初始化文档结构、终端快捷使用、通过sshkey关联远程仓库并上传、创建登录分支、使用layui框架快速开发界面、添加表单验证，调用注册和登录接口、ajaxPrefilter、将项目登录分支提交到git、快速开发index后台页面、互斥效果以及默认选中、iframe标签使用、清除layui的默认a标签动画、文字头像、ajax请求头、控制用户访问权限、创建user分支、个人中心基本资料面板、将用户信息获取放到baseapi内、隐藏域、调取父页面的方法、重置密码校验规则、cropper的基本使用、文件选择以及图片更换、上传图片至服务器、base64格式、裁剪框的优化、user分支上传且合并、express插件使用、使用模板引擎渲染表格、弹出层绘制内容、获取并将id渲染到属性、定义查询对象、时间格式过滤器、下拉选择框、初始化下拉选择框、实现筛选、分页、分页结构、删除文章页码逻辑、富文本编辑器、封面裁剪、判断发布还是草稿、Jqformdata文件上传、

接口文档：<https://www.showdoc.com.cn/escook/3707372986195375>

初始化文档结构：

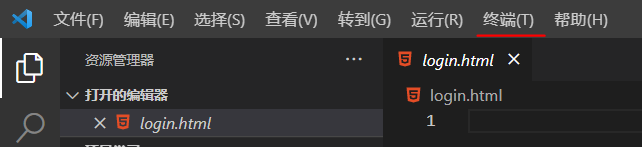
assets即配置文件夹

内存放css、js、images、lib等文件夹

js存放自建js

lib用于存放js插件等框架

终端快捷使用：



点击新建终端

快速初始化仓库

通过sshkey关联远程仓库并上传：



除了Gitbash还可以用上方的终端来进行git操作

创建登录分支：



使用layui框架快速开发界面：

layui框架：<https://layui.itze.cn/>

创建版心：



以及之后的页面绘制可以根据文档来绘制

添加表单验证：

根据文档添加

// 表单验证

  let form = layui.form;

  form.verify({

    accountVerify: function (value, item) {

      //value：表单的值、item：表单的DOM对象

      if (!new RegExp("^[a-zA-Z0-9\_\u4e00-\u9fa5·]+$").test(value)) {

        return "账号不能有特殊字符以及空格";

      }

      if (/(^\\_)|(\\_\_)|(\\_+$)/.test(value)) {

        return "账号首尾不能出现下划线'\_'";

      }

      if (/^\d+\d+\d$/.test(value)) {

        return "账号不能全为数字";

      }

      if (!/^[\S]{3,18}$/.test(value)) {

        return "账号必须3到18位以内";

      }

    },

    //我们既支持上述函数式的方式，也支持下述数组的形式

    //数组的两个值分别代表：[正则匹配、匹配不符时的提示文字]

    passwordVerify: [/^[\S]{6,12}$/, "密码必须6到12位，且不能出现空格"],

    passwordConfirmVerify: function (value) {

      let pwdvalue = $(".regContainer [name=regPassword]").val();

      if (value !== pwdvalue) {

        return "两次密码不一致";

      }

    },

  });

调用注册接口以及登录接口：

接口文档：<https://www.showdoc.com.cn/escook/3707372986195375>

其中通过登录接口会获得token串

用于授权页面登录认证，例如一些需要登录之后才能访问的页面

可以将token串存于本地

ajaxPrefilter：

每次调用$.ajax/get/post前都会调用此函数

在发送每个请求之前和由$处理之前，处理自定义Ajax选项或修改现有选项

传入配置对象，即url等参数

$. ajaxPrefilter(function(options){})

// baseApi——ajaxPrefilter,url根路径拼接

  $.ajaxPrefilter(function (options) {

    options.url = "http://www.liulongbin.top:3007" + options.url

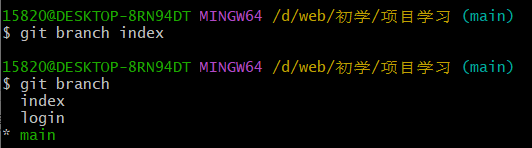
  })

当根路径变化时，那么可以随时方便更改

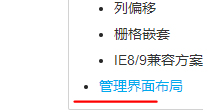
将项目登录分支提交到git：

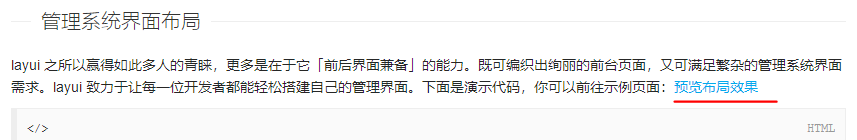
同时创建index分支





快速开发index后台页面：



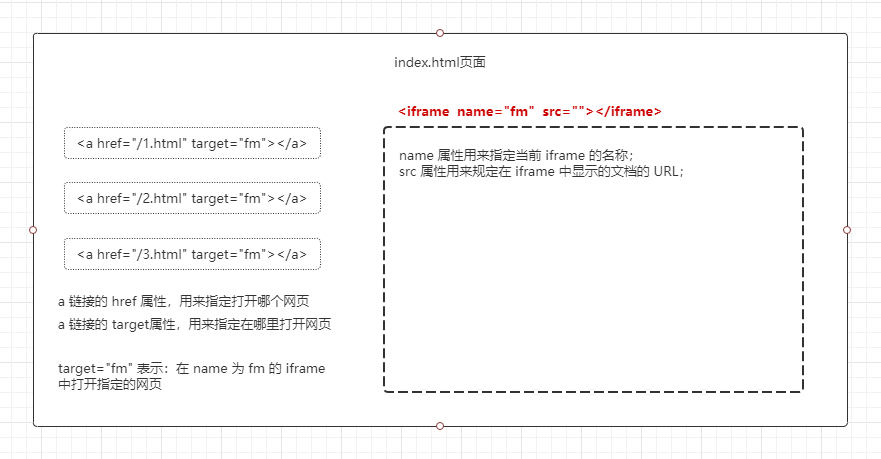
互斥效果以及默认选中：







iframe标签使用：



使用a标签的target属性会唤起对应name的iframe标签

iframe的src属性可以打开对应的url

如果iframe开始时就指定src属性，那么网页一打开就会显示对应内容

清除layui的默认a标签动画：

a {

  transition: none !important;

}

文字头像：

.textAvatar {

  display: inline-block;

  background-color: rgb(0, 150, 136);

  width: 40px;

  height: 40px;

  border-radius: 50%;

  text-align: center;

  line-height: 40px;

  margin-right: 10px;

  font-size: 16px;

  user-select: none;

}

ajax请求头：

$.ajax({

    type: "get",

    url: "/my/userinfo",

    headers: { Authorization: localStorage.getItem("token") || "" },

    success: function (res) {

      if (res.status !== 0) {

        console.log("用户暂未设置头像，自动设置文字头像");

      } else {

        console.log(res);

      }

    },

  });

可以统一请求头：

$.ajaxPrefilter(function (options) {

  options.url = "http://www.liulongbin.top:3007" + options.url;

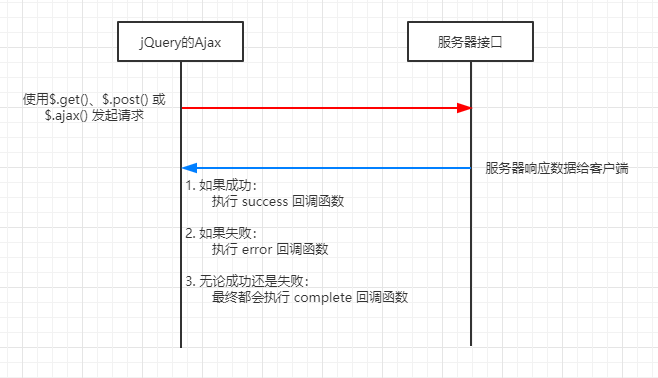
  if (options.url.indexOf("my") !== -1) {

    options.headers = { Authorization: localStorage.getItem("token") || "" };

  }

});

控制用户访问权限：



$.ajaxPrefilter(function (options) {

  options.url = "http://www.liulongbin.top:3007" + options.url;

  if (options.url.indexOf("my") !== -1) {

    options.headers = { Authorization: localStorage.getItem("token") || "" };

  }

  options.complete = function (res) {

    if (res.responseJSON.status === 1) {

      localStorage.removeItem("token");

      location.href = "./login.html";

    }

  };

});

创建user分支：



个人中心基本资料面板：

根据layui快速搭建

input标签添加readonly可以变成只读

将用户信息获取放到baseapi内：

// 获取用户基本信息

function getinfo(fun) {

  // 获取用户基本信息

  $.ajax({

    type: "get",

    url: "/my/userinfo",

    success: function (res) {

      if (res.status !== 0) {

        return layer.msg(res.message);

      } else {

        fun(res.data);

      }

    },

  });

}

表单快速赋值：

//给表单赋值

  let form = layui.form;

  form.val("baseinfoForm", {

    //formTest 即 class="layui-form" 所在元素属性 lay-filter="" 对应的值

    usernameLogined: user.username, // "name": "value"

    usernickname: user.nickname || "",

    useremail: user.email || "",

  });

隐藏域：

<input type="hidden" name="hiddenid">

调取父页面的方法：

window.parent.refreshinfo();

跨域访问需要liveserver插件支持

重置密码校验规则：

// layui表单验证

  let form = layui.form;

  form.verify({

    passwordSameVerify:function (value) {

      let oldvalue = $("#pwdForm [name=oldpwd]").val();

      if (value === oldvalue) {

        return "新密码不能与原来一致";

      }

    },

    passwordVerify: [/^[\S]{6,12}$/, "密码必须6到12位，且不能出现空格"],

    passwordConfirmVerify: function (value) {

      let pwdvalue = $("#pwdForm [name=newpwd]").val();

      if (value !== pwdvalue) {

        return "两次密码不一致";

      }

    },

  });

cropper基本用法：

引入css以及js，引入对应框架，获取裁剪区元素，指定配置对象：

$(function () {

  // 1.1 获取裁剪区域的 DOM 元素

  var $image = $("#image");

  // 1.2 配置选项

  const options = {

    // 纵横比

    aspectRatio: 1,

    // 指定预览区域

    preview: ".img-preview",

  };

  // 1.3 创建裁剪区域

  $image.cropper(options);

});

文件选择以及图片更换：

在页面中放置一个隐藏的input文件选择框

<input type="file" id="imgfile" style="display: none" />

当用户点击上传按钮时触发即可，将裁剪区的图片替换为选择的图片

指定文件类型：

<input type="file" id="imgfile" style="display: none" accept=".jpg,.png" />

图片更换前需要检测input文件框的变更状态，再通过事件对象获取文件

// 输入框变更图片事件

  $("#imgfile").on("change", function (e) {

    let filelist = e.target.files;

    if (filelist.length == 0) {

      return;

    } else {

      let imgGet = filelist[0];

      let newImgURL = URL.createObjectURL(imgGet);

      $image

        .cropper("destroy") // 销毁旧的裁剪区域

        .attr("src", newImgURL) // 重新设置图片路径

        .cropper(options); // 重新初始化裁剪区域

    }

  });

其中将文件转换成了url地址然后赋给裁剪区

上传图片至服务器：

// 将裁剪的图片上传到服务器

  $("#uploadimg").click(function () {

    let imgdataURL = $image

      .cropper("getCroppedCanvas", {

        // 创建一个 Canvas 画布

        width: 100,

        height: 100,

      })

      .toDataURL("image/png"); // 将 Canvas 画布上的内容，转化为 base64 格式的字符串;

    $.ajax({

      type: "post",

      url: "/my/update/avatar",

      data: { avatar: imgdataURL },

      success: function (res) {

        if (res.status !== 0) {

          return layer.msg(res.message);

        } else {

          layer.msg(res.message);

          window.parent.refreshinfo();

        }

      },

    });

  });

base64格式：

查看1.html

图片转base64格式网址：<https://www.css-js.com/tools/base64.html>

可以减少不必要的图片请求

但是图片字符串较长，图片体积会比原来大30%左右

故小图片比较适合，大图片不适合转成base64格式

base64优缺点：<https://www.jianshu.com/p/cf79157f7a5c>

裁剪框的优化：

将图片铺满且不可放大

更多参数参考：

<https://www.cnblogs.com/cnlisiyiii-stu/p/11570396.html>

或者参考cropper参数参考文件

默认参数：

var DEFAULTS = {

// Define the view mode of the cropper

viewMode: 0,

// 0, 1, 2, 3

// Define the dragging mode of the cropper

dragMode: DRAG\_MODE\_CROP,

// 'crop', 'move' or 'none'

// Define the initial aspect ratio of the crop box

initialAspectRatio: NaN,

// Define the aspect ratio of the crop box

aspectRatio: NaN,

// An object with the previous cropping result data

data: null,

// A selector for adding extra containers to preview

preview: '',

// Re-render the cropper when resize the window

responsive: true,

// Restore the cropped area after resize the window

restore: true,

// Check if the current image is a cross-origin image

checkCrossOrigin: true,

// Check the current image's Exif Orientation information

checkOrientation: true,

// Show the black modal

modal: true,

// Show the dashed lines for guiding

guides: true,

// Show the center indicator for guiding

center: true,

// Show the white modal to highlight the crop box

highlight: true,

// Show the grid background

background: true,

// Enable to crop the image automatically when initialize

autoCrop: true,

// Define the percentage of automatic cropping area when initializes

autoCropArea: 0.8,

// Enable to move the image

movable: true,

// Enable to rotate the image

rotatable: true,

// Enable to scale the image

scalable: true,

// Enable to zoom the image

zoomable: true,

// Enable to zoom the image by dragging touch

zoomOnTouch: true,

// Enable to zoom the image by wheeling mouse

zoomOnWheel: true,

// Define zoom ratio when zoom the image by wheeling mouse

wheelZoomRatio: 0.1,

// Enable to move the crop box

cropBoxMovable: true,

// Enable to resize the crop box

cropBoxResizable: true,

// Toggle drag mode between "crop" and "move" when click twice on the cropper

toggleDragModeOnDblclick: true,

// Size limitation

minCanvasWidth: 0,

minCanvasHeight: 0,

minCropBoxWidth: 0,

minCropBoxHeight: 0,

minContainerWidth: 200,

minContainerHeight: 100,

// Shortcuts of events

ready: null,

cropstart: null,

cropmove: null,

cropend: null,

crop: null,

zoom: null

};

user分支上传且合并：

将user分支推送到git

将user分支合并到主分支

创建article分支：



express插件使用：

安装插件后

ctrl+shift+p打开命令输入然后输入express

使用模板引擎渲染表格：

<script type="text/html" id="categoryTem">

      {{each data}}

      <tr>

        <td>{{$value.name}}</td>

        <td>{{$value.alias}}</td>

        <td>

          <button type="button" class="layui-btn layui-btn-xs">编辑</button>

          <button type="button" class="layui-btn layui-btn-xs layui-btn-danger">

            删除

          </button>

        </td>

      </tr>

      {{/each}}

    </script>

let categoryTable = template("categoryTem", res);

        $("#tb").html(categoryTable);

弹出层绘制内容：

$(".addcate").click(function () {

    layer.open({

      type: 1,

      area: ["500px", "230px"],

      title: "添加类别",

      content: $("#addcateContent").html(),

    });

  });

其中content的内容另起一个script包裹：

获取并将id渲染到属性：

<button

            type="button"

            class="layui-btn layui-btn-xs editbtn"

            data-id="{{$value.Id}}"

          >

            编辑

          </button>

        let categoryTable = template("categoryTem", res);

        $("#tb").html(categoryTable);

定义查询对象：

  let inquireObj = {

    pagenum: 1,

    pagesize: 2,

    cate\_id: "",

    state: "",

  };

时间格式过滤器：

<td>{{$value.pub\_date|dateFormat}}</td>

  template.defaults.imports.dateFormat = function (date) {

    let time = new Date(date);

    let year = time.getFullYear();

    let month = addzero(time.getMonth() + 1);

    let day = addzero(time.getDate());

    let hour = addzero(time.getHours());

    let minute = addzero(time.getMinutes());

    let second = addzero(time.getSeconds());

    return `${year} 年${month} 月${day} 日${hour} 时${minute} 分${second} 秒`;

  };

  function addzero(n) {

    n = n > 9 ? n : "0" + n;

    return n

  }

下拉选择框：

<div class="layui-form-item layui-inline">

            <select name="selectCate">

              <option value="">所有分类</option>

              <option value="010">北京</option>

              <option value="021">上海</option>

              <option value="0571">杭州</option>

            </select>

          </div>

置于form表单内

初始化下拉选择框：

function initSelect() {

    $.ajax({

      type: "get",

      url: "/my/article/cates",

      success: function (res) {

        if (res.status !== 0) {

          return layer.msg(res.message);

        } else {

          let selectlist = template("selectTem", res);

          $("#sl").html(selectlist);

          form.render();

        }

      },

    });

  }

其中调用layui的render方法重新渲染下拉选择框表单

let form = layui.form;

  initList();

  initSelect();

实现筛选：

$("#selectForm").on("submit", function (e) {

    e.preventDefault();

    inquireObj.cate\_id = $("[name=selectCate]").val();

    inquireObj.state = $("[name=selectStatus]").val();

    initList();

  });

分页：

每次当列表刷新后，分页也要跟着刷新，故：

function initList() {

    $.ajax({

      type: "get",

      url: "/my/article/list",

      data: inquireObj,

      success: function (res) {

        if (res.status !== 0) {

          return layer.msg(res.message);

        } else {

          let catelistTable = template("catelistTem", res);

          $("#tb").html(catelistTable);

          renderPage(res.total)

        }

      },

    });

  }

分页结构：

  let laypage = layui.laypage;

function renderPage(n) {

    layui.use("laypage", function () {

      //执行一个laypage实例

      laypage.render({

        elem: "pageContainer", //注意，这里的 test1 是 ID，不用加 # 号

        count: n, //数据总数，从服务端得到

        limit: inquireObj.pagesize,

        limits: [2, 3, 4, 5, 6],

        curr: inquireObj.pagenum,

        layout: ["count", "limit", "prev", "page", "next", "skip"],

        // 点击页码以及调用laypage.render都会调用jump函数

        // 如果是后者，则first为true

        // 切换每页显示条数也会触发jump

        jump: function (obj, first) {

          //obj包含了当前分页的所有参数，比如：

          console.log(obj.curr); //得到当前页，以便向服务端请求对应页的数据。

          console.log(obj.limit); //得到每页显示的条数

          inquireObj.pagenum = obj.curr;

          inquireObj.pagesize = obj.limit;

          //首次不执行

          if (!first) {

            //do something

            initList();

          }

        },

      });

    });

  }

删除文章页码逻辑：

当在最后一页删除完文章之后，应当页码-1且刷新列表，同时如果页码为1的话不能再减：

$("#tb").on("click", ".deletebtn", function () {

    let deleteid = $(this).attr("data-id");

    let btnLength = $(".deletebtn").length;

    layer.confirm(

      "是否删除该文章",

      { icon: 3, title: "删除文章" },

      function (index) {

        //do something

        $.ajax({

          type: "get",

          url: "/my/article/delete/" + deleteid,

          success: function (res) {

            if (res.status !== 0) {

              layer.close(index);

              return layer.msg(res.message);

            } else {

              layer.close(index);

              layer.msg(res.message);

              if (btnLength === 1 && inquireObj.pagenum !== 1) {

                inquireObj.pagenum--;

              }

              initList();

            }

          },

        });

      }

    );

  });

富文本编辑器：

<div class="layui-form-item">

            <!-- 左侧的 label -->

            <label class="layui-form-label">文章内容</label>

            <!-- 为富文本编辑器外部的容器设置高度 -->

            <div class="layui-input-block" style="height: 400px">

              <!-- 重要：将来这个 textarea 会被初始化为富文本编辑器 -->

              <textarea name="content"></textarea>

            </div>

          </div>

    <!-- 富文本 -->

    <script src="/assets/lib/tinymce/tinymce.min.js"></script>

    <script src="/assets/lib/tinymce/tinymce\_setup.js"></script>

  // 初始化富文本编辑器

  initEditor();

封面裁剪：

    <link rel="stylesheet" href="../assets/lib/cropper/cropper.css">

    <script src="/assets/lib/cropper/Cropper.js"></script>

    <script src="/assets/lib/cropper/jquery-cropper.js"></script>

<div class="layui-form-item">

            <!-- 左侧的 label -->

            <label class="layui-form-label">文章封面</label>

            <!-- 选择封面区域 -->

            <div class="layui-input-block cover-box">

              <!-- 左侧裁剪区域 -->

              <div class="cover-left">

                <img id="image" src="/assets/images/sample2.jpg" alt="" />

              </div>

              <!-- 右侧预览区域和选择封面区域 -->

              <div class="cover-right">

                <!-- 预览的区域 -->

                <div class="img-preview"></div>

                <!-- 选择封面按钮 -->

                <button type="button" class="layui-btn layui-btn-danger">

                  选择封面

                </button>

              </div>

            </div>

          </div>

其余js部分与头像js相似

判断发布还是草稿：

let articleStatus = '已发布'

  $("#publishbtn").click(function () {

    articleStatus = "已发布";

  });

  $("#draftbtn").click(function () {

    articleStatus = "草稿";

  });

jqformdata文件上传：

$("#articleForm").on("submit", function (e) {

    e.preventDefault();

    let fd = new FormData($(this)[0]);

    fd.append("state", articleStatus);

    $image

      .cropper("getCroppedCanvas", {

        // 创建一个 Canvas 画布

        width: 400,

        height: 280,

      })

      .toBlob(function (blob) {

        // 将 Canvas 画布上的内容，转化为文件对象

        // 得到文件对象后，进行后续的操作

        fd.append("cover\_img", blob);

      });

    fd.forEach(function (value, key) {

      console.log(value, key);

    });

    $.ajax({

      type: "post",

      url: "/my/article/add",

      data: fd,

      contentType: false,

      processData: false,

      success: function (res) {

        if (res.status !== 0) {

          return layer.msg(res.message);

        } else {

          layer.msg(

            res.message,

            {

              time: 2000, //2秒关闭（如果不配置，默认是3秒）

            },

            function () {

              location.href = "./catelist.html";

            }

          );

        }

      },

    });

  });

完整项目上传github：

