登录操作
- 请求参数：

```
{
   "account": "账号",      // string类型
   "password": "密码",          // string类型
   "valcode": "验证码"      // string类型
}
```

- 返回参数：

```
{
  "state":"状态码", // 0登录成功，1账号不存在，2密码错误，3验证码错误
  "stateInfo":"状态信息", // 0登录成功，1账号不存在，2密码错误，3验证码错误
  "data": null
}
```

4. 用户模块

4.1 查询当前登录用户信息

- 请求 URL：http://ip:8000/user/userInfo
- 请求方式：GET
- 参数提交方式：application/json
- 备注：登录操作
- 请求参数：无

- 返回参数：

```
{
  "state":"状态码", // 0成功，1失败
  "stateInfo":"状态信息",
  "data": {
    "id":"主键" // number类型
    "userName":"用户名" // string类型
    "account":"账号" // string类型
     "admin":"权限" // string类型，0管理员，1普通用户
  }
}
```

4.2 修改密码

- 请求 URL：http://ip:8000/user/updatePwd
- 请求方式：POST
- 参数提交方式：application/json
- 备注：
- 请求参数：

```
{
   "newPassword": "新密码",         // string类型
   "oldPassword": "旧密码",         // string类型
}
```

- 返回参数：

```
{
  "state":"状态码", // 0表示成功，1表示失败
  "stateInfo":"状态信息", // 0修改成功，1密码错误
  "data": null
}
```

4.3 查看所有用户信息

- 请求 URL：http://ip:8000/user/all
- 请求方式：GET
- 参数提交方式：application/json
- 备注：请求所有用户信息
- 请求参数：无
- 返回参数：

```
{
  "state":"状态码", // 0表示成功，1表示失败
  "stateInfo":"状态信息",
  "data": [
    {
      "id":"用户主键",  // number类型
      "userName":"用户名",  // string类型
      "account":"账号" // string类型
      "admin":"权限标记",  // string类型，0管理员，1普通用户
    },
    ...
  ]  // object数组，用户信息数组
}
```

4.4 添加用户

- 请求 URL：http://ip:8000/user/add
- 请求方式：POST
- 参数提交方式：application/json
- 备注：添加用户
- 请求参数：

```
{
  "userName":"用户名",  // string类型
  "account":"账号" // string类型
  "password": "密码",      // string类型
  "admin":"权限标记",  // string类型，0管理员，1普通用户
}
```

- 返回参数：

```
{
  "state":"状态码", // 0添加成功，1用户名已存在，2账号已存在
  "stateInfo":"状态信息", // 0添加成功，1用户名已存在，2账号已存在
  "data": {
      "id":"刚添加的用户的主键",  // number类型
      "userName":"用户名",  // string类型
      "account":"账号" // string类型
      "admin":"权限标记",  // string类型，0管理员，1普通用户
  } // 刚添加的该用户信息
}
```

4.5 删除用户

- 请求 URL：http://ip:8000/user/delete
- 请求方式：POST
- 参数提交方式：application/json
- 备注：请求所有用户信息
- 请求参数：

```
{
  "id":"待删除用户的主键",  // number类型
}
```

- 返回参数：

```
{
  "state":"状态码", // 0删除成功，1删除失败
  "stateInfo":"状态信息", // 0删除成功，1删除失败
  "data": {
      "id":"刚删除的用户的主键",  // number类型
  } // 刚删除的该用户信息
}
```

4.6 修改单个用户信息

- 请求 URL：http://ip:8000/user/update
- 请求方式：POST
- 参数提交方式：application/json
- 备注：修改单个用户信息。密码不修改时，请求参数 "password":""
- 请求参数：

```
{
  "id":"待修改用户的主键",  // number类型
  "userName":"用户名",  // string类型
  "account":"账号" // string类型
  "password": "密码",      // string类型
  "admin":"权限标记",  // string类型，0管理员，1普通用户
}
```

- 返回参数：

```
{
  "state":"状态码", // 0修改成功，1用户名已存在，2账号已存在
  "stateInfo":"状态信息", // 0修改成功，1用户名已存在，2账号已存在
  "data": {
      "id":"待修改用户的主键",  // number类型
      "userName":"用户名",  // string类型
      "account":"账号" // string类型
      "admin":"权限标记",  // string类型，0管理员，1普通用户
  } // 修改后的该用户信息
}
```

5. 导入数据模块

5.1 导入原始数据

- 请求 URL：http://ip:8000/data/import
- 请求方式：POST

- 参数提交方式：formData
- 备注：使用者通过Excel文件导入原始数据。
- 请求参数：

```
{
  "dataName":"数据集的名字",  // string类型
  "file":"上传的文件"  //
}
```

- 返回参数：

```
{
  "state":"状态码", // 0导入成功，1导入失败
  "stateInfo":"状态信息", // 0导入成功，1导入失败的原因
  "data": null
}
```

5.2 查看所有的原始数据集

- 请求 URL：http://ip:8000/data/all
- 请求方式：GET
- 参数提交方式：application/json
- 备注：使用者查看导入的所有数据集。
- 请求参数：无
- 返回参数：

```
{
  "state":"状态码", // 0成功，1失败
  "stateInfo":"状态信息", //
  "data": [
    {
      "id":"数据集的id", // number类型
      "dataName":"数据集的名字" // string类型
    },
    ...
  ] // object数组，每个元素是一个数据集的信息
}
```

5.3 查看单个数据集

- 请求 URL：http://ip:8000/data/single
- 请求方式：POST
- 参数提交方式：application/json
- 备注：查看单个数据集的内容。
- 请求参数：

```
{
  "id":"数据集的id"  // number类型

}
```

- 返回参数：

```
{
  "state":"状态码", // 0成功，1失败
  "stateInfo":"状态信息", //
  "data": {
      "list":"数据集内容的二维数组" // 二维数组
  }
}
```

5.4 删除单个数据集

- 请求 URL：http://ip:8000/data/delete
- 请求方式：POST
- 参数提交方式：application/json
- 备注：删除单个数据集。
- 请求参数：

```
{
  "id":"数据集的id"  // number类型
}
```

- 返回参数：

```
{
  "state":"状态码", // 0成功，1失败
  "stateInfo":"状态信息", // 0删除成功，1删除失败
  "data": {
      "id":"刚删除的数据集的id"  // number类型
  }
}
```

5.5 设置预测的默认值

- 请求 URL：http://ip:8000/user/setDefault
- 请求方式：POST
- 参数提交方式：application/json
- 备注：设置预测的默认值，包括时间、数据集、算法。
- 请求参数：

```
{
  "month":"要预测的月份数",   // number类型
  "dataSetId":"要设置为默认数据集的id",   // number类型
  "algorithm":"要设置为默认的算法",   // string数组类型
}
```

- 返回参数：

```
{
  "state":"状态码", // 0成功，1失败
  "stateInfo":"状态信息", // 0设置成功，1设置失败
  "data": { // 刚设置的默认值
    "month":"默认要预测的月份数",   // number类型
    "dataSetId":"默认数据集的id",   // number类型
    "dataSetName":"数据集的名字",  // string类型
    "algorithm":"默认的算法"   // string数组类型
  }
}
```

5.6 查询预测的默认值

- 请求 URL：http://ip:8000/user/selectDefault
- 请求方式：GET
- 参数提交方式：application/json
- 备注：查询当前用户的预测的默认值
- 请求参数：无
- 返回参数：

```
{
  "state":"状态码", // 0成功，1失败
  "stateInfo":"状态信息", // 0设置成功，1设置失败
  "data": {
    "month":"默认要预测的月份数",   // number类型
    "dataSetId":"默认数据集的id",   // number类型
    "dataSetName":"数据集的名字",  // string类型
    "algorithm":{ // 算法
        "group":"组合模型数组", // string数组
        "single":"单模型数组" // string数组
    }
  }
}
```

6. 预测模块

6.1 查询当前的预测状态

- 请求 URL：http://ip:8000/forecast/state
- 请求方式：GET
- 参数提交方式：application/json
- 备注：查询系统当前的预测状态。
- 请求参数：无
- 返回参数：

```
{
  "state":"状态码", // 0当前没有在预测，1正在预测中
  "stateInfo":"状态信息", // 0当前没有在预测，1正在预测中
  "data":null
}
```

6.2 正式预测前的预请求

- 请求 URL：http://ip:8000/forecast/before
- 请求方式：POST
- 参数提交方式：application/json
- 备注：预测前将时间、数据集、算法发到后台，后台判断是否已有预测结果。
- 请求参数：

```
{
  "month":"要预测的月份数",   // number类型
  "dataSetId":"用于预测的数据集的id",   // number类型
  "algorithm":"用于预测的的算法",   // string数组类型
}
```

- 返回参数：

```
{
  "state":"状态码", // 0已有预测数据，1没有预测数据
  "stateInfo":"状态信息", // 0“已有部分预测数据，是否利用已有结果”，1“当前无预测数据”
  "data":null
}
```

6.3 正式的预测请求

- 请求 URL：http://ip:8000/forecast/do
- 请求方式：POST
- 参数提交方式：application/json
- 备注：预测前将时间、数据集、算法发到后台，进行正式的预测。
- 请求参数：

```
{
  "month":"要预测的月份数",   // number类型
  "dataSetId":"用于预测的数据集的id",   // number类型
  "algorithm":"用于预测的的算法",   // string数组类型
  "cover":"是否覆盖已有的预测结果"
  // string类型，'0'表示不覆盖，'1'表示覆盖，'2'表示没有已有的预测结果
}
```

- 返回参数：

```
{
  "state":"状态码", // 0预测请求提交成功，1预测请求提交失败
  "stateInfo":"状态信息", // 0预测请求提交成功，1预测请求提交失败
  "data":null
}
```

6.4 查询所有算法

- 请求 URL：http://ip:8000/forecast/allAlgorithm
- 请求方式：GET
- 参数提交方式：application/json
- 备注：查询所有算法。
- 请求参数：无
- 返回参数：

```
{
  "state":"状态码", // 0当前没有在预测，1正在预测中
  "stateInfo":"状态信息", // 0当前没有在预测，1正在预测中
  "data":{
      "algorithm":{ // 算法
        "group":"组合模型数组", // string数组
        "single":"单模型数组" // string数组
    }
  }
}
```

6.5 单独设置默认算法

- 请求 URL：http://ip:8000/user/setDefaultAlgorithm
- 请求方式：POST
- 参数提交方式：application/json
- 备注：设置预测的默认算法。
- 请求参数：

```
{
  "algorithm":"要设置为默认的算法",   // string数组类型
}
```

- 返回参数：

```
{
  "state":"状态码", // 0成功，1失败
  "stateInfo":"状态信息", // 0设置成功，1设置失败
  "data": { // 刚设置的默认值
    "algorithm":"默认的算法"    // string数组类型
  }
}
```

7. 可视化模块

7.1 查询用电客户列表

- 请求 URL：http://ip:8000/customer/all
- 请求方式：GET
- 参数提交方式：application/json
- 备注：查询用电客户列表
- 请求参数：无
- 返回参数：

```
{
  "state":"状态码", // 0成功，1失败
  "stateInfo":"状态信息", //
  "data": {
      "area":{
            "地区1":"用电客户数组",  // string数组
            "地区2":"用电客户数组",  // string数组
            ...
            "地区n":"用电客户数组"  // string数组
      }
      "industry":{
            "行业1":"用电客户数组",  // string数组
            "行业2":"用电客户数组",  // string数组
            ...
            "行业n":"用电客户数组"  // string数组
      }
  } // 按地区、行业对用电客户分组
}
```

7.2 查询算法列表

- 请求 URL：http://ip:8000/visualization/algorithm
- 请求方式：POST
- 参数提交方式：application/json
- 备注：选择用电客户后，查询该用电客户对应的算法列表。
- 请求参数：

```
{
  "customer":"用电客户",   // string类型，单选
}
```

- 返回参数：

```
{
  "state":"状态码", // 0成功，1失败
  "stateInfo":"状态信息", //
  "data":{
    "algorithm":{ // 算法
        "group":"组合模型数组", // string数组
        "single":"单模型数组" // string数组
    }
  }
}
```

7.3 用电量可视化

- 请求 URL：http://ip:8000/visualization/elec
- 请求方式：POST
- 参数提交方式：application/json
- 备注：查询单个用电客户在单个算法模型下的电量。这部分涉及Echarts，不同的图表返回的数据形式不同。
- 请求参数：

```
{
  "customer":"用电客户",   // string类型，单选
  "algorithm":"算法",   // string类型，单选
  "chart":"图的类型",   // string类型。折线图'line'，目前用电量可视化只有折线图。
}
```

- 返回参数：

```
{
  "state":"状态码", // 0成功，1失败
  "stateInfo":"状态信息", //
  "data":
  {
    'line': // 折线图绘图数据
    {
        "x":"x轴，时间序列",  //  string数组
        "y":"用电客户在时间区间内的用电量",    // number数组
        "max":"上阈值"      // number类型
        "min":"下阈值"      // number类型
        "firstForecastMonth":"预测数据的第一个月"
        // string类型，是x轴时间序列上的某个点，如"2017-03"
    }
  }
}
```

7.4 误差可视化-误差柱状图

- 请求 URL：http://ip:8000/visualization/errorBar
- 请求方式：POST
- 参数提交方式：application/json
- 备注：查询单个用电客户的误差。这部分涉及Echart。

- 请求参数：

```
{
  "customer":"用电客户",   // string类型，单选
}
```

- 返回参数：

```
{
  "state":"状态码", // 0成功，1失败
  "stateInfo":"状态信息", //
  "data":
  {
    'bar': // 柱状图绘图数据
    {
        "x":"x轴，用电客户",  //  string类型
        "y":{                  // 误差数据，number数组
            "算法1":"算法1对应该用电客户的误差",    // number类型
            "算法2":"算法2对应该用电客户的误差",    // number类型
            ...,
            "算法n":"算法n对应该用电客户的误差"    // number类型
        },
        "max":"阈值"     // number类型
        "algorithm":"算法数组"         //  string数组，用于图例
    }
 }
}
```