GitHub个人主页/名片网站搭建操作手册

📋 学习路径说明

🔴 **必做核心** - 必须掌握的基础技能
🟡 **重要推荐** - 建议完成，提升实践能力
🟢 **扩展选做** - 有余力时进一步学习

项目概述

创建一个个人技术名片网站，通过GitHub Pages部署，实现从需求规划到自动化上线的完整开发流程。

核心目标

- 创建个人在线简历/作品集网站
- 掌握完整的Git工作流程
- 学习项目管理和CI/CD自动化部署
- 获得可用于求职的在线作品展示

技术栈

- **项目管理**: GitHub Issues
- **代码生成**: Python (生成静态HTML)
- **版本控制**: Git & GitHub
- **CI/CD**: GitHub Actions
- **部署托管**: GitHub Pages (免费)

🔴 第一阶段：项目初始化与规划【必做核心】

🔴 1. 创建GitHub仓库【必做】

步骤1.1：创建新仓库

1. 登录GitHub，点击右上角的 `+` 按钮
2. 选择 `New repository`
3. 仓库名称： `my-dev-card` (或其他你喜欢的名称)
4. 设置为 **Public** (公开仓库才能使用GitHub Pages)
5. 勾选 `Add a README file`
6. 选择 `.gitignore` 模板： **Python**
7. 点击 `Create repository`

步骤1.2：获取仓库地址

创建完成后，复制仓库的HTTPS地址：

```
https://github.com/你的用户名/my-dev-card.git
```

🟡 2. 使用GitHub Issues进行项目规划【重要推荐】

步骤2.1：创建项目需求Issues

1. 进入仓库页面，点击 `Issues` 标签
2. 点击 `New issue` 创建以下三个Issue：

Issue #1: 创建生成网页内容的基础Python脚本

```
标题: Create basic Python script to generate homepage
描述:
- 创建 build_page.py 脚本
- 脚本能够生成基础的 index.html 文件
- 包含个人基本信息和技能列表
- 本地测试脚本功能正常
```

Issue #2: 添加个人信息和技能列表到网页

```
标题: Add personal information and skills to webpage
描述:
- 完善个人信息展示
- 添加技能列表
- 优化网页样式和布局
- 添加联系方式和项目链接
```

Issue #3: 配置GitHub Actions实现自动化部署

```
标题: Configure GitHub Actions for automated deployment
描述:
```

```
- 创建 GitHub Actions 工作流
- 实现代码推送后自动构建
- 配置 GitHub Pages 自动部署
- 验证完整的CI/CD流程
```

🔴 第二阶段：Git基础操作与本地开发【必做核心】

🔴 3. Git环境配置【必做】

步骤3.1：安装Git（如未安装）

- **Windows**: 下载 Git for Windows
- **macOS**: 使用 Homebrew `brew install git` 或下载官方安装包
- **Linux**: `sudo apt-get install git` 或 `sudo yum install git`

步骤3.2：配置Git用户信息

```
git config --global user.name "你的姓名"
git config --global user.email "你的邮箱@example.com"
```

步骤3.3：验证配置

```
git config --list
```

🔴 4. 克隆仓库到本地【必做】

步骤4.1：克隆仓库

```
# 在你想要存放项目的目录下执行
git clone https://github.com/你的用户名/my-dev-card.git
cd my-dev-card
```

步骤4.2：查看仓库状态

```
git status
git log --oneline
```

🔴 5. Git分支管理基础【必做】

步骤5.1：创建功能分支（针对Issue #1）

```
# 创建并切换到新分支
git checkout -b feature/add-script

# 验证当前分支
git branch
```

Git分支命令参考

```
# 查看所有分支
git branch -a

# 切换分支
git checkout 分支名

# 创建新分支
git branch 新分支名

# 创建并切换到新分支
git checkout -b 新分支名

# 删除分支
git branch -d 分支名
```

🔴 第三阶段：核心功能开发【必做核心】

🔴 6. 编写Python脚本【必做】

步骤6.1：创建build_page.py

在项目根目录创建 `build_page.py` 文件：

```
def main():
    # 个人信息配置
    personal_info = {
        "name": "张三",
        "title": "AI Engineer",
        "description": "AI Engineer passionate about MLOps and automation.",
        "skills": ["Python", "TensorFlow", "Git", "CI/CD", "Docker"],
        "projects": [
```

```
            {"name": "项目一", "url": "https://github.com/yourusername/project1"},
            {"name": "项目二", "url": "https://github.com/yourusername/project2"}
        ],
        "contact": {
            "github": "https://github.com/yourusername",
            "linkedin": "https://linkedin.com/in/yourprofile"
        }
    }

    # HTML模板
    html_content = f"""
    <!DOCTYPE html>
    <html lang="zh-CN">
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>{personal_info['name']} - Tech Card</title>
        <style>
            body {{
                font-family: Arial, sans-serif;
                max-width: 800px;
                margin: 0 auto;
                padding: 20px;
                line-height: 1.6;
                background-color: #f4f4f4;
            }}
            .container {{
                background: white;
                padding: 30px;
                border-radius: 10px;
                box-shadow: 0 0 10px rgba(0,0,0,0.1);
            }}
            h1 {{
                color: #333;
                text-align: center;
                margin-bottom: 10px;
            }}
            .title {{
                text-align: center;
                color: #666;
                font-size: 1.2em;
                margin-bottom: 20px;
            }}
            .skills {{
                display: flex;
                flex-wrap: wrap;
                gap: 10px;
                margin: 20px 0;
            }}
            .skill {{
                background: #007bff;
                color: white;
```

```
                padding: 5px 15px;
                border-radius: 20px;
                font-size: 0.9em;
            }}
            .projects, .contact {{
                margin: 20px 0;
            }}
            a {{
                color: #007bff;
                text-decoration: none;
            }}
            a:hover {{
                text-decoration: underline;
            }}
        </style>
    </head>
    <body>
        <div class="container">
            <h1>{personal_info['name']}</h1>
            <div class="title">{personal_info['title']}</div>
            <p>{personal_info['description']}</p>

            <h2>技能</h2>
            <div class="skills">
    """

    # 添加技能标签
    for skill in personal_info['skills']:
        html_content += f'                <span class="skill">{skill}</span>\n'

    html_content += """
            </div>

            <h2>项目经历</h2>
            <div class="projects">
                <ul>
    """

    # 添加项目链接
    for project in personal_info['projects']:
        html_content += f'                    <li><a href="{project["url"]}"
target="_blank">{project["name"]}</a></li>\n'

    html_content += f"""
                </ul>
            </div>

            <h2>联系方式</h2>
            <div class="contact">
                <p>
                    <a href="{personal_info['contact']['github']}"
target="_blank">GitHub</a> |
                    <a href="{personal_info['contact']['linkedin']}"
```

```
target="_blank">LinkedIn</a>
                </p>
            </div>
        </div>
    </body>
    </html>
    """

    # 写入文件
    with open("index.html", "w", encoding="utf-8") as f:
        f.write(html_content)

    print("index.html generated successfully!")

if __name__ == "__main__":
    main()
```

步骤6.2：测试脚本

```
# 运行脚本
python build_page.py

# 验证生成的文件
ls -la index.html

# 在浏览器中打开查看效果
# Windows: start index.html
# macOS: open index.html
# Linux: xdg-open index.html
```

🔴 7. Git提交操作【必做】

步骤7.1：查看文件变化

```
# 查看工作区状态
git status

# 查看具体变化内容
git diff
```

步骤7.2：暂存文件

```
# 添加单个文件
git add build_page.py

# 或添加所有变化的文件
```

```
git add .

# 查看暂存区状态
git status
```

步骤7.3：提交代码

```
# 提交并关联Issue
git commit -m "feat: Add basic script to generate homepage (closes #1)"
```

Git提交规范

- `feat:` 新功能
- `fix:` 修复bug
- `docs:` 文档更新
- `style:` 代码格式调整
- `refactor:` 代码重构
- `test:` 测试相关
- `chore:` 构建或辅助工具变动

步骤7.4：推送到远程仓库

```
# 推送当前分支到远程
git push origin feature/add-script
```

🔴 8. 创建Pull Request【必做】

步骤8.1：在GitHub创建PR

1. 推送完成后，访问GitHub仓库页面
2. 会出现 `Compare & pull request` 按钮，点击它
3. 填写PR信息：
   - **标题**: `Add basic script to generate homepage`
   - **描述**:

```
## 变更内容
- 创建了 build_page.py 脚本用于生成个人主页
- 脚本包含个人信息、技能列表、项目经历展示
- 支持生成响应式HTML页面

## 测试
```

```
- [x] 本地运行脚本成功
- [x] 生成的HTML文件可正常显示

Closes #1
```


步骤8.2：代码审查与合并

1. 如果有其他同学，可以请他们review代码
2. 检查无误后，点击 `Merge pull request`
3. 选择 `Squash and merge` 或 `Create a merge commit`
4. 删除功能分支（可选）

🟡 第四阶段：CI/CD自动化部署【重要推荐】

🟡 9. 配置GitHub Actions【重要推荐】

步骤9.1：创建新的功能分支

```
# 切换回主分支并拉取最新代码
git checkout main
git pull origin main

# 创建CI配置分支
git checkout -b feature/add-ci
```

步骤9.2：创建工作流文件

创建目录结构和文件：

```
mkdir -p .github/workflows
```

创建 `.github/workflows/deploy.yml`：

```
name: Deploy Personal Tech Card

on:
  push:
    branches:
      - main  # 只在main分支更新时触发

jobs:
  build-and-deploy:
```

```
    runs-on: ubuntu-latest

    # 为GitHub Pages设置权限
    permissions:
      contents: read
      pages: write
      id-token: write

    steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: '3.10'

    - name: Generate HTML file
      run: python build_page.py

    - name: Setup Pages
      uses: actions/configure-pages@v4

    - name: Upload artifact
      uses: actions/upload-pages-artifact@v2
      with:
        path: '.'  # 上传整个目录，GitHub Pages会找到index.html

    - name: Deploy to GitHub Pages
      id: deployment
      uses: actions/deploy-pages@v3
```

步骤9.3：提交CI配置

```
git add .github/workflows/deploy.yml
git commit -m "feat: Add GitHub Actions workflow for automated deployment
(closes #3)"
git push origin feature/add-ci
```

🔴 10. 配置GitHub Pages【必做】

步骤10.1：启用GitHub Pages

1. 进入仓库的 `Settings` 页面
2. 在左侧菜单找到 `Pages`
3. 在 `Build and deployment` 部分：
   - **Source**: 选择 `GitHub Actions`

4. 保存设置

步骤10.2：合并CI分支

1. 创建PR将 `feature/add-ci` 合并到 `main`
2. 合并完成后，GitHub Actions会自动触发
3. 在 `Actions` 标签页可以查看部署进度

步骤10.3：访问网站

部署成功后，访问：

```
https://你的用户名.github.io/my-dev-card/
```

🟡 第五阶段：功能完善与维护【重要推荐】

🟡 11. 完善个人信息 (Issue #2)【重要推荐】

步骤11.1：更新个人信息

```
# 创建新分支
git checkout main
git pull origin main
git checkout -b feature/update-info
```

修改 `build_page.py` 中的个人信息，更新为你的真实信息：

- 姓名和职位
- 个人描述
- 技能列表
- 真实的项目链接
- 联系方式

步骤11.2：测试和提交

```
python build_page.py
# 在浏览器中检查效果

git add build_page.py
git commit -m "feat: Update personal information and skills (closes #2)"
```

```
git push origin feature/update-info
```


步骤11.3：创建PR并合并

合并后，GitHub Actions会自动重新部署网站。

🟡 12. 日常维护和更新【重要推荐】

更新流程

1. 创建新分支
2. 修改代码
3. 本地测试
4. 提交并推送
5. 创建PR
6. 合并后自动部署

常用Git命令参考

```
# 查看提交历史
git log --oneline --graph

# 查看文件变化
git diff HEAD~1 HEAD

# 撤销工作区变化
git checkout -- 文件名

# 撤销暂存区变化
git reset HEAD 文件名

# 修改最后一次提交
git commit --amend

# 查看远程仓库信息
git remote -v

# 同步远程分支
git fetch origin
git merge origin/main
```

🟡 故障排除【重要推荐】

常见问题及解决方案

1. GitHub Pages没有部署成功

症状: Actions运行成功但网站无法访问
解决方案:

1. 检查仓库是否为Public
2. 确认Pages设置中Source选择了GitHub Actions
3. 查看Actions日志是否有错误信息

2. Python脚本运行失败

症状: `python build_page.py` 报错
解决方案:

1. 检查Python版本：`python --version`
2. 确认文件编码为UTF-8
3. 检查文件路径和权限

3. Git推送失败

症状: `git push` 提示认证失败
解决方案:

1. 检查Git配置：`git config --list`
2. 使用GitHub Personal Access Token
3. 检查仓库权限

4. 分支合并冲突

症状: 合并时出现冲突
解决方案:

1. 手动解决冲突文件
2. 添加解决后的文件：`git add 文件名`
3. 完成合并：`git commit`

🎯 核心必做任务清单

📍 学员必须完成的关键步骤

🔴 阶段一：GitHub基础操作

- [ ] 创建GitHub账号并建立public仓库
- [ ] 学会仓库的基本设置和管理
- [ ] 获取仓库克隆地址

🔴 阶段二：Git本地操作

- [ ] 安装并配置Git环境
- [ ] 掌握 `git clone` 、 `git status` 、 `git log` 基础命令
- [ ] 学会创建和切换分支： `git checkout -b`
- [ ] 掌握Git三步提交流程： `add` → `commit` → `push`

🔴 阶段三：核心开发

- [ ] 编写并运行Python脚本生成HTML页面
- [ ] 掌握Git提交规范和消息格式
- [ ] 学会创建和管理Pull Request

🔴 阶段四：网站发布

- [ ] 配置GitHub Pages基本设置
- [ ] 成功访问个人主页网站： `https://用户名.github.io/仓库名/`

💡 学习建议

1. **循序渐进**：严格按照🔴必做核心部分的顺序学习
2. **动手实践**：每个命令都要亲自操作，不要只看不做
3. **理解原理**：不仅要知道怎么做，还要理解为什么这样做
4. **记录笔记**：记录遇到的问题和解决方案
5. **反复练习**：Git操作需要反复练习才能熟练掌握

📚 快速参考

最常用Git命令（必须记住）

```
# 基础操作
```

```
git status                # 查看仓库状态
git add .                 # 添加所有文件到暂存区
git commit -m "消息"      # 提交变更
git push origin 分支名    # 推送到远程

# 分支操作
git checkout -b 新分支名 # 创建并切换到新分支
git checkout 分支名      # 切换分支
git branch               # 查看本地分支

# 同步操作
git pull origin main    # 拉取远程main分支最新代码
git clone 仓库地址      # 克隆远程仓库
```

GitHub Pages访问地址格式

```
https://你的GitHub用户名.github.io/仓库名/
```