

link: null  
title: 珠峰架构师成长计划  
description: .umirc.ts  
keywords: null  
author: null  
date: null  
publisher: 珠峰架构师成长计划  
stats: paragraph=59 sentences=302, words=3031

### 1.UMi4+Formily2

### 2.创建项目

```
mkdir cms-front && cd cms-front
pnpm dlx create-umi@latest
cnpm i ahooks query-string @formily/core @formily/react @formily/antd --save
pnpm dev
```

### 3.配置项目

.umirc.ts

```
import { defineConfig } from '@umijs/max';
export default defineConfig({
  antd: {},
  access: {},
  model: {},
  initialState: {},
  + request: { dataField: 'data' },
  layout: {
  +   title: 'CMS',
  },
  routes: [
    {
      path: '/',
      redirect: '/home',
    },
    {
      name: '首页',
      path: '/home',
      component: './Home',
    },
  + {
  +   name: '实体管理',
  +   path: '/entity',
  +   component: './Table',
  + }
  ],
  npmClient: 'cnpm',
  + proxy: {
  +   '/api': {
  +     target: 'http://localhost:7001',
  +     pathRewrite: {
  +       '^/api': ''
  +     },
  +     changeOrigin: true
  +   }
  + }
});
```

src/lapp.ts

```
+import { request } from 'umi';
export async function getInitialState(): Promise {
  + return { name: 'CMS' };
}

export const layout = () => {
  return {
    logo: 'https://img.alicdn.com/tfs/TBLYHEpwUTlgK0jSZPhXXaAtVXa-28-27.svg',
    menu: {
      locale: false,
  +   request: () => request('/api/menu').then(res => res.data.list)
    },
  };
};
```

tsconfig.json

```
{
  "extends": "../src/.umi/tsconfig.json",
  + "compilerOptions": {
  +   "strict": false,
  +   "noImplicitAny": false
  + }
}
```

### 4.查询删除实体

.umirc.ts

```
{
  name: '实体管理',
  path: '/entity',
  + component: './Entity',
}
```

src/pages/Entity/index.tsx

```
import { useState } from 'react';
import { Table, Card, Row, Col, Pagination, Space, Modal, List, message, Button } from 'antd';
```

```

import { createForm } from '@formily/core'
import { createSchemaField } from '@formily/react'
import { Form, FormItem, Input, FormGrid, Submit, Reset, DatePicker, FormButtonGroup } from '@formily/antd'
import { FooterToolbar, PageContainer } from '@ant-design/pro-components';
import { request, useRequest, history } from 'umi';
import { useToggle } from 'ahooks';
import moment from 'moment';
import { ExclamationCircleOutlined, SearchOutlined } from '@ant-design/icons';
import Qs from 'query-string';
import { formToValues } from '@utils/transformer/fieldValue';
const searchForm = createForm({
  validateFirst: true,
})
const SchemaField = createSchemaField({
  components: {
    FormItem,
    Input,
    FormGrid,
    DatePicker,
    FormButtonGroup
  }
})
const Entity = () => {
  const [{ current, pageSize }, setPageConfig] = useState({ current: 1, pageSize: 10 });
  const [sorter, setSorter] = useState();
  const [searchVisible, { toggle }] = useToggle(false);
  const [selectedRowKeys, setSelectedRowKeys] = useState([]);
  const [selectedRows, setSelectedRows] = useState([]);
  const loadQuery = useRequest((params = {}) =>
    request('/api/entity', {
      params: {
        current,
        pageSize,
        sort: sorter?.field,
        order: sorter?.order?.slice(0, -3),
        ...params
      },
      paramsSerializer: (params) {
        return Qs.stringify(params, { arrayFormat: 'comma', skipEmptyString: true, skipNull: true })
      },
    }), {
    refreshDeps: [current, pageSize, sorter]
  });
  const deleteQuery = useRequest((ids) =>
    request('/api/entity/${ids[0]}', {
      method: 'DELETE',
      data: ids
    }), {
    manual: true,
    onSuccess: (data) {
      message.success(data.message);
      loadQuery.refresh();
    },
    formatResult: (res) {
      return res;
    }
  });
  const deleteRecords = (records) => {
    Modal.confirm({
      title: '确定删除以下记录吗?',
      icon: <ExclamationCircleOutlined />,
      content: (
        <List
          bordered
          dataSource={records.map(record => record.name)}
          renderItem={item => (
            <List.Item>{item}</List.Item>
          )}
        />
      ),
      okText: '是',
      cancelText: '否',
      okType: 'danger',
      onOk: () {
        return deleteQuery.run(records.map(record => record.id))
      }
    });
  }
  const rowSelection = {
    selectedRowKeys,
    onChange: (selectedRowKeys, selectedRows) => {
      setSelectedRowKeys(selectedRowKeys);
      setSelectedRows(selectedRows);
    }
  }
  const handleSubmit = (values) => {
    loadQuery.run(formToValues(values));
  }
  const columns = [
    { title: 'ID', dataIndex: 'id', key: 'id', sorter: true },
    { title: '名称', dataIndex: 'name', key: 'name' },
    { title: '标题', dataIndex: 'title', key: 'title' },
    {
      title: '操作', dataIndex: 'operations', key: 'operations', render: (_, record) => (
        <Space>
        <Button
          type="default"
          onClick={() => history.push(`/entity/edit?id=${record.id}`)}
        >编辑Button</Button>
        <Button
          type="primary"
          onClick={() => deleteRecords([record])}
        >删除Button</Button>
        <Space>

```

```

    )
  }
}
return (
  <PageContainer>
    {
      searchVisible && (
        <Card key="search">
          <Form
            layout="inline"
            form={searchForm}
            onAutoSubmit={handleSubmit}
          >
            <SchemaField>
              <SchemaField.Void
                x-component="FormGrid"
                x-component-props={{
                  maxColumns: 4,
                  minColumns: 2,
                }}
              >
                <SchemaField.String
                  name="name"
                  title="名称"
                  x-decorator="FormItem"
                  x-component="Input"
                />
                <SchemaField.String
                  name="title"
                  title="标题"
                  x-decorator="FormItem"
                  x-component="Input"
                />
                <SchemaField.String
                  name="created"
                  title="创建时间"
                  x-decorator="FormItem"
                  x-decorator-props={{ gridSpan: 2 }}
                  x-component="DatePicker.RangePicker"
                  x-component-props={{
                    showTime: true, ranges: {
                      '今天': [moment().startOf('day'), moment()],
                      '本月': [moment().startOf('month'), moment().endOf('month')],
                      '上周': [moment().subtract(7, 'days'), moment()],
                      '上月': [moment().subtract(1, 'months').startOf('month'), moment().subtract(1, 'months').endOf('month')]
                    }
                  }}
                />
              <SchemaField.Void>
                <SchemaField>
                  <FormButtonGroup.FormItem>
                    <Submit>
                      查询
                    </Submit>
                    <Reset >
                      重置
                    </Reset>
                  </FormButtonGroup.FormItem>
                </Form>
              </SchemaField>
            </Card>
          )
        </Card>
        <Row>
          <Col xs={24} style={{ textAlign: 'right', padding: '10px' }}>
            <Space>
              <Button
                shape='circle'
                icon={<SearchOutlined />}
                onClick={toggle}
                type={searchVisible ? 'primary' : 'default'}
              />
              <Button
                type="primary"
                onClick={() => history.push('/entity/edit')}
              >添加Button</Button>
            </Space>
          </Col>
        </Row>
        <Table
          dataSource={loadQuery.data?.list}
          columns={columns}
          loading={loadQuery.loading}
          rowKey={row => row.id}
          pagination={false}
          onChange={(_, __, sorter) => setSorter(sorter)}
          rowSelection={rowSelection}
        />
        <Row>
          <Col xs={24} style={{ textAlign: 'right', padding: '10px' }}>
            <Pagination
              total={loadQuery.data?.total || 0}
              current={loadQuery.data?.current || 1}
              pageSize={loadQuery.data?.pageSize || 10}
              showSizeChanger
              showQuickJumper
              showTotal={total => `总计${total}条`}
              onChange={current, pageSize => setPageConfig({ current, pageSize })}
            />
          </Col>
        </Row>
      </Card>
      {
        selectedRowKeys.length > 0 && (

```

```
        <FooterToolbar extra={
          <Space>
            <Button
              type="primary"
              onClick={() => deleteRecords(selectedRows)}
            >删除Button<
            <Space>
          </>
        >
      </>
    )
  }
  <PageContainer>
    <
  </>
}
export default Entity
```

src\utils\transformer\fieldValue.tsx

```
import moment from 'moment';
export function formToValues(values) {
  let result = Array.isArray(values) ? [] : {};
  for (let key in values) {
    let value = values[key];
    if (typeof value === 'boolean') {
      value = value ? 1 : 0;
    } else if (moment.isMoment(value)) {
      value = value.format();
    } else if (Array.isArray(value)) {
      value = formToValues(value)
    }
    result[key] = value;
  }
  return result;
}
```

5.添加修改实体

.umirc.ts

```
+ {
+   name: '实体编辑',
+   path: '/entity/edit',
+   component: './EntityEdit',
+ }
```

src\constants\enums.ts

```

export enum FIELD_TYPES {
  text = 'text',
  number = 'number',
  select = 'select',
  datetime = 'datetime',
  switch = 'switch'
}

export enum BUTTON_ACTION_TYPES {
  add = 'add',
  update = 'update',
  delete = 'delete',
  refresh = 'refresh'
}

export enum BUTTON_TYPES {
  default = 'default',
  primary = 'primary',
}

export enum METHOD_TYPES {
  DELETE = 'DELETE',
  POST = 'POST',
  PUT = 'PUT',
  GET = 'GET'
}

export const FIELD = [
  { label: '文本', value: FIELD_TYPES.text },
  { label: '数字', value: FIELD_TYPES.number },
  { label: '下拉框', value: FIELD_TYPES.select },
  { label: '日期时间', value: FIELD_TYPES.datetime },
  { label: '开关', value: FIELD_TYPES.switch }
]

export const BUTTON = [
  { label: '默认', value: BUTTON_TYPES.default },
  { label: '主要', value: BUTTON_TYPES.primary }
]

export const BUTTON_ACTION = [
  { label: '添加', value: BUTTON_ACTION_TYPES.add },
  { label: '更新', value: BUTTON_ACTION_TYPES.update },
  { label: '删除', value: BUTTON_ACTION_TYPES.delete },
  { label: '刷新', value: BUTTON_ACTION_TYPES.refresh }
]

export const METHOD = [
  { label: 'DELETE', value: METHOD_TYPES.DELETE },
  { label: 'POST', value: METHOD_TYPES.POST },
  { label: 'PUT', value: METHOD_TYPES.PUT },
  { label: 'GET', value: METHOD_TYPES.GET }
]

export const BOOLEAN = [
  { label: '是', value: 1 },
  { label: '否', value: 0 }
]

```

src/pages/EntityEdit/index.tsx

```

import { Card, message, List } from 'antd'
import {
  FormItem, Input, ArrayTable, Editable, FormButtonGroup, Submit, Select, Checkbox, Switch,
  FormLayout, FormGrid,
} from '@formily/antd'
import { createForm, onFormValidateFailed } from '@formily/core'
import { FormProvider, createSchemaField } from '@formily/react'
import { onFieldValueChange, FormPath } from '@formily/core';
import { PageContainer } from '@ant-design/pro-components';
import { request, useRequest, history, useLocation } from 'umi';
import qs from 'query-string';
import { FIELD, BUTTON_ACTION, BUTTON, METHOD, BUTTON_ACTION_TYPES } from '@/constants/enums';
import { initialEntityValues } from './initialValues';
import { useMount } from 'ahooks';
const { parse } = FormPath;
const SchemaField = createSchemaField({
  components: {
    FormItem,
    Editable,
    Input,
    ArrayTable,
    Select,
    Checkbox,
    Switch,
    FormLayout,
    Card,
    FormGrid
  },
},
))

const form = createForm({
  effects(form) {
    onFieldValueChange('name', ({ value }) => {
      form.setFieldState(`${(page?.data,record?.data)}`, (state: any) => {
        let dataValue = form.getValuesIn(state.address.entropy);
        for (let item of dataValue) {
          if (item.name === 'url') {
            let action = form.getValuesIn(parse('.action', state.address.entropy).entropy);
            item.value = getUrl(action, value)
          }
        }
        function getUrl(action, value) {

```

```

        switch (action) {
          case BUTTON_ACTION_TYPES.add:
            return `/api/${value}`;
          case BUTTON_ACTION_TYPES.update:
            return `/api/${value}/:id`;
          case BUTTON_ACTION_TYPES.delete:
            return `/api/${value}/:id`;
          default:
            break;
        }
      }
    });
  });
  onFormValidateFailed((form) => {
    message.error(<List
      bordered
      dataSource={form.errors.map(error => `${error.address} ${error.messages.join(',')}')}`
      renderItem={(item: any) => (
        <List.Item>{item}List.Item</List.Item>
      )}
    />);
  });
}
))

export default function () {
  const location = useLocation();
  const query = Qs.parse(location.search);
  const loadQuery = useRequest((id) =>
    request(`/api/entity/${id}`, {
      method: 'GET'
    }), {
    manual: true,
    onSuccess(data: any) {
      form.setValues(data);
    }
  });
  useMount(() => {
    if (query.id) {
      loadQuery.run(query.id);
    } else {
      form.setValues(initialEntityValues);
    }
  });
  const addQuery = useRequest((values) =>
    request(`/api/entity`, {
      method: 'POST',
      data: values
    }), {
    manual: true,
    onSuccess() {
      history.back();
    },
    formatResult(res) {
      return res;
    }
  });
  const updateQuery = useRequest((id, values) =>
    request(`/api/entity/${id}`, {
      method: 'PUT',
      data: values
    }), {
    manual: true,
    onSuccess() {
      history.back();
    },
    formatResult(res) {
      return res;
    }
  });
  const handleSubmit = (values) => {
    console.log('values', values);
    if (query.id) {
      updateQuery.run(query.id, values);
    } else {
      addQuery.run(values);
    }
  }

  return (
    <PageContainer>
      <FormProvider form={form}>
        <SchemaField>
          <SchemaField.Void
            x-component="FormGrid"
            x-component-props={{ maxColumns: 4, minColumns: 2 }}
          >
            <SchemaField.String
              name="name"
              title="名称"
              x-decorator="FormItem"
              required
              x-component="Input"
            />
            <SchemaField.String
              name="title"
              title="标题"
              x-decorator="FormItem"
              required
              x-component="Input"
            />
            <SchemaField.Void>
              <SchemaField.Array
                name="fields"

```

```

x-decorator="FormItem"
x-component="ArrayTable"
>
<SchemaField.Object>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ width: 50, title: '排序', align: 'center' }}
  >
    <SchemaField.Void
      x-decorator="FormItem"
      required
      x-component="ArrayTable.SortHandle"
    />
  </SchemaField.Void>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ width: 80, title: '索引', align: 'center' }}
  >
    <SchemaField.Void
      x-decorator="FormItem"
      required
      x-component="ArrayTable.Index"
    />
  </SchemaField.Void>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ title: '名称', dataIndex: 'name', width: 200 }}
  >
    <SchemaField.String
      name="name"
      x-decorator="FormItem"
      required
      x-component="Input"
    />
  </SchemaField.Void>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ title: '标题', width: 200 }}
  >
    <SchemaField.String
      x-decorator="FormItem"
      name="title"
      required
      x-component="Input"
    />
  </SchemaField.Void>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ title: '类型', width: 200 }}
  >
    <SchemaField.String
      x-decorator="FormItem"
      name="type"
      required
      x-component="Select"
      enum={FIELD}
    />
  </SchemaField.Void>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ title: '数据', width: 200 }}
  >
    <SchemaField.Array
      name="data"
      title="配置字段数据"
      x-decorator="Editable.Popover"
      x-component="ArrayTable"
      x-reactions={({ field: any }) => {
        let value = field.getState().value;
        if (value.length > 0) {
          field.title = value.map(item => item.title).join(',');
        }
      }}
    />
  </SchemaField.Void>
  <SchemaField.Object>
    <SchemaField.Void
      x-component="ArrayTable.Column"
      x-component-props={{ title: '标题', dataIndex: 'name', width: 200 }}
    >
      <SchemaField.String
        name="title"
        x-decorator="FormItem"
        required
        x-component="Input"
      />
    </SchemaField.Void>
    <SchemaField.Void
      x-component="ArrayTable.Column"
      x-component-props={{ title: '值', dataIndex: 'value', width: 200 }}
    >
      <SchemaField.String
        name="value"
        x-decorator="FormItem"
        required
        x-component="Input"
      />
    </SchemaField.Void>
    <SchemaField.Void
      x-component="ArrayTable.Column"
      x-component-props={{
        title: '操作',
        dataIndex: 'operations',
        width: 200,
        fixed: 'right'
      }}
    >

```

```

    })
  >
  <SchemaField.Void x-component="FormItem">
    <SchemaField.Void x-component="ArrayTable.Remove" />
    <SchemaField.Void x-component="ArrayTable.MoveDown" />
    <SchemaField.Void x-component="ArrayTable.MoveUp" />
    SchemaField.Void>
  SchemaField.Void>
  SchemaField.Object>
  <SchemaField.Void
    x-component="ArrayTable.Addition"
    title="添加字段配置"
  />
  SchemaField.Array>
  SchemaField.Void>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ title: '支持排序', width: 200 }}
  >
  <SchemaField.String
    x-decorator="FormItem"
    name="sorter"
    x-component="Switch"
    x-reactions={{
      "dependencies": [".type"],
      when: "{{ $deps[0] === 'number' }}",
      fulfill: {
        state: {
          value: true
        }
      },
      otherwise: {
        state: {
          value: false
        }
      }
    }},
  />
  SchemaField.Void>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ title: '列表隐藏', width: 200 }}
  >
  <SchemaField.String
    x-decorator="FormItem"
    name="hideInColumn"
    x-component="Switch"
  />
  SchemaField.Void>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ title: '不允许编辑', width: 200 }}
  >
  <SchemaField.String
    x-decorator="FormItem"
    name="disabled"
    x-component="Switch"
  />
  SchemaField.Void>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{
      title: '操作',
      dataIndex: 'operations',
      width: 200,
      fixed: 'right'
    }}
  >
  <SchemaField.Void x-component="FormItem">
    <SchemaField.Void x-component="ArrayTable.Remove" />
    <SchemaField.Void x-component="ArrayTable.MoveDown" />
    <SchemaField.Void x-component="ArrayTable.MoveUp" />
    SchemaField.Void>
  SchemaField.Void>
  SchemaField.Object>
  <SchemaField.Void
    x-component="ArrayTable.Addition"
    title="添加字段"
  />
  SchemaField.Array>
  <SchemaField.Array
    name="page"
    x-decorator="FormItem"
    x-component="ArrayTable"
  >
  <SchemaField.Object>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ width: 50, title: '排序', align: 'center' }}
  >
  <SchemaField.Void
    x-decorator="FormItem"
    required
    x-component="ArrayTable.SortHandle"
  />
  SchemaField.Void>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ width: 80, title: '索引', align: 'center' }}
  >
  <SchemaField.Void
    x-decorator="FormItem"
    required
    x-component="ArrayTable.Index"
  >

```



```

/>
SchemaField.Void>
<SchemaField.Void
  x-component="ArrayTable.Column"
  x-component-props={{ title: '文本', dataIndex: 'title', width: 200 }}
>
  <SchemaField.String
    name="title"
    x-decorator="Editable"
    required
    x-component="Input"
  />
SchemaField.Void>
<SchemaField.Void
  x-component="ArrayTable.Column"
  x-component-props={{ title: '类型', width: 200 }}
>
  <SchemaField.String
    x-decorator="FormItem"
    name="type"
    required
    x-component="Select"
    enum={BUTTON}
  />
SchemaField.Void>
<SchemaField.Void
  x-component="ArrayTable.Column"
  x-component-props={{ title: '操作', width: 200 }}
>
  <SchemaField.String
    x-decorator="FormItem"
    name="action"
    required
    x-component="Select"
    enum={BUTTON_ACTION}
  />
SchemaField.Void>
<SchemaField.Void
  x-component="ArrayTable.Column"
  x-component-props={{ title: '地址', dataIndex: 'url', width: 200 }}
>
  <SchemaField.Array
    name="data"
    title="配置按钮数据"
    x-decorator="Editable.Popover"
    x-component="ArrayTable"
    x-reactions={(field: any) => {
      let value = field.getState().value;
      if (value.length > 0) {
        field.title = '{ ' + value.map(item => item.name).join(',') + ' }';
      }
    }}
  >
  <SchemaField.Object>
    <SchemaField.Void
      x-component="ArrayTable.Column"
      x-component-props={{ title: '属性1', dataIndex: 'name', width: 200 }}
    >
    <SchemaField.String
      name="name"
      x-decorator="FormItem"
      required
      x-component="Input"
    />
  SchemaField.Void>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ title: '值', dataIndex: 'value', width: 200 }}
  >
  <SchemaField.String
    name="value"
    x-decorator="FormItem"
    required
    x-component="Input"
    x-reactions={{
      dependencies: ['.name'],
      when: "({$deps[0] === 'method'})",
      fulfill: {
        schema: {
          'x-component': "Select",
          enum: METHOD
        }
      },
      otherwise: {
        schema: {
          'x-component': "Input",
          enum: null
        }
      },
    }}
  />
SchemaField.Void>
<SchemaField.Void
  x-component="ArrayTable.Column"
  x-component-props={{
    title: '操作',
    dataIndex: 'operations',
    width: 200,
    fixed: 'right'
  }}
>
  <SchemaField.Void x-component="FormItem">
    <SchemaField.Void x-component="ArrayTable.Remove" />
    <SchemaField.Void x-component="ArrayTable.MoveDown" />
  </SchemaField.Void>
</SchemaField.Void>

```

```

        <SchemaField.Void x-component="ArrayTable.MoveUp" />
        SchemaField.Void>
        SchemaField.Void>
        SchemaField.Object>
        <SchemaField.Void
            x-component="ArrayTable.Addition"
            title="添加配置"
        />
        SchemaField.Array>
    SchemaField.Void>
    <SchemaField.Void
        x-component="ArrayTable.Column"
        x-component-props={{
            title: '操作',
            dataIndex: 'operations',
            width: 200,
            fixed: 'right'
        }}
    >
    <SchemaField.Void x-component="FormItem">
        <SchemaField.Void x-component="ArrayTable.Remove" />
        <SchemaField.Void x-component="ArrayTable.MoveDown" />
        <SchemaField.Void x-component="ArrayTable.MoveUp" />
        SchemaField.Void>
        SchemaField.Void>
        SchemaField.Object>
        <SchemaField.Void
            x-component="ArrayTable.Addition"
            title="添加页面操作"
        />
        SchemaField.Array>
    <SchemaField.Array
        name="record"
        x-decorator="FormItem"
        x-component="ArrayTable"
    >
    <SchemaField.Object>
        <SchemaField.Void
            x-component="ArrayTable.Column"
            x-component-props={{ width: 50, title: '排序', align: 'center' }}
        >
        <SchemaField.Void
            x-decorator="FormItem"
            required
            x-component="ArrayTable.SortHandle"
        />
        SchemaField.Void>
        <SchemaField.Void
            x-component="ArrayTable.Column"
            x-component-props={{ width: 80, title: '索引', align: 'center' }}
        >
        <SchemaField.Void
            x-decorator="FormItem"
            required
            x-component="ArrayTable.Index"
        />
        SchemaField.Void>
        <SchemaField.Void
            x-component="ArrayTable.Column"
            x-component-props={{ title: '文本', dataIndex: 'title', width: 200 }}
        >
        <SchemaField.String
            name="title"
            x-decorator="Editable"
            required
            x-component="Input"
        />
        SchemaField.Void>
        <SchemaField.Void
            x-component="ArrayTable.Column"
            x-component-props={{ title: '类型', width: 200 }}
        >
        <SchemaField.String
            x-decorator="FormItem"
            name="type"
            required
            x-component="Select"
            enum={BUTTON}
        />
        SchemaField.Void>
        <SchemaField.Void
            x-component="ArrayTable.Column"
            x-component-props={{ title: '操作', width: 200 }}
        >
        <SchemaField.String
            x-decorator="FormItem"
            name="action"
            required
            x-component="Select"
            enum={BUTTON_ACTION}
        />
        SchemaField.Void>
        <SchemaField.Void
            x-component="ArrayTable.Column"
            x-component-props={{ title: '数据', dataIndex: 'data', width: 200 }}
        >
        <SchemaField.Array
            name="data"
            title="配置数据"
            x-decorator="Editable.Popover"
            x-component="ArrayTable"
            x-reactions={(field: any) => {
                let value = field.getState().value;

```

```

        if (value.length > 0) {
            field.title = '{ ' + value.map(item => item.name).join(', ') + ' }';
        }
    })
}
>
<SchemaField.Object>
  <SchemaField.Void
    x-component="ArrayTable.Column"
    x-component-props={{ title: '属性', dataIndex: 'name', width: 200 }}
  >
    <SchemaField.String
      name="name"
      x-decorator="FormItem"
      required
      x-component="Input"
    />
  <SchemaField.Void>
    <SchemaField.Void
      x-component="ArrayTable.Column"
      x-component-props={{ title: '值', dataIndex: 'value', width: 200 }}
    >
      <SchemaField.String
        name="value"
        x-decorator="FormItem"
        required
        x-component="Input"
      />
    <SchemaField.Void>
      <SchemaField.Void
        x-component="ArrayTable.Column"
        x-component-props={{
          title: '操作',
          dataIndex: 'operations',
          width: 200,
          fixed: 'right'
        }}
      >
        <SchemaField.Void x-component="FormItem">
          <SchemaField.Void x-component="ArrayTable.Remove" />
          <SchemaField.Void x-component="ArrayTable.MoveDown" />
          <SchemaField.Void x-component="ArrayTable.MoveUp" />
        </SchemaField.Void>
      </SchemaField.Void>
    </SchemaField.Void>
  </SchemaField.Void>
  <SchemaField.Object>
    <SchemaField.Void
      x-component="ArrayTable.Addition"
      title="添加配置"
    />
  </SchemaField.Void>
  <SchemaField.Void>
    <SchemaField.Void
      x-component="ArrayTable.Column"
      x-component-props={{
        title: '操作',
        dataIndex: 'operations',
        width: 200,
        fixed: 'right'
      }}
    >
      <SchemaField.Void x-component="FormItem">
        <SchemaField.Void x-component="ArrayTable.Remove" />
        <SchemaField.Void x-component="ArrayTable.MoveDown" />
        <SchemaField.Void x-component="ArrayTable.MoveUp" />
      </SchemaField.Void>
    </SchemaField.Void>
  </SchemaField.Void>
  <SchemaField.Void>
    <SchemaField.Void
      x-component="ArrayTable.Addition"
      title="添加记录操作"
    />
  </SchemaField.Void>
  <SchemaField.Void>
    <FormButtonGroup>
      <Submit onSubmit={handleSubmit}>提交Submit</Submit>
    </FormButtonGroup>
  </SchemaField.Void>
  <FormProvider>
    <PageContainer>
      </PageContainer>
    </FormProvider>
  </SchemaField.Void>
}
}

```

src/pages/EntityEdit/initialValues.ts

```
import { FIELD_TYPES, BUTTON_ACTION_TYPES, BUTTON_TYPES, METHOD_TYPES } from '@/constants/enums';
export const initialEntityValues = {
  name: 'user',
  title: '用户',
  fields: [
    {
      disabled: 0,
      hideInColumn: 0,
      name: "name",
      sorter: 0,
      title: "名称",
      type: FIELD_TYPES.text
    }
  ],
  page: [
    {
      action: BUTTON_ACTION_TYPES.add,
      title: "添加",
      type: BUTTON_TYPES.primary,
      data: [
        { name: 'url', value: '/api/user' },
        { name: 'method', value: METHOD_TYPES.POST }
      ]
    }
  ],
  record: [
    {
      action: BUTTON_ACTION_TYPES.update,
      title: "更新",
      type: BUTTON_TYPES.default,
      data: [
        { name: 'url', value: '/api/user/:id' },
        { name: 'method', value: METHOD_TYPES.PUT }
      ]
    },
    {
      action: BUTTON_ACTION_TYPES.delete,
      title: "删除",
      type: BUTTON_TYPES.primary,
      data: [
        { name: 'url', value: '/api/user/:id' },
        { name: 'method', value: METHOD_TYPES.DELETE }
      ]
    }
  ]
}
}
```

## 6.查看实体

.umirc.ts

```
+ {
+   name: '实体',
+   path: '/entity/view',
+   component: './EntityView',
+ }
```

src/pages/EntityView/index.tsx

```
import { useState } from 'react';
import { Table, Card, Row, Col, Pagination, Space, Modal, List, message, Button } from 'antd';
import { createForm } from '@formily/core';
import { Form, Submit, Reset, FormButtonGroup } from '@formily/antd';
import { FooterToolbar, PageContainer } from '@ant-design/pro-components';
import { request, useRequest, history, useLocation } from 'umi';
import { useToggle, useMount, useSetState } from 'ahooks';
import { ExclamationCircleOutlined, SearchOutlined } from '@ant-design/icons';
import Qs from 'query-string';
import EditModal from '@/components/EditModal';
import { renderColumns } from '@/utils/render/column';
import { BUTTON_ACTION_TYPES } from '@/constants/enums';
import { formToValues } from '@/utils/transformer/fieldValue';
import { renderOperations } from '@/utils/render/operation';
import { renderSearchFields } from '@/utils/render/field';
const searchForm = createForm({
  validateFirst: true,
})
const Entity = () => {
  const [{ current, pageSize }, setPageConfig] = useState({ current: 1, pageSize: 10 });
  const [sorter, setSorter] = useState();
  const [searchVisible, { toggle }] = useToggle(false);
  const [selectedRowKeys, setSelectedRowKeys] = useState([]);
  const [selectedRows, setSelectedRows] = useState([]);
  const location = useLocation();
  const query = Qs.parse(location.search);
  const [modalState, setModalState] = useSetState({ visible: false, title: '', data: [], row: {} });
  const loadQuery = useRequest({ params = {} } =>
    request(`/api/${query.name}`, {
      params: {
        current,
        pageSize,
        sort: sorter?.field,
        order: sorter?.order?.slice(0, -3),
        ...params
      },
    }, {
      paramsSerializer(params) {
        return Qs.stringify(params, { arrayFormat: 'comma', skipEmptyString: true, skipNull: true })
      },
    }, {
      refreshDeps: [current, pageSize, sorter]
    });
  useMount(() => {
    if (query.id) {
      loadQuery.run();
    }
  });
}
```

```

    }
  });
  const deleteQuery = useRequest((ids) => {
    request(`/api/${query.name}/${ids[0]}`, {
      method: 'DELETE',
      data: ids
    }), {
      manual: true,
      onSuccess(data) {
        message.success(data.message);
        loadQuery.refresh();
      },
      formatResult(res) {
        return res;
      }
    });
  const deleteRecords = (records) => {
    Modal.confirm({
      title: '确定删除以下记录?',
      icon: <ExclamationCircleOutlined />,
      content: (
        <List
          bordered
          dataSource={records.map(record => record.name)}
          renderItem={item => {
            <List.Item>{item}</List.Item>
          }}
        />
      ),
      okText: '是',
      cancelText: '否',
      okType: 'danger',
      onOk() {
        return deleteQuery.run(records.map(record => record.id))
      }
    });
  }
  const rowSelection = {
    selectedRowKeys,
    onChange: (selectedRowKeys, selectedRows) => {
      setSelectedRowKeys(selectedRowKeys);
      setSelectedRows(selectedRows);
    }
  }
  const handleSubmit = (values) => {
    loadQuery.run(formToValues(values));
  }

  const onAction = (operation, row) => {
    switch (operation.action) {
      case BUTTON_ACTION_TYPES.add:
      case BUTTON_ACTION_TYPES.update:
        setModalState({ visible: true, title: operation.title, data: operation.data, row })
        break;
      case BUTTON_ACTION_TYPES.delete:
        deleteRecords([row]);
        break;
      case BUTTON_ACTION_TYPES.refresh:
        loadQuery.refresh();
        break;
      default:
        break;
    }
  }
  return (
    <PageContainer>
      {
        searchVisible && (
          <Card key="search">
            <Form
              layout="inline"
              form={searchForm}
              onAutoSubmit={handleSubmit}
            >
              {renderSearchFields(loadQuery.data?.fields)}
            </Form>
            <FormButtonGroup>
              <FormButtonGroup.FormItem>
                <Submit>
                  查询
                </Submit>
                <FormButtonGroup.FormItem>
                <Reset>
                  重置
                </Reset>
              </FormButtonGroup>
            </FormButtonGroup>
          </Card>
        )
      }
    </PageContainer>
  )
  <Card>
    <Row>
      <Col xs={24} style={{ textAlign: 'right', padding: '10px' }}>
        <Space>
          <Button
            shape="circle"
            icon={searchIcon}
            onClick={toggle}
            type={searchVisible ? 'primary' : 'default'}
          />
          {renderOperations(loadQuery.data?.page, onAction)}
        </Space>
      </Col>
    </Row>
    <Table>
      {renderTable(loadQuery.data?.list, onAction)}
    </Table>
  </Card>

```

```

        loading={loadQuery.loading}
        rowKey={row => row.id}
        pagination={false}
        onChange={(_, __, sorter) => setSorter(sorter)}
        rowSelection={rowSelection}
      />
</Row>
<Col xs={24} style={{ textAlign: 'right', padding: '10px' }}>
  <Pagination
    total={loadQuery.data?.total || 0}
    current={loadQuery.data?.current || 1}
    pageSize={loadQuery.data?.pageSize || 10}
    showSizeChanger
    showQuickJumper
    showTotal={total => `总计${total}条`}
    onChange={current, pageSize => setPageConfig({ current, pageSize })}
  />
</Col>
</Row>
<Card>
  <EditModal
    title={modalState.title}
    visible={modalState.visible}
    data={modalState.data}
    row={modalState.row}
    fields={loadQuery.data?.fields}
    onOk={() => {
      setModalState({ visible: false });
      loadQuery.refresh();
    }}
    onCancel={() => setModalState({ visible: false })}
  />
  {
    selectedRowKeys.length > 0 && (
      <FooterToolbar extra={
        <Space>
          <Button
            type="primary"
            onClick={() => deleteRecords(selectedRows)}
          >删除</Button>
        </Space>
      } />
    )
  }
</PageContainer>
)
}
export default Entity

```

src/components/EditModal/index.tsx

```

import { message, Modal } from 'antd'
import { createForm } from '@formily/core'
import { FormProvider } from '@formily/react'
import { request, useRequest } from 'umi';
import { renderFormFields } from '@/utils/render/field';
import { useEffect } from 'react';
const form = createForm({
  validateFirst: true,
})
export default function ({ title, visible, onOk, onCancel, row, data = [], fields = [] }) {
  const options = any = data.reduce(memo, item) => {
    let value = item.value;
    if (Object.keys(row).length > 0 && item.name === 'url') {
      value = value.replace(/:([^\s]+)/g, (match, key) => row[key]);
    }
    memo[item.name] = value;
    return memo
  }, {});
  const actionQuery = useRequest(({ method, url, data }) => request(url, { method, data }));
  {
    manual: true,
    onSuccess(data) {
      message.success(data.message);
      onOk();
    },
    formatResult(res) {
      return res;
    }
  });
  useEffect(() => {
    if (visible && row) {
      form.setValues(row);
    }
  });
  return (
    <Modal
      title={title}
      visible={visible}
      onOk={() => {
        actionQuery.run({ url: options.url, method: options.method, data: form.values });
      }}
      onCancel={onCancel}
      destroyOnClose
      maskClosable={false}
      forceRender
    >
      <FormProvider form={form}>
        {renderFormFields(fields)}
      </FormProvider>
    </Modal>
  )
}

```

src\utils\renderColumn.tsx

```
import moment from 'moment';
import { Tag, Space } from 'antd';
import { FIELD_TYPES } from '@constants/enums';
import { renderOperations } from './operation';
const renderColumn = (column) => {
  switch (column.type) {
    case FIELD_TYPES.datetime:
      column.render = (value) => moment(value).format('YYYY-MM-DD HH:mm:ss');
      break;
    case FIELD_TYPES.switch:
      column.render = (value) => {
        const { title } = column.data.find(item => item.value === value);
        return <Tag color={value ? 'green' : 'red'}> {title} Tag</Tag>
      }
      break;
    default:
      break;
  }
}

export function renderColumns(data, onAction) {
  if (data) {
    const { fields, record } = data;
    return fields
      .filter(field => !field.hideInColumn)
      .map(field => {
        const column = { ...field };
        column.key = column.dataIndex = field.name;
        renderColumn(column);
        return column;
      }).concat([
        {
          title: '操作', dataIndex: 'operations', key: 'operations', render: (_, row) => (
            <Space>
              {renderOperations(record, onAction, row)}
            </Space>
          )
        }
      ]);
  }
  return [];
}
```

src\utils\renderfield.tsx

```
import { FIELD_TYPES } from '@constants/enums';
import moment from 'moment';
import { Card, InputNumber } from 'antd';
import {
  FormItem, Input, ArrayTable, Editable, Select, Checkbox, Switch,
  FormLayout, FormGrid
} from '@formily/antd';
import { createSchemaField } from '@formily/react';
const SchemaField = createSchemaField({
  components: {
    FormItem,
    Editable,
    Input,
    ArrayTable,
    Select,
    Checkbox,
    Switch,
    FormLayout,
    Card,
    FormGrid,
    InputNumber
  },
});

export const renderFormFields = (fields = []) => {
  return (
    <SchemaField>
      {
        fields.map(field => {
          switch (field.type) {
            case FIELD_TYPES.datetime:
              return (
                <SchemaField.String
                  name={field.name}
                  title={field.title}
                  x-decorator="FormItem"
                  required
                  x-component="DatePicker"
                  x-component-props={{ showTime: true }}
                />
              )
            case FIELD_TYPES.switch:
              return (
                <SchemaField.String
                  name={field.name}
                  title={field.title}
                  x-decorator="FormItem"
                  required
                  x-component="Switch"
                />
              )
            case FIELD_TYPES.number:
              return (
                <SchemaField.String
                  name={field.name}
                  title={field.title}
                  x-decorator="FormItem"
                  required
                  x-component="InputNumber"
                />
              )
          }
        })
      }
    </SchemaField>
  );
}
```

```

    );
    default:
      return (
        <SchemaField.String
          name={field.name}
          title={field.title}
          x-decorator="FormItem"
          required
          x-component="Input"
        />
      )
    }
  })
}
</SchemaField>
)
}
}

export const renderSearchFields = (fields = []) => {
  return (
    <SchemaField>
      <SchemaField.Void
        x-component="FormGrid"
        x-component-props={{
          maxColumns: 4,
          minColumns: 2,
        }}
      >
        {
          fields.map(field => {
            switch (field.type) {
              case 'number':
                return (
                  <SchemaField.String
                    name={field.name}
                    title={field.title}
                    x-decorator="FormItem"
                    x-component="InputNumber"
                  />
                )
              case 'datetime':
                return (
                  <SchemaField.String
                    name={field.name}
                    title={field.title}
                    x-decorator="FormItem"
                    x-decorator-props={{ gridSpan: 2 }}
                    x-component="DatePicker.RangePicker"
                    x-component-props={{
                      showTime: true, ranges: {
                        '今天': [moment().startOf('day'), moment()],
                        '本月': [moment().startOf('month'), moment().endOf('month')],
                        '上周': [moment().subtract(7, 'days'), moment()],
                        '上月': [moment().subtract(1, 'months').startOf('month'), moment().subtract(1, 'months').endOf('month')]
                      }
                    }}
                  />
                )
              default:
                return (
                  <SchemaField.String
                    name={field.name}
                    title={field.title}
                    x-decorator="FormItem"
                    x-component="Input"
                  />
                )
            }
          })
        }
      </SchemaField.Void>
    </SchemaField>
  )
}
}

```

src/utils/renderOperation.tsx

```

import { Space, Button } from 'antd';
export const renderOperations = (record = [], onAction, row = {}) => {
  return (
    <Space>
      {
        record.map(operation => {
          <Button
            key={operation.title}
            type={operation.type}
            onClick={() => onAction(operation, row)}
          >{operation.title}</Button>
        })
      }
    </Space>
  )
}

```

参考