



上海大学

SHANGHAI UNIVERSITY

本科毕业论文（设计）

UNDERGRADUATE THESIS (PROJECT)

题 目：基于 Go Hugo 的静态网站设计和开发

学 院：中欧工程技术学院

专 业：信息工程

学 号：19124644

学生姓名：朱馨宁

指导教师：杜金鑫

起讫日期：2022/12/26 — 2023/5/31



姓 名：朱馨宁

学号：19124644

论文题目：基于 Go Hugo 的静态网站设计和开发

原创性声明

本人声明：所呈交的论文是本人在指导教师指导下进行的研究工作。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已发表或撰写过的研究成果。参与同一工作的其他同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

签 名：_____日 期：_____

本论文使用授权说明

本人完全了解上海大学有关保留、使用学位论文的规定，即：学校有权保留论文及送交论文复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容。

（保密的论文在解密后应遵守此规定）

签 名：_____指导教师签名：_____日期：_____

摘 要

作为静态网页搭建的重要应用方案，官方网站能够作为各种企业以及个人开展各种宣传和推广活动的重要平台，从而帮助快速创建一份专业的网络上的介绍。但是，等困难。因此，本文提出一种基于静态网站搭建的自动化部署方案，以应对以上困难。

该方案的目的在于帮助用户在具有基本电脑操作技能的情况下快速、高效地完成网站的建设和部署，并提高网站建设的效率和质量。本文通过 Hugo 静态网站生成器和 GitHub Pages 免费的静态网页托管服务帮助用户搭建一个美观的网站而无需编写复杂的代码，同时，使用自动化部署方案可以实现网站的快速更新和部署，从而大大提高网站建设的效率和质量。

本研究源于中法合作的 METISLAB 实验室官网的搭建，对 Hugo 静态站点生成工具进行了深入的研究，并结合了 GitHub 交互工具。本文通过详尽探讨 Hugo 的设置与用法、主题与模板引擎的运用，并讨论了如何结合 GitHub Actions 实现自动构建与部署的方法。

本研究成果对于静态网站建设和日常内容数据的管理发布具有重要的实际应用价值。

关键词：Hugo；静态网站；GitHub Actions；模板引擎

ABSTRACT

As an important application solution for static web building, an official website can serve as an important platform for various businesses and individuals to carry out various publicity and promotional activities, thus helping to quickly create a professional presentation on the web. However, in the process of website construction, the people involved usually encounter difficulties such as insufficient technical skills and difficulties in meeting business needs. Therefore, this paper proposes an automated deployment solution based on static website building to cope with these difficulties.

The aim of this solution is to help users to build and deploy websites quickly and efficiently with basic computer skills, and to improve the efficiency and quality of website construction. This paper uses the Hugo static website builder and the GitHub Pages free static web hosting service to help users build a beautiful website without writing complex code, while using an automated deployment solution to enable rapid updates and deployments, thus greatly improving the efficiency and quality of website construction.

This research originated from the construction of the official website of the METISLAB lab, a Sino-French collaboration, and provides an in-depth study of the Hugo static site generation tool, combined with the GitHub interactive tool. The paper explores in detail the setup and usage of Hugo, the use of themes and template engines, and discusses how to automate the build and deployment in conjunction with GitHub Pages.

The research results have important practical application value for static website construction and daily content data management and distribution.

Keywords: Hugo; static websites; GitHub Actions; template engine

目 录

摘 要.....I

ABSTRACT..... II

目 录.....I

1 绪论..... 1

1.1 研究背景和意义..... 1

1.2 国内外研究现状及发展趋势..... 1

1.2.1 静态网页的发展..... 1

1.2.2 动态网页的发展..... 2

1.2.3 动态生成静态网页和静态网页生成器技术..... 4

1.3 论文组织架构..... 8

2 相关理论..... 9

2.1 静态网站的构建..... 9

2.1.1 SSG 技术和 Hugo..... 9

2.1.2 模板引擎的基本原理..... 12

2.1.3 在 Hugo 中模板引擎的实现..... 14

2.2 静态网页的自动化部署..... 16

2.2.1 CI/CD 流程..... 16

2.2.2 GitHub Actions 和 GitHub Pages..... 18

3 系统的需求分析与设计..... 20

3.1 系统的需求分析..... 20

3.2 系统设计..... 22

3.2.1 架构设计..... 22

3.2.2 模块设计..... 25

4 系统实现..... 31

4.1 环境搭建..... 31

4.1.1 Hugo	31
4.1.2 GitHub Actions	31
4.2 系统功能实现	32
4.2.1 网站结构	33
4.2.2 网站内容	35
4.2.3 多语言支持	43
4.2.4 网站维护	44
5 总结与展望	45
5.1 本文总结	45
5.2 展望	46
参考文献	48
致 谢	50

1 绪论

本课题来源于 METISLAB 国际联合实验室，该实验室由中欧学院牵头，上海大学、法国里昂国立应用科学学院、哈尔滨工业大学、哈尔滨医科大学共同启动，并获得法国科学院的资助和支持。该实验室致力于在医学成像及智能处理技术、在体传感及内窥检测技术和自主感知及柔性机器人等方向展开研究工作，并通过开放式国际实验室的方式促进国际间科研合作。该论文的选题正是基于 METISLAB 所建设的网站，在 METISLAB 国际联合实验室的支持下，本课题得以更好地展开，同时为该实验室的研究工作提供更好的展示平台。

1.1 研究背景和意义

目前，在社会各个领域，网站已成为获取信息资源、进行科研与学习交流的重要手段之一。不同专业和行业的机构和企业众多，其相应网站的建立需要通过较为便捷、易上手的工具加以实现。本课题旨在研究和探索使用 Go Hugo 静态网站生成器进行静态网站的设计和开发，并结合 GitHub Actions 技术，实现静态网站的自动构建、部署和发布，提高网站建设的效率和质量。

Diaz, Chris.在《Minimal Computing with Progressive Web Apps》^[1]一文中提出：“静态网站有助于降低我们的成本，同时使我们能够灵活地为学生和学者制作可定制、现代、安全和内容丰富的网站。”相较于动态网站，静态网站的优点是访问速度快、安全性高、易于部署等。静态网站无需在服务器端动态生成页面，完全由浏览器展示，因此可以省去服务器的运行资源和带宽资源，同时也减少了服务器维护的难度。

本课题采用 Go Hugo 这一静态网站生成器工具，实现实验室网站的搭建。Go Hugo 的模板引擎和易于使用的结构使开发人员可以轻松创建漂亮而高效的网站。同时它还支持各种主题和插件集成，实现更高级别的定制和扩展。此外，本课题基于 GitHub Actions 技术，实现网站的自动构建和自动部署。通过自定义 GitHub Pages 的工作流程，将静态网站的生成和发布任务自动交给 GitHub 平台处理，自动构建和部署实验室网站，从而实现高效、快速、可靠的静态网站更新和发布，显著提高网站建设的质量和效率。

对于实验室人员而言，实验研究是他们的首要任务，而其对于网站开发和运营能力相对薄弱。然而，随着互联网技术的不断发展和实验室国际合作的不断扩大，建设并维护一个具有良好可读性和用户友好度的实验室网站极有必要。因此，本研究的意义在于探索了基于 Go Hugo 的静态网站设计和开发方法和应用。在提高网站的访问速度、安全性和可靠性的同时，降低网站的维护成本和开发难度。从而，为学生和学者提供了一个现代、安全、内容丰富和可定制的学术交流平台。本研究的方法和应用也为其他学术实验室和静态网站受众提供了一个可行的网站建设方案，具有一定的推广和应用价值。

1.2 国内外研究现状及发展趋势

1.2.1 静态网页的发展

网页是指在浏览器上呈现的一种视觉化页面，也被称为 Web 页面，通常以文件形式存储在与互联网相连的服务器上。静态网页是指网页的内容固定不变，使用标准的 HTML 代码编写，文件通常以 `htm`、`html` 等为后缀名。静态网页是 Web 开发中最早的形式之一，它是由 HTML、CSS 和 JavaScript 等静态资源组成的，通常在服务器端生成好后，直接返回给用户。

在 Web1.0 时代，静态网页是互联网主要的网页形式。1991 年，Tim Berners-Lee 发明 Web，创造了现代互联网的基础，网站只采用静态网页。随着互联网的发展，Web 进入了图形化界面时代、商业化运营、普及和进步，1995 年 Netscape Navigator 浏览器的发布进一步地推动了静态网页的扩展。至今，静态网页仍是一种重要的网页形式，在 Web 开发中得到广泛应用。

与此同时，在 Web1.0 时代，中国互联网及静态网页的发展相对于国际互联网较为滞后，中国的网站数量和网民数量也相对较少。1997 年，中国第一家互联网服务提供商——中国互联网络信息中心（CNNIC）的成立，标志着中国互联网的发展进入了一个新的阶段，为中国静态网页的发展奠定了基础。2000 年百度公司成立，开始提供搜索引擎服务，搜索结果页面基本上都是静态网页，包括搜索结果列表、网页快照等内容，成为中国互联网发展的重要里程碑。随着 2003 年中国开始大规模建设宽带网络，推动了中国互联网的普及和发展，为静态网页的传播提供了更快的速度和更好的用户体验。

在实际应用中，在 Web 开发的早期，静态网页是主流的开发方式，但是随着 Web 应用程序的复杂性和用户需求的增加，静态网页逐渐无法满足需求，动态网页逐渐成为主流。

静态网站的网页具有如下特点^[2]:

(1) 静态网站的每个网页都有一个固定的 URL,且网页 URL 以.htm.html 等常见形式为后缀，但不含有“?”；

(2) 静态网站的网页内容一经发布到网站服务器上,无论是否有用户访问,每个静态网站的网页的内容都是保存在网站服务器上的，每个网页都是一个独立的文件；

(3) 静态网站的网页的内容相对稳定，因此容易被搜索引擎检索；

(4) 静态网站的网页没有数据库的支持，在网站制作和维护方面工作量较大，因此当网站信息量很大时完全依靠静态网页制作方式比较困难；

(5) 静态网站的网页的交互性较差,在功能方面有较大的限制。

总之，静态网站的特点适用于一些简单的网站需求，如企业宣传网站、个人博客等。此类网站内容相对固定，静态页面可以更好地满足需求，同时也可以降低网站的维护成本。于此同时，对搜索引擎友好，可以更好地提升网站的排名和曝光度。

1.2.2 动态网页的发展

动态网页是在服务器端根据用户请求生成动态内容的网页，通常使用动态网页技术（如 PHP、ASP.NET 等）来实现。动态网页的优点是可以实现动态交互和个性化定制，但是缺点是访问速度慢、安全性较低。随着 Web 应用程序的发展和技术的进步，动态网页的性能和安全性得到了很大的提升，同时也出现了一些新的技术和框架，例如 AJAX、Node.js 等，使得动态网页的开发更加方便和高效。

动态网页生成技术是 Web2.0 时代的一个重要发展方向，它的发展历程可以分为以下几个阶段：

1. CGI（公共网关接口）时代（1993 年-1996 年）：CGI 是一种 Web 服务器与外部应用程序之间的标准接口，它可以让 Web 服务器调用外部程序来生成动态内容。在 CGI 时代，动态网页的生成主要依靠 CGI 技术，例如 Perl、Python

等脚本语言。

2. ASP^[3]（Active Server Pages）时代（1996 年-2002 年）：ASP 是用于 Web 应用程序开发的一种主流开发技术，它可以把用 VBScript 语言写的服务器端脚本嵌入到 Web 页面中，在服务器端动态生成页面内容，还可以过 COM 组件与数据库连接，从而提供强大的事务处理功能，因此，ASP 技术已经被广泛的用于开发 Web 应用程序。但是，由于 ASP 技术自身的一些特点，使得它并不适于开发大规模、复杂的 Web 应用程序，

3. PHP^[4]（Hypertext Preprocessor）时代（2000 年-至今）：PHP 是一种开源的动态网站开发语言，最初只是一个简单的用 Perl 语言编写的程序，后来越来越多的网站使用了 PHP，并要求扩充 PHP 的新特性，逐渐发展为今天功能强大的用于开发动态网站的 PHP 语言。它可以让 Web 服务器调用 PHP 脚本来生成动态内容。在 PHP 时代，动态网页的生成主要依靠 PHP 技术，它可以与数据库进行交互，实现动态内容的生成和管理。

4. AJAX^[5]（Asynchronous JavaScript and XML）时代（2005 年-至今）：Ajax 是一种用于创建交互式网页应用的网页开发技术，它通过异步操作向服务器获取数据，并在后端处理数据和执行代码，从而打破了页面重复加载的惯例，减轻了服务器端的负担。Ajax 技术实现了无刷新的更新页面，减少了用户的实际和心理等待时间，创造和实现了更丰富、更动态的应用程序界面。Ajax 是 web 应用的一种新方法。它并不是一门新的语言或技术，实际上是几种已经在各自领域大行其道技术的强强结合，Ajax 混合了 XHTML/CSS, DOM,XML 及 XSLT 等几项技术，并且利用 Javascript 来整合上述技术。Ajax 为交互较多，频繁读数据，数据分类良好的 Web 应用提供了一个很好的解决方案。

在动态网页技术的发展方面，中国和国际是比较同步的。中国的研究者和开发者在动态网页技术的研究和应用方面积极探索和创新，同时也与国际接轨，吸收和借鉴国际先进的技术和经验。例如，中国的阿里巴巴、腾讯、百度等企业在动态网页技术的研究和应用方面都处于领先地位，它们的技术水平和应用效果已经达到或者超过了国际水平。

同时，中国的开发者和学者也在为动态网页技术的发展及其劣势的弥补积极贡献。例如，王晓强在《基于 HTML5 的 CSRF 攻击与防御技术研究》^[6]中针对

动态网站在丰富 Web 应用的同时存在的安全漏洞，特别是 CSRF 攻击，分析了基于 HTML5 的 Web 存在的安全问题，并提出了如何利用工具和手动方式对 CSRF 漏洞进行检测，并提出了一些防御措施。畅玉洁在《NET 与数据库技术在动态网站开发中的探讨》^[7]中探讨了 NET 与数据库技术在动态网站开发中的具体应用路径。在刘奇旭等人的专利《一种 Node.js 代码安全检测方法及系统》^[8]中提出了一种 Node.js 代码安全检测方法及系统，弥补了现有检测方法不能检测出控制流和数据流较为复杂中代码的安全风险的缺陷，有效地提高了 Node.js 代码的安全性。

总的来说，动态网页生成技术是 Web2.0 时代的一个重要发展方向，它的发展历程经历了 CGI、ASP、PHP 和 AJAX 等不同的阶段。这些技术的发展推动了动态网页的应用和发展，为互联网的快速发展提供了更好的技术支持。

尽管动态网页技术在实现网页交互性、美观性、功能性方面有着显著的优势，但是它也存在着一些劣势，如运行速度等性能问题、安全问题、兼容性问题、维护问题等。这些问题制约了动态网页技术的发展，需要不断地进行技术改进和优化。

中外研究者在动态网页技术的发展中做出了很多努力和贡献，不断地推动技术创新和发展，以满足不断变化的用户需求和提高用户体验。然而，在一些对性能和安全要求较高的网站上，使用动态网页技术可能并不是最优的选择。

1.2.3 动态生成静态网页和静态网页生成器技术

动态网页需要通过服务器端脚本语言动态生成网页内容，因此加载速度较静态网页慢。有实验结果表明，经过静态化处理后网页响应速度明显提升，并大幅度提升系统处理并发请求数量^[9]。此外，存在一定的安全性问题，如 SQL 注入、跨站脚本攻击等，需要采取相应的安全措施进行防范。

（1）动态生成静态网页

为了解决动态网站的缺点，研究者们一直在探索新的技术和方法。刘耀钦在《基于 Smarty 模板引擎的 Web 页静态化研究与性能分析》^[10]一文中提出：动态 Web 页的静态化处理不仅可以减少用户请求等待的时间,而且还可以增强应用系统的运行性能,更重要的是可以有效的降低并发访问量较大时服务器的工作负载。

真静态化处理方案中用户请求等待时间最短,服务器传输速率最快,在解决好 HTML 文件存储空间的情况下,是中大型 Web 应用项目常用的静态化处理方案。除此之外,冯兴利等人^[11]在《基于 PHP+MySQL 的 Web 系统安全防范及全站静态化》一文中提出:随着网站信息量的增加,不仅 Web 应用服务器的负担随之增加,而且不利于提高网站内容被搜索引擎搜索到的几率。通过静态化操作可以确保系统安全,并提高 Web 服务器的运行效率。

雷海卫、张萍^[12]借助 ASP 技术生成静态页,其中主要使用了文件对象来完成对文件生成、读取等操作。在此基础上,曾春华、江南雨^[13]介绍了利用动态网页技术生成静态 HTML 页面的方法。利用这种技术,网站内容管理人员在添加网页时直接利用后台管理发布程序就把页面存放成 HTML 静态文件,生成页面简单、快速。进一步地,徐白、宋玲和吴昊^[14],基于模板方法和基于 URL 方法,将需要生成的数据,转成流输出成文件,实现 JSP 静态网页生成技术。随后,在他们的基础之上,黄立冬^[15]利用 HTML+Java Script+Ajax+动态技术(ASP),实现静态与动态相结合的半静态模式的网站。

此外杨昕等人的专利发明《一种全站静态化的方法和页面静态化的方法》^[16]充分利用 web 服务器处理纯静态化内容高效的特点,提高静态化后网站的效率,还提高了动态网页静态化的效率。郑丽敏等人的专利发明《网页内容静态化处理方法及系统》^[17]提供了一种网站内容静态化处理方法及系统,通过接收网站访问请求,并根据网站首页静态化模板和访问请求,生成目标网站对应的首页显示内容,从而使得网站响应能得到大幅度提升,尽可能的避免了响应超时情况。

这些研究者们提出的方法都是为了将动态网站转换成静态网站,从而提高网站的性能和用户体验。他们利用 ASP 技术、动态网页技术、模板方法等技术手段,将动态网站的内容生成为静态 HTML 文件,从而实现网站的静态化。其工作作为网站的静态化提供了重要的思路和方法,为网站的性能和用户体验的提升做出了贡献。

（2）静态网页生成器技术（SSG）/Static Site Generator

与上述研究者们提出的方法类似,静态网页生成器技术（SSG）/Static Site Generator 是一种专门用于静态网站生成的工具,它具有更高的效率和更易于使用的特点。SSG 技术可以将网站的内容预先生成为静态文件,而不是在每次用户

请求时动态生成，从而提高网站的性能和安全性，同时也可以提高网站的维护性和兼容性。此外，使用 SSG 技术可以将网页部署到 CDN 上，以提高网页的访问速度和可靠性。

综上所述，SSG 技术是一种较为成熟的解决方案，可以有效地解决动态网站的缺点，提高网站的性能、安全性和搜索引擎友好度。在建设网站时可以考虑采用 SSG 技术，以提高网站的质量和用户体验。这样既可以保证访问速度和安全性，又可以实现动态交互和个性化定制。

在国外的研究中，Petersen 和 Hillar^[18]对比了 Jekyll、Hexo 和 Hugo 三款最受欢迎的静态网站生成器，发现它们各有特点，其中，Hugo 使用 Go 编写，只能支持外部助手，但具有更高的性能，适用于大型网站，且易于使用。Chris Diaz^[19]在西北大学图书馆的案例中使用 Jekyll 和 Bookdown 这两个开源的静态网站生成器，用于其数字出版服务，在其研究中指出，静态网站生成器有助于为学术或教育出版物创建简单、强大、低成本的网站。EP Williamson^[20]等人使用静态网站生成器 Jekyll 创建了一个可持续的、敏捷的开发模式，并使用其促进跨部门的沟通、协作，为不同技术能力的图书馆工作人员提供创新的功能开发。针对 Go Hugo，Prayudi Utomo 和 Falahah^[21]进行了多项测试来检查网站的质量，例如功能测试和性能测试。测试结果表明使用 Go Hugo 构建的网站虽然通过 API 与其他服务连接，但仍能以良好的性能运行。

总的来说，静态网站生成器作为一种越来越受欢迎的技术，在国外得到了广泛的应用，并有相应的研究。研究证明，SSG 作为新兴的静态网站生态系统的一部分，在实践中展现其良好的性能，并正在扩大和成长为产生令人惊叹效果的工具集。

在国内，随着互联网技术的不断发展，越来越多的开发者开始关注 SSG 技术。有学者提出^[22]：使用静态网页生成器可有效解决很多公司想拥有自己的网站但缺乏时间和技术的问题。使用这种方法可以使公司节省制作网站的时间，从而使商家有更多的精力考虑客户的需求和兴趣。然而，国内的 SSG 应用主要集中在个人博客、企业官网、电商网站等领域，关于 SSG 的学术研究还比较有限，未来还有很大的发展空间。

1.2.4 总结

静态网页和动态网页是 Web 开发中两个重要的概念，它们的发展历程和技术演进对于 Web 应用程序的设计和实现具有重要的意义。随着 Web 应用程序的复杂性和用户需求的增加，动态网页逐渐成为主流，但是静态网页仍然具有一定的优势。

SSG 和动态网页静态化都是为了实现更高的性能、安全性和更好地用户体验而发展起来的技术。随着技术的不断进步和应用场景的不断扩展，提高网站的性能和安全性将成为 Web 开发的一个重要趋势，SSG 技术的研究有着重要的意义。

在实验室官网搭建这一案例中，使用静态网页而非动态网页的原因是：

1. 静态网站没有数据库和服务端脚本，不需要与服务器进行交互，因此安全性高，不易受到黑客攻击。

2. 由于静态网站的页面都是独立的 HTML 文件，没有使用服务端脚本语言生成动态内容，因此搜索引擎更容易识别和索引这些页面，从而提高实验室网站的曝光度。

3. 静态网页的管理和维护比动态网页更容易且成本更低，因为每个页面都是一个独立的 HTML 文件，可以通过简单的文件操作来进行更新和维护。这种优势与静态网站生成器（SSG）技术相结合，可以进一步提高网站的管理和维护效率。

总之，SSG 技术使得静态网页的生成和更新更加容易和快速，弥补了静态网页手动更新复杂的缺陷，同时也可以提高网站的性能和安全性。当然，这并不意味着静态网页适用于所有的网站，对于需要处理大量访问和高流量的网站，动态网页会更加适合。实验室官网作为小型的信息展示平台，纯静态的网站能够较好地满足实验室需求，无需增加动态交互等功能增加维护成本。

本文探索使用 Go Hugo 这一 SSG 来设计和开发静态网站的可行性和优势。在中国，Hugo 的研究现状相对较少，但是随着 SSG 的普及，Hugo 也在逐渐被研究和应用。本文为其他开发者提供一个参考和借鉴的案例，帮助他们更好地理解和应用静态网页生成器技术。

1.3 论文组织架构

整篇论文一共分以下为五章：

第一章主要阐述研究来源、研究背景和意义，并介绍了静态网页和动态网页的发展过程，以及为了弥补动态网页技术缺陷而发展出的动态生成静态网页技术和 SSG 技术。

第二章主要介绍在实现静态网站的建设和自动部署过程中所应用的相关理论。介绍 SSG 技术、模板引擎的基本原理和实现方法，并且探究在 Hugo 中实现 SSG 和模板引擎的具体方法。此外，介绍了 CI/CD 流程，以及使用 GitHub Actions 进行自动化部署的基本原理和方法。

第三章是对系统的需求性分析与设计，针对 METISLAB 的需求进行分析，并进而从架构、模块两个部分着手设计。

第四章针对第三章的需求分析，对系统的实现方法进行了介绍。共分为两个部分，第一部分介绍了 Git、Hugo 和 GitHub Actions 环境的搭建。第二部分介绍系统功能的实现，包括网站结构、内容和多语言支持等功能的实现以及网站后期的运维。

结论部分，总结了本文的研究成果和贡献，同时展望未来的研究方向和发展趋势。

2 相关理论

本章主要介绍使用在实现静态网站自动化部署过程中所应用的相关理论。其中，第一部分介绍了动态生成静态网页的原理，将介绍 SSG 技术、Hugo 和模板引擎的基本原理和实现方法，并且探究在 Hugo 中实现 SSG 和模板引擎的具体方法。第二部分，即自动化部署和持续，重点介绍了 CI/CD 的基本原理，以及使用 GitHub Actions 进行自动化部署的基本原理和方法。通过对该章节的学习，读者将能够系统化地理解 Hugo 和 GitHub Actions 的实现原理，为静态网站的设计与开发提供一定的指导和参考。

2.1 静态网站的构建

对于中小型网站而言，使用静态网站可以大大提升网站性能和降低服务器负担。而对于大型动态网站而言，静态化页面也可以作为降低服务器负荷的有效策略。静态网站具有加速网站访问速度、优化搜索引擎排名、提高网站稳定性和安全性等多个优点，值得中小型网站进行尝试。

2.1.1 SSG 技术和 Hugo

（1）SSG 技术

静态网页生成器（SSG）/Static Site Generator 是当前流行的一种制作静态网站的工具，其主要思想是在开发阶段将动态网站转换为纯静态的 HTML 文件，以提高网站的访问速度和安全性。与传统的 Web 应用不同，静态站点生成器不需要在服务器端进行动态生成页面，而是在开发阶段就将网站内容和布局固定下来，以提升用户访问体验和开发效率。

在 SSG 中，整个网站的生成过程就是模板引擎将布局文件和内容文件结合起来，生成网站的各个页面，并将其存储在指定的目录下。这些页面之间通常是相互关联、相互链接的。

构建是 SSG 工具的核心功能，其主要作用是将网站的源文件解析和转换成 HTML 文件，并生成相应的 CSS 和 JavaScript 等文件。具体来说，构建流程如图 2.1 所示，一般包括以下几个步骤：

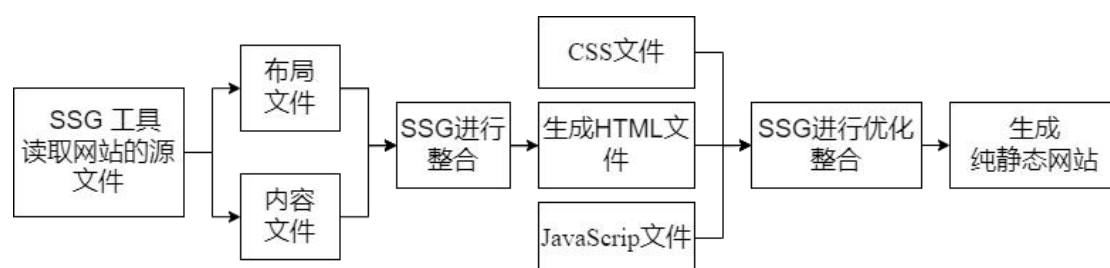


图 2.1 SSG 部署过程

首先，SSG 工具会读取网站的源文件，包括布局和内容两种类型的文件。布局文件通常包括模板、组件和样式等内容，用于定义网站的页面结构和外观。内容文件则包括文章、图片、音频、视频等资源，用于填充网页的具体内容。

接下来，SSG 工具会将布局和内容文件进行解析和组合，以生成新的 HTML 文件。这一过程中，SSG 工具会使用特定的模板语言（如 Jekyll、Hugo、Hexo 等），将源文件中的变量替换为实际的内容，并通过网页中的指令和标签生成各种布局效果和交互功能。

最后，SSG 工具会将生成的 HTML 文件和相应的 CSS、JavaScript 等文件进行整合和优化，以生成最终的纯静态网站。通常情况下，SSG 工具会支持各种优化选项，例如代码压缩、图片优化、资源合并等，以进一步提升网站的性能和用户体验。

如果开发人员对网站进行了更改，构建过程将重新开始，重新创建整个静态网站。SSG 工具还支持多种文件格式和模板语言，方便开发者根据自己的需求进行定制化开发。

需要注意的是，虽然 SSG 制作的网站是静态网站，但是可以借助一些动态技术（如 Ajax、JavaScript 等）实现动态效果。

相比传统的 Web 应用，静态站点生成器具有很多优势。首先，它可以极大地提高网站的访问速度，避免了动态页面生成和数据库查询等操作，同时也降低了服务器负荷，提高了可靠性和安全性。其次，静态站点生成器使用简单，不需要太多的后端语言知识，也不需要数据库或服务器的支持，可以在本地电脑上进行开发和预览。最后，静态站点生成器还支持多种扩展插件和自定义主题，可以满足不同场景下的开发需求。

（2）Hugo

Hugo 是一种静态网站生成器（Static Site Generator, SSG），它使用 Go 语言编写，可以将 Markdown 文件或其他格式的文本文件转换为 HTML 文件，并生成静态网站。Hugo 的目标是提供一个简单、高效、易用的静态网站生成器，使用户可以快速搭建和发布静态网站。与其他 SSG 技术相比，它具有以下优势和特点：

1. 速度快：Hugo 使用 Go 语言编写，Go 语言具有高效的并发处理和内存管理能力。Hugo 还使用了一些优化技术来提高生成速度。其中一项技术是增量构建（Incremental Builds），具体而言，当 Hugo 检测到有文件发生了变化，它会快速地比对新旧文件的差异，然后仅仅更新那些发生变化的文件、或者依赖变化的文件。这些变化可能包括内容修改、文件重命名、添加/删除文件等等。这样做能够消除不必要的重复处理，可以有效提高网站的生成速度。

除了增量构建，Hugo 还使用了其他许多优化技术，例如将网站内容划分为多个部分，使用缓存机制、并行处理等等。这些技术的应用，都能够让 Hugo 的生成速度更加迅速、高效。因此，Hugo 的速度非常快，可以在几秒钟内生成数千个页面。

2. 易于使用：Hugo 的使用非常简单，只需要安装 Hugo 并学习一些基本命令即可。Hugo 还提供了丰富的主题和插件，可以轻松地创建和定制自己的网站。对于不具备专业编程技能的用户，使用 Hugo 可以快速搭建网站，因为它操作简单并提供了易于理解的指导文档。

3. 可扩展性强：Hugo 支持多种数据格式和模板语言，可以方便地集成其他工具和服务，如 GitHub、Netlify 等。同时，Hugo 还支持自定义输出格式和内容类型，可以满足不同用户的需求。

4. 社区活跃：Hugo 拥有固定的用户社区和开发者社区，提供了丰富的文档和教程，用户可以通过社区获得支持和帮助。

总之，Hugo 是一种高效、安全、易用、可扩展、社区活跃的静态网站生成器，具有很多优势和特点，可以满足不同用户的需求。虽然 Hugo 在国内的知名度相对较低，但是随着静态网站生成器的流行，越来越多的人开始关注和使用 Hugo，Hugo 在国内的社区也在逐渐壮大。

2.1.2 模板引擎的基本原理

SSG 工具在处理并输出 HTML 文件时，需要借助模板引擎技术来处理 Web 模板和内容信息并生成 Web 文档。在传统的 Web 应用程序中，业务逻辑和表现逻辑常常混在一起，使得代码复杂、难以维护、不易扩展。模板引擎作为一种解决方案，可以将业务逻辑与表现逻辑分离，通过数据模型和模板文件的组合，生成 HTML、XML、JSON 等格式的页面输出。这种方式可以使得开发人员专注于业务逻辑的实现，而不需要关心页面的设计和排版。同时，也方便了前端开发人员针对同一数据模型，快速地修改页面样式。

模板引擎的另一个作用是提高视图组件的重用性。通过将视图组件封装成模板文件，可以在多个页面中进行复用，从而减少冗余的代码和工作量。这样，不仅可以提高开发效率，还可以减小代码维护的成本。

此外，由于模板引擎将业务逻辑和表现逻辑分离，使得开发人员可以更加方便地扩展和维护系统的功能。在需要修改或增加页面的情况下，只需要针对数据模型和模板文件进行调整即可，而不会对业务逻辑造成任何影响。

模板引擎的使用方法一般包括三个步骤：

步骤一：选择模板文件。模板引擎提供了丰富的模板文件供开发人员选择，这些文件通常包含了网站页面的 HTML 框架和布局。开发人员需要选择适合自己项目需求的模板文件。

步骤二：获取数据值。数据通常来自于 Web 应用程序的后台，可以是 JSON 格式的数据、对象或者数组等等。

步骤三：填充数据。在获取数据值之后，开发人员需要将这些数据值填充到模板文件中，从而形成完整的 HTML 页面代码。

模板引擎的内部结构包括词法器、语法器和解释器，它们的主要作用如下：

词法器：逐个字符读入源代码文本，并将其分类成操作符、关键字、变量名、数字等不同类型。词法器根据定义好的正则表达式进行识别，将源代码文本转化为一系列的标记。

语法器：根据事先定义好的文法分析标记序列，并生成树形结构的抽象语法树（AST）。AST 是一种用于描述源代码文本的语法结构的数据结构。创建 AST 的过程也称为“解析”。

解释器：解释器遍历 AST，并将其中的变量值和操作指令转化为实际的 HTML 代码。解释器通常使用双缓冲技术来避免不必要的 HTML 重绘。

传统模板引擎的结构流程图 2.2 所示：首先进行词法分析，即将源代码转化为标记流；然后进行语法分析，即将标记流转化为抽象语法树；接着从后台数据库中获取数据，将数据值填充到抽象语法树中；最后，通过模板引擎解释器解释生成 HTML 文件，最终传输到客户端渲染显示。

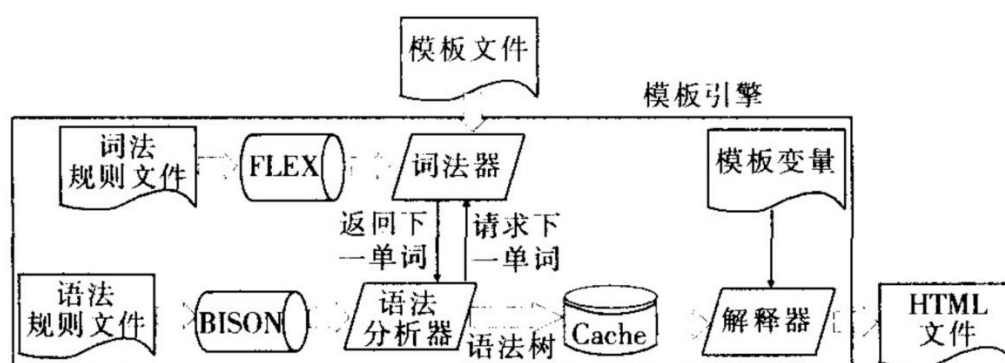


图 2.2 模板引擎的框架结构图^[23]

随着 Web 应用程序的需求越来越复杂，现代的模板引擎除了基本的词法器、语法器和解释器之外，还包括了各种性能优化、缓存、分布式部署等功能。

模板引擎的使用已经遍布各种 Web 开发框架，例如 Angular、React、Vue.js 等等。尽管不同的模板引擎实现方式不同，但其基本原理和使用方法都非常相似。模板引擎作为一种有效的技术手段，可以帮助我们更加轻松地实现 Web 应用程序的开发和维护，提高开发效率和代码质量，从而更好地满足业务需求和用户需求。

在静态网站生成器的使用过程中，开发者需要定义一些模板，包括页面框架、数据注入位置和数据展示方式等。这些模板可以使用模板引擎技术进行操作，从而生成最终的静态页面。具体来说，开发者通常会采用一些本地或云端的工具来实现 SSG 功能，而这些工具往往也自带一些内置的模板引擎工具，可以轻松实现数据的填充和渲染。例如，Hugo 静态网站生成器中包含了自带的模板引擎，开发者可以根据自己的需求进行选择和配置。

2.1.3 在 Hugo 中模板引擎的实现

Hugo 是一种静态网站生成器 SSG，采用 Go 语言程序设计，可快速构建静态网站。Hugo 使用的是 Go 语言编写的模板引擎，名为"Go Templates"。

Go Templates 的出现是建立在"html/template"包之上的。"html/template"包提供了稳定、安全的基础模板引擎，然后 Go Templates 则在此基础上添加了更多的易用特性，以满足开发人员更灵活、更高效的需求。

"html/template"包是 Go 标准库中封装的一种模板引擎，用于渲染 HTML 格式的文本。它提供了一套类似于 Go 语法的模板语言，支持条件判断、迭代、变量赋值等基本操作，同时也支持在 HTML 中嵌入 CSS、JavaScript 等内容。"html/template"包的特点是安全可靠，能够有效防止跨站脚本攻击等安全问题。这个模板引擎采用了基于文本替换的方式，将模板中的占位符替换为数据，生成最终的 HTML 文本。与其他基于字符串替换的模板引擎不同的是，"html/template"包支持自动转义，可以防止一些常见的 Web 安全问题，例如跨站脚本攻击（XSS）等。此外，"html/template"包还支持嵌套模板、条件语句、循环语句等高级特性，可以帮助用户快速构建复杂的网站。

Go Templates 在语法上简化了"html/template"的一些不必要的细节，使得模板语法更加易于理解和使用。相比较其他一些模板引擎，如 Jinja 等，Go Templates 的语法更加简单直接。Go Template 的语法结构主要包括以下两部分：

Define: 模板定义，即模板文本的放置位置，可以是字符串、文件等。模板定义类似于 Jinja2 中的模板文件，定义了每个模板中可以插入的内容。

Execute: 模板执行，即模板生成的过程。执行时需要传入相应的数据，用来填充定义好的模板。

模板引擎的基本使用流程一般包括三个步骤：解析数据、应用合并和输出。

在 Go 语言中，使用 `template` 包可以创建模板对象，并创建一个包含空 `common` 结构的模板（Common Group），然后通过 `Parse` 函数（`parseFuncs`）将需要使用的模板文件解析成模板对象，最后通过 `Execute` 函数（`execFuncs`）将数据填充到模板中，形成最终的 HTML 页面代码。相应结构如图 2.3 显示：

第一步：解析（Parsing）：当定义模板完成后，需要进行标记解析，并生成相应的模板对象。此时，模板文件被解析成抽象语法树（AST），包含了所有的

结构信息。这个过程由 Go 语言的标准库实现，主要有 `text/template` 和 `html/template` 两个包。

第二步：执行（Execution）：模板对象生成之后，可以通过向其传入参数来生成最终文本，即将指定的数据和模板进行匹配并进行处理。一旦传入了所需的参数之后，模板会从解析的 AST 中查找并获取相应的信息，然后进行处理并输出给用户。

第三步：输出（Output）：最后一步是将执行结果发送到 IO 流或者文件等输出。此时，模板已经使用传入的数据渲染好了 HTML 或者 XML 内容，如果一切正常，这时可将结果输出到浏览器或者文件等地方。

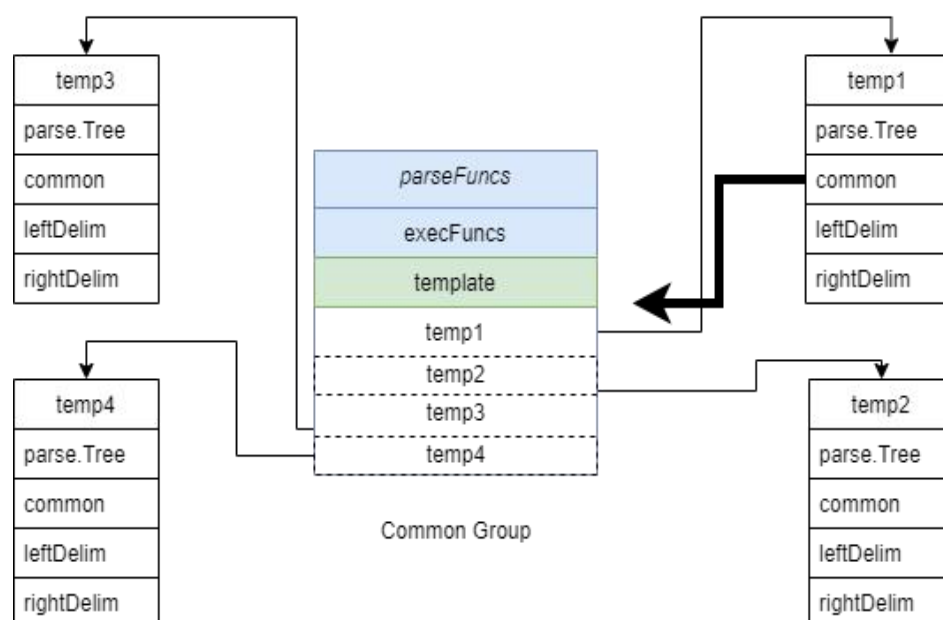


图 2.3 Go template 模板结构图

总体来说，Go Template 是 Go 语言搭建 Web 应用的必备工具，它既优雅，又简洁；能够真正做到实现设计模式中的 MVC 思想，将数据和模板分离，简化了 Web 应用的开发流程。Hugo 的模板引擎继承了 Go Templates 的全部特性，同时又进行了一些定制和扩展，比如增加了一些额外的控制函数和内置函数，以更好地满足网站生成的需求。Go Templates 的语法清晰、简洁、易学易懂，因此逐渐成为 Hugo 的默认模板引擎。在 Hugo 的模板文件中，开发者可以像使用普通 HTML 标签一样轻松地使用 Go Templates 的语法，如条件判断、循环、变量定义等，还能自定义函数等等，方便开发者更快速、有效地完成网站的开发与维护。

2.2 静态网页的自动化部署

静态网站部署的原理是将网站的静态资源（如 HTML、CSS、JavaScript、图片、视频等文件）上传到一个 Web 服务器上，并将这些文件存储在服务器的文件系统中。当用户访问网站时，Web 服务器会将这些静态资源发送给用户的浏览器，从而呈现出网站的内容和样式。具体来说，静态网站部署的过程包括以下几个步骤：

1. 编写静态网站的源代码：静态网站的源代码通常由 HTML、CSS、JavaScript 等文件组成，可以使用文本编辑器或专门的静态网站生成器来编写。
2. 将源代码构建为静态文件：使用静态网站生成器（如 Hugo、Jekyll、Gatsby 等）将源代码构建为静态文件，并生成一个包含所有静态资源的目录。
3. 上传静态文件到 Web 服务器：将生成的静态文件上传到 Web 服务器上，并将这些文件存储在服务器的文件系统中。
4. 配置 Web 服务器：配置 Web 服务器，使其能够正确地响应用户的请求，并将静态资源发送给用户的浏览器。通常情况下，可以使用 Nginx、Apache 等 Web 服务器来完成这个过程。
5. 测试网站：测试静态网站是否能够正常访问和显示，以确保用户能够正常地访问网站。

总之，静态网页自动化部署是指使用自动化工具和脚本来将静态网页部署到服务器上。这个过程可以实现快速、精准、可重复的网页发布，从而节省了人工时间和减少了人工出错的可能性。

CI/CD 在静态网站部署中扮演着非常重要的角色，可以自动化构建和部署静态网站，快速迭代和发布，测试和质量保证，管理和监控网站的部署和运行情况

2.2.1 CI/CD 流程

CI/CD 是一种基于敏捷开发原则的软件开发和交付方法。在传统的软件开发中，开发和运维是分离的，会导致代码稳定性不高、交付延迟和人为错误等问题。而 CI/CD 则通过自动化、持续集成、持续交付、持续部署等手段，打破了传统开发流程的限制，加快了开发周期，并促进了软件的稳定性和质量。

CI/CD 的管道包括开发、编译、测试、部署和监控等环节，每个环节都需要

符合持续自动化和监控的原则，确保代码质量和应用的稳定性。运维人员需要与开发团队紧密合作，确保持续交付和持续部署的顺利进行。CI/CD 管道的优点是实现了快速交付和反馈，加快了开发和部署的速度，提高了软件的质量和用户满意度。

CI/CD 流程通常包括以下几个步骤，其关系如图 2.4 所示：

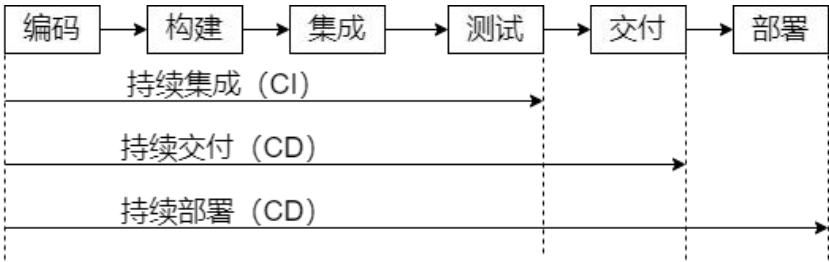


图 2.4 CI/CD 流程图

1.持续集成：持续集成是指将代码集成到主干分支中，并自动运行测试用例和代码质量检查，以确保代码的质量和稳定性。每当有新的代码提交时就会自动地进行构建、集成和测试。在 CI / CD 过程中，多个开发人员都可以在同一个代码库上工作，他们的代码的变更被 push 到同一个代码库上后，持续集成服务器会自动集成这些代码并执行测试。

2.持续交付：持续交付是指将代码部署到测试环境中，并进行自动化测试和验收测试，以确保代码的功能和性能符合要求。开发人员每次提交代码后，这个代码会经过自动化的构建，测试和部署过程，并被自动地打包成可部署的代码包。这个过程可以确保软件是稳定可靠的，可以自动化地部署到预生产环境中供测试。

3.持续部署：持续部署是持将代码部署到生产环境中，并进行自动化测试和验收测试，以确保代码的功能和性能符合要求，并且不会影响现有的业务流程。它是持续交付的一个扩展，将可部署的代码包自动化地部署到生产环境中。持续部署的过程经过多个别的测试步骤，只需要一个触发点就可以部署代码。

在 CI/CD 过程中，持续集成是关键环节，它指的是开发人员在代码变更后，将代码自动更新到代码库，然后发起一系列自动化测试、编译和构建过程。如果测试未通过，就会自动回滚到之前版本，以确保软件稳定性和质量。持续交付和持续部署则是 CI/CD 的扩展，它们允许开发人员在完成持续集成后自动地将代码部署到生产环境中，包括灰度发布、AB 测试等。

2.2.2 GitHub Actions 和 GitHub Pages

GitHub Actions 是 GitHub 提供的 CI/CD 自动化部署服务的平台，它可以帮助开发者更好的实现自动构建、自动测试、自动部署等功能。这些功能和优势，都为开发高质量和可维护的软件提供了有力的支持和保障。相对于传统的持续集成服务，GitHub Actions 具有以下优势：

1. 与 GitHub 高度整合：与 Source Control（源代码控制）紧密结合，更方便的集成到开发流程中。

2. 配置优异的虚拟服务器环境：GitHub Actions 提供了很多不同环境，包括 Linux, Windows, MacOS 等，可以进行构建、测试、打包、部署项目，且不需要购买服务器。这意味着开发者可以使用几乎与实际生产环境相同的环境对代码进行集成测试，并且可以避免环境因素带来的问题。

3. 云环境性能优异：GitHub Actions 提供了很多可选的环境和定制化配置。在持续集成环境中，GitHub Actions 相对其他云环境具有较高的性能表现和稳定性。

在 GitHub Actions 工作流中，每个作业都需要一个运行器来执行任务，运行器是执行工作流作业的计算机或虚拟机。在 GitHub 上使用运行器时，用户需要指定所使用的运行器。指定运行器后，GitHub Actions 会自动在指定的运行器上执行相应的任务，例如编译、测试、构建和部署等任务。GitHub Actions 中的运行器不仅可以执行 GitHub 上的任务，还可以配置为在本地进行开发和调试，以提高开发效率。

在实际运行过程中，GitHub Actions 会在运行任务时创建一台虚拟机，然后将用户编写的自动化代码下载至虚拟机中进行运行。平台会根据用户规定的任务流程来逐一执行每个任务，如果执行过程中遇到错误就会停止执行，并向用户反馈错误信息。如果全部任务都执行完毕并且没有发现错误，GitHub Actions 会自动清理环境并将结果反馈给用户。用户通过编写一个包含任务流程的 workflow 文件来规定需要执行的任务，而平台会在指定触发时机（如提交代码、发起 PR 请求等）时在虚拟机上运行这些任务来检查代码的正确性。用户可以随时查看代码运行到哪一步，以及在哪一步出现问题等信息，以便于进行调试和改进。流程如图 2.5 所示：

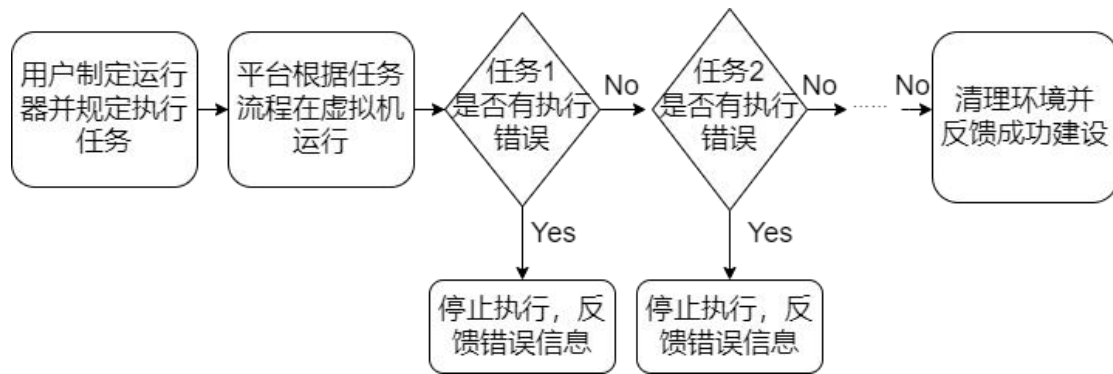


图 2.5 GitHub Actions 运行流程图

与此同时，GitHub Pages 是 GitHub 提供的一项免费的静态网站托管服务，可以将静态网页、博客、文档等发布到 GitHub 上，方便快捷地分享和发布内容。用户可以使用 Hugo 等静态网站生成器来生成网站内容，然后将生成的网站文件推送到 GitHub 仓库中，GitHub Pages 会自动将这些文件托管到互联网上。用户只需要在代码仓库中创建特定的 `gh-pages` 分支，将网站的静态资源和配置文件放置在该分支上，GitHub 就会自动将该静态网站部署到公共互联网上。同时，GitHub Pages 支持自定义域名，可以将自己的网站绑定到自己的域名上。

综上所述，GitHub Pages 和 GitHub Actions 都是 GitHub 提供的有力工具，这两个服务之间存在着互相支持和依赖的关系。在实际应用中，GitHub Actions 可以结合 GitHub Pages 使用，实现自动化构建和部署静态网站。GitHub Actions 可以通过自动化构建的方式自动将网站代码部署到 GitHub Pages 上，实现自动化发布静态网站的目的。通过这样的方式，用户可以更好地管理和发布自己的代码和所构建的静态网站。

3 系统的需求分析与设计

在 web 开发的过程中，需求分析与设计是核心组成部分，它们的重要性不言而喻。首先，对用户需求进行认真分析和设计，确保了开发出的网站符合期望，从而提高用户的满意度。其次，严格的设计有助于确保软件的可维护性和可扩展性。此外，需求分析和设计提供了一个完整的计划和蓝图，降低系统开发的风险。

本章针对 METISLAB 的需求开始分析，并进行相应的设计。确保网站的质量、可维护性和系统性，减少风险，降低成本和时间。

3.1 系统的需求分析

本论文选题来源于 METISLAB 国际联合实验室。该实验室致力于在医学成像及智能处理技术、在体传感及内窥检测技术和自主感知及柔性机器人等方向展开研究工作，并通过开放式国际实验室的方式促进国际间科研合作。该实验室由一批具有丰富经验和创新精神的研究人员组成，实验室人员的研究成果得到了国内外学术界和工业界的广泛关注和认可，包括在期刊上发表的论文、开源项目和技术博客等。在这个背景下，METISLAB 的官网是实验室对外宣传和交流的重要平台，需要提供实验室的介绍、成员、研究方向、项目、研究成果等内容，以使用户了解实验室的最新动态和研究成果，同时也方便实验室与用户进行交流与合作。因此，对 METISLAB 的官网进行功能性需求分析是十分必要和重要的。

网站的需求分析是对建立网站目标和用户需求进行深入分析的过程。本文从以下几个步骤对 METISLAB 官网建设进行分析：

1. 定义网站目标：METISLAB 的网站目标包括展示实验室的研究成果和项目、吸引更多的访问者、提高实验室的知名度和影响力、吸引潜在的合作伙伴等。为了实现这些目标，需要在网站设计中注重内容的精准定位和用户体验的提升，同时需要将网站的设计与实验室的研究方向和目标相匹配。

2. 定义目标受众：实验室官网的受众群体包括学术界、行业界、大众读者、潜在合作伙伴和投资者等。由于 METISLAB 的成员和客户来自不同的国家和地区，官网需要提供多语言支持。网站需要提供多种研究人员的交流方式，如邮箱、领英账号等，以方便不同受众的沟通和互动。

3. 定义网站功能和特性：METISLAB 的网站需要提供的功能和特性包括但不限于：实验室介绍、研究方向、项目展示、研究成果等。这些功能需要在网站设计中合理布局和组织。

4. 制定网站结构：METISLAB 的网站结构需要考虑到网站目标和功能，以及用户体验。网站的结构需要清晰明了，便于用户使用并且能够快速找到需要的信息。同时，网站的导航和布局也需要考虑到不同受众的需求和使用习惯，以提升用户体验。

5. 制定网站内容策略：METISLAB 的网站内容策略需要与目标受众一致。内容的分类和定位需要基于用户群体的需求和行为。内容需要包括文字、图像、音频和视频等各种形式，以满足不同受众的需求。同时，内容的更新和维护也需要注重实验室的研究成果和项目，以便展示实验室的优势和特色。

6. 开发技术需求：METISLAB 的网站开发需要考虑到实验室的技术实力和预算。需要选择最适合实验室的技术和平台，以便提升网站的可用性和可维护性。同时，也需要注重网站的安全性和稳定性，以保障用户的信息安全和网站的正常运行。另外，官网需要遵守隐私政策和版权规定，保证采用合法的图片、视频、音频等素材，并保证不泄露用户的个人信息。

综上所述，METISLAB 的网站需求分析需要充分考虑到实验室的研究方向和目标、目标受众、网站功能及特性、网站结构、网站内容策略和技术需求等各个方面。在具体实现过程中，需要根据实验室的实际情况进行针对性的调整和优化，以满足实验室的需求和目标

根据以上分析，可以总结出 METISLAB 网站建设中的需求：

1. 研究人员介绍清晰。
2. 实验室的研究成果和项目突出。
3. 提供多语言支持。
4. 简单明了的网站结构。
5. 适合实验室的技术和平台。
6. 遵守隐私政策和版权规定。

3.2 系统设计

为了实现系统需求，METISLAB 应在网站设计中注重内容的精准定位和用户体验的提升，同时匹配网站的设计与实验室的研究方向和目标。METISLAB 的网站作为展示实验室研究成果和项目的窗口，需要吸引更多的访问者和潜在合作伙伴，提高实验室的知名度和影响力。在这个过程中，系统设计是非常重要的的一环。

3.2.1 架构设计

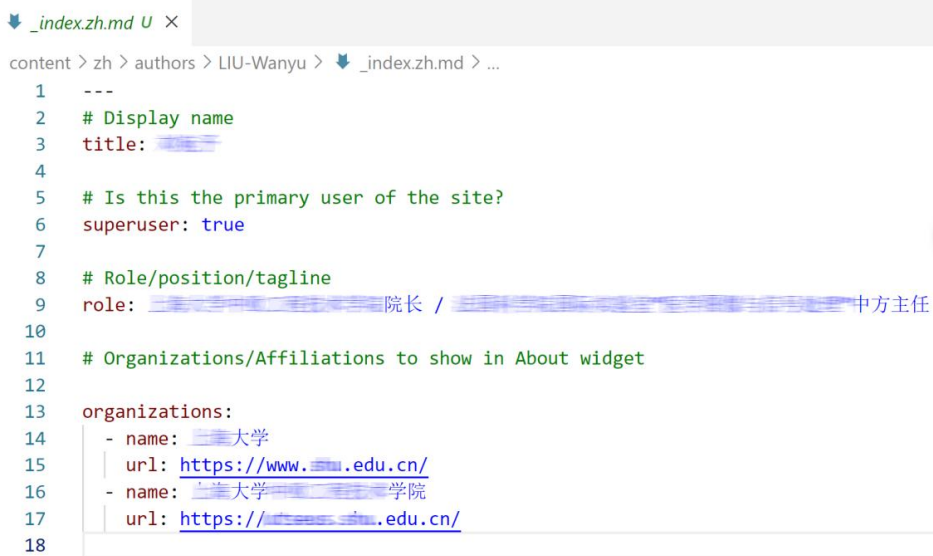
本研究主要采用 Hugo 这一静态网站生成器，实现快速构建出高效、美观、易于维护的静态网站。因此，本研究的架构设计基于 Hugo 的架构设计，主要包括以下几个方面：

1. 文件结构：为了使网站的管理和维护变得更加简单和直观，METISLAB 网站文件结构是扁平化的。扁平化的网站文件结构是一种常见的设计方式，它将所有的内容和模板都放在同一个文件夹中，而不是将它们分散在多个文件夹中。这种设计方式的优点在于，它可以使网站的管理和维护变得更加简单和直观。具体而言，扁平化的文件结构可以帮助网站管理员更快地找到需要修改的文件，而不需要在多个文件夹中进行查找。此外，扁平化的文件结构也可以简化网站的备份和迁移，因为所有的文件都在同一个文件夹中，可以很方便地进行整体复制和移动。本研究的文件结构如图 3.1 所示：



图 3.1 文件结构图

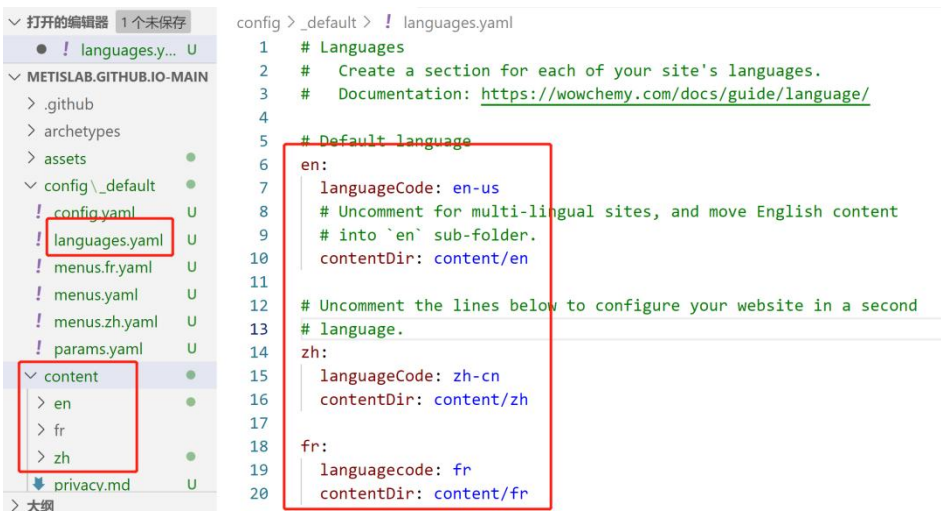
2. 内容管理：本网站的内容管理采用 Markdown 格式，这种格式简单易懂，同时又可以轻松地为 HTML 格式。每个 Markdown 文件都对应着一个网页或一篇文章。这些文件可以存放在不同的文件夹中，以便进行分类和管理。在创建 Markdown 文件时，需要指定一些元数据，例如标题 title、职称 role、组织 organization 等。同时，网站还支持多种内容类型，如文章、页面、图像等，皆存放于 content 这一文件夹内。本研究中， Markdown 文件如图 3.1 所示：



```
content > zh > authors > LIU-Wanyu > _index.zh.md > ...
1 ---
2 # Display name
3 title: 
4 
5 # Is this the primary user of the site?
6 superuser: true
7 
8 # Role/position/tagline
9 role: 院长 / 中方主任
10 
11 # Organizations/Affiliations to show in About widget
12 
13 organizations:
14   - name: 大学
15     url: https://www. .edu.cn/
16   - name: 大学 学院
17     url: https:// .edu.cn/
18
```

图 3.2 文件内容管理

元数据可以用 YAML、TOML 或 JSON 格式来表示。在本研究中，采用 YAML 格式来定义。例如，多语言支持的实现，是通过在 yaml 文件中定义不同语言的元数据，然后在相应的 content 里使用不同语言添加相应内容来实现的，如下图 3.3 所示：



```
config > _default > ! languages.yaml
1 # Languages
2 # Create a section for each of your site's languages.
3 # Documentation: https://wowchemy.com/docs/guide/language/
4 
5 # Default language
6 en:
7   languageCode: en-us
8   # Uncomment for multi-lingual sites, and move English content
9   # into `en` sub-folder.
10   contentDir: content/en
11 
12 # Uncomment the lines below to configure your website in a second
13 # language.
14 zh:
15   languageCode: zh-cn
16   contentDir: content/zh
17 
18 fr:
19   languagecode: fr
20   contentDir: content/fr
```

图 3.3 元数据定义（以多语言为例）

3. 主题系统：本系统的主题使用的是基于 Hugo 的框架主题 wowchemy 和 wowchemy-cms，其中包含一组预定义的模板和样式，可以轻松定制网站的外观和功能，而无需编写复杂的 HTML 和 CSS 代码。主题系统是基于 Go 的模板语言和 HTML/CSS/JavaScript 技术实现的。

主题的安装是将主题文件夹复制到 METISLAB 网站的 module 目录下。然后在网站的配置文件中指定使用的主题名称。文件夹中包含了多个模板文件，例如首页模板、文章列表模板、文章详情模板等，开发者和维护者根据需要进行修改。

选择 Wowchemy 作为网站模板的原因主要有以下几点：

（1）面向科学研究人员和学术机构：METISLAB 是一个科研实验室，面向科学研究人员和学术机构的网站模板能更好的适应网站内容。相比其他的 Hugo 模板，Wowchemy 更加专注于学术领域，提供了更多的科研相关的功能和元素。

（2）可视化的内容管理系统：Wowchemy 提供了一个可视化的内容管理系统，可以帮助用户更方便地管理网站的内容和主题。METISLAB 的成员可能不是专业的网站开发人员，因此需要一个易用的网站模板。相比其他的 Hugo 模板，Wowchemy 更加注重用户体验，提供了更加友好和易用的用户界面。

（3）多样化的布局和元素：Wowchemy 提供了多样化的布局和元素，可以帮助用户创建具有个性化和独特性的网站。METISLAB 需要一个能够展示实验室研究成果和项目经验的网站模板。Wowchemy 提供了多种不同的文章和项目布局，帮助用户更好地展示自己的研究成果和项目经验。

综上所述，Wowchemy 是一个专门针对学术领域的 Hugo 模板，提供了可视化的内容管理系统和多样化的布局和元素，非常适合 METISLAB 这样的科研实验室使用。

4. 构建和部署：本系统的构建和部署借助了 GitHub Actions 和 GitHub Pages。主要过程如下：

（1）在 GitHub 上创建一个仓库来存储网站的源代码。在仓库中，需要包含 Hugo 的配置文件和 Markdown 文件等网站内容。可以使用 Git 命令将本地的网站源代码推送到 GitHub 仓库中。

（2）配置 GitHub Actions 来自动构建网站。GitHub Actions 可以在代码推送到仓库后自动运行一系列的操作。使用 GitHub Actions 来自动构建网站并将构建好的网站文件推送到 GitHub Pages 上。具体来说，在仓库中创建一个名为 `.github/workflows/main.yml` 的文件，用来配置 GitHub Actions。在该文件中，需要指定构建网站的步骤和部署网站的步骤。

（3）使用 Hugo 命令构建网站并推送至分支。使用 `hugo --minify` 命令来

生成压缩后的网站文件。构建完成后，可以将网站文件推送到 `gh-pages` 分支上。

（4）在 GitHub Pages 的设置中启用网站，并指定 `gh-pages` 分支作为网站的源。这样，当代码推送到仓库后，GitHub Actions 就会自动构建和部署网站，网站的更新会实时展现在 GitHub Pages 上。

总之，METISLAB 网站通过 GitHub Actions 和 GitHub Pages 进行构建和部署。通过配置 GitHub Actions，可以实现自动构建和部署网站的功能，大大简化了网站发布的过程。同时，使用 GitHub Pages 还可以免费地托管网站，非常适合个人和小型团队使用。

3.2.2 模块设计

为了满足 METISLAB 的需求，网站的模块设计需要考虑以下几点。首先，信息架构应该清晰明了，以便用户快速找到所需的信息；其次，网站需要提供丰富的内容展示，并且应该提供多种展示方式，以便用户能够更好地获取所需信息；此外，用户体验需要友好和易用，网站的设计风格 and 布局应该统一，而且也要考虑到不同设备和浏览器的兼容性，确保网站能够在各种设备上正常显示和使用。

1. 导航栏模块：METISLAB 的导航栏模块能够清晰明了地展示实验室的主要模块和页面，同时也要考虑到用户体验和网站的可读性。通过收起/展开功能、搜索栏和语言选择等功能来提高用户体验和网站的可用性。METISLAB 的导航栏模块如图 3.4 所示：



图 3.4 导航栏模块（展开状态）

（1）Logo：Logo 是网站的标识，放置在导航栏的中央位置，方便用户快速识别和记忆。

（2）导航菜单：导航菜单是网站的核心功能，包括实验室的主要模块和页面。根据需求分析，METISLAB 的导航菜单包括负责人、中方团队、法方团队、我们的学生、发展历程、承担项目和发表论文等模块。

为了避免导航栏过于拥挤和混乱，当用户使用竖屏页面或手机端时，导航菜单将自动收起，可以通过点击图标展开，从而提高用户体验和网站的可读性。收起状态如图 3.5 所示：

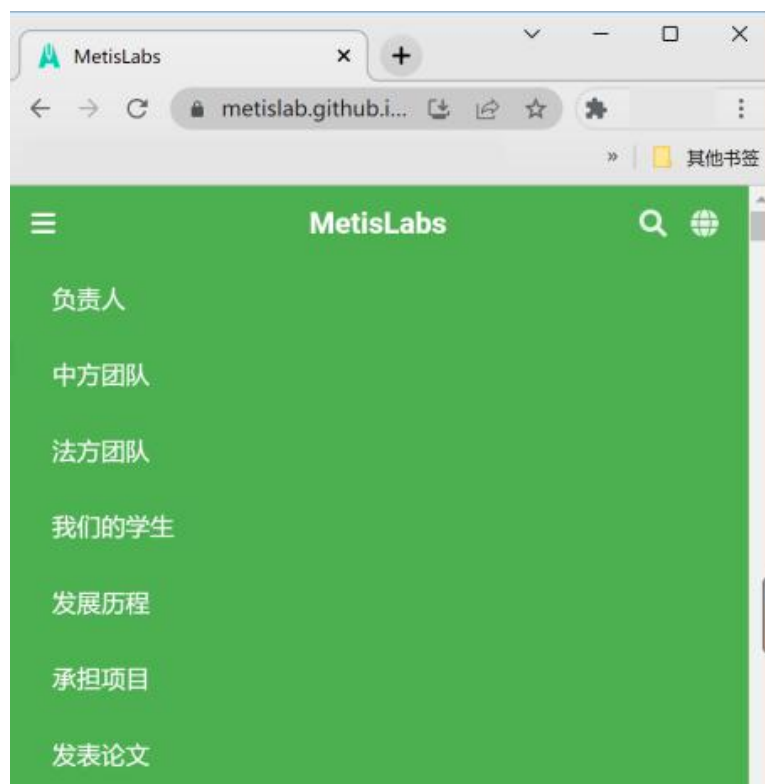


图 3.5 导航栏模块（收起状态）

（3）搜索栏：搜索栏可以帮助用户快速搜索和定位所需的信息。搜索栏被放置在导航栏的右侧，方便用户进行搜索操作。

（4）语言选择：语言选择提供用户中文、英文和法语三种语言版本。语言选择放置在导航栏的右侧，方便用户进行切换。

2. 展示页模块：METISLAB 的展示页提供了实验室的基本信息、研究方向等内容。为了提高页面的可读性和吸引用户的注意力，采用自动或手动轮播的方式来展示以上内容。同时也可以提供前进、后退等功能，方便用户进行浏览和操作。为了提高页面的可视性和可读性，将图片和文字进行结合展示，以便用户更加直观地了解实验室的研究方向和成果。展示页模块如图 3.6 所示：



图 3.6 展示页模块

3. 人员模块：METISLAB 官网需要展示实验室成员的信息，包括姓名、职称、研究方向等内容。在首页点击每位成员，会进入其相应的详情页，其中包括实验室成员的简介、照片、研究方向以及联系方式等。人员模块如图 3.7 所示：

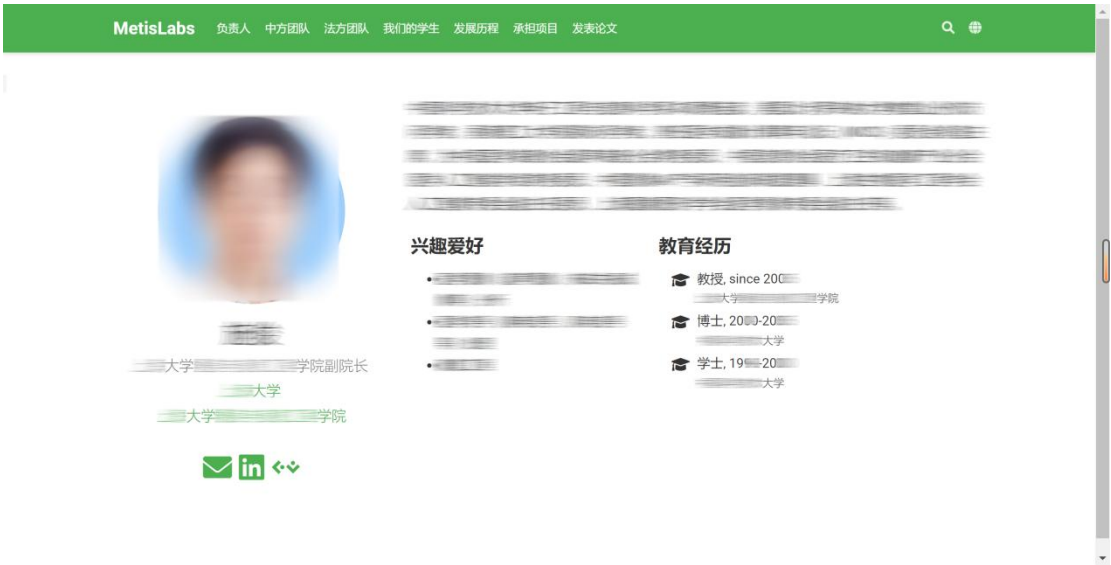


图 3.7 人员模块详情页

4. 发展历程模块：该模块展示 METISLAB 的发展过程中的关键时间和事件，将实验室的发展历程按照时间顺序进行排列，并在时间轴上标注相应的事件和时间节点。为了让用户更加直观地了解实验室的发展历程，在时间线上添加图片、视频、音频等多媒体元素，以提高页面的可视性和吸引力。发展历程模块如图 3.8 所示：

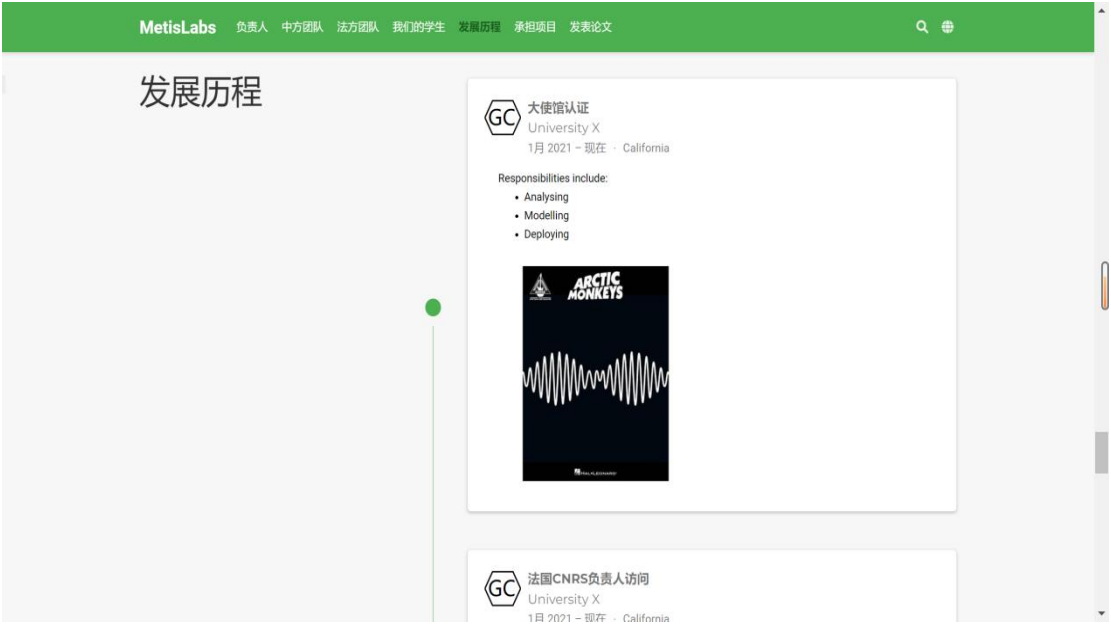


图 3.8 发展历程模块

5. 承担项目模块：该模块展示实验室所承担的最新项目。为了让用户更加清晰地了解实验室的承担项目，将该板块分成几个主要的类别进行展示，如深度学习、相关演讲等，以使用户更加系统地了解实验室的承担项目和相关的演讲等内容信息。承担项目模块如图 3.9 所示：

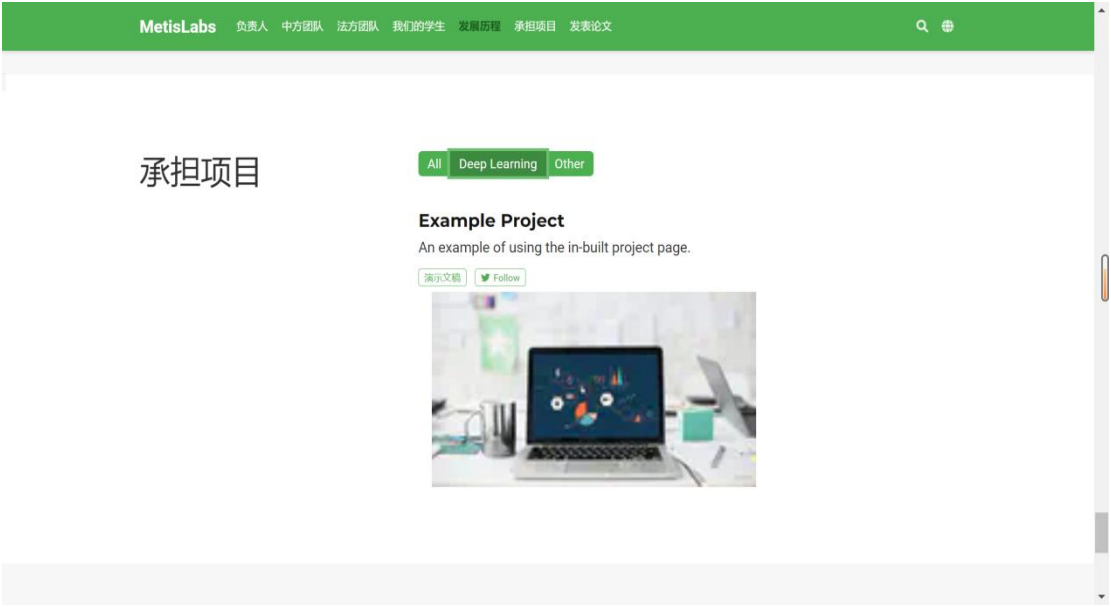


图 3.9 承担项目模块

点击可进入相应的介绍，如图 3.10 所示：

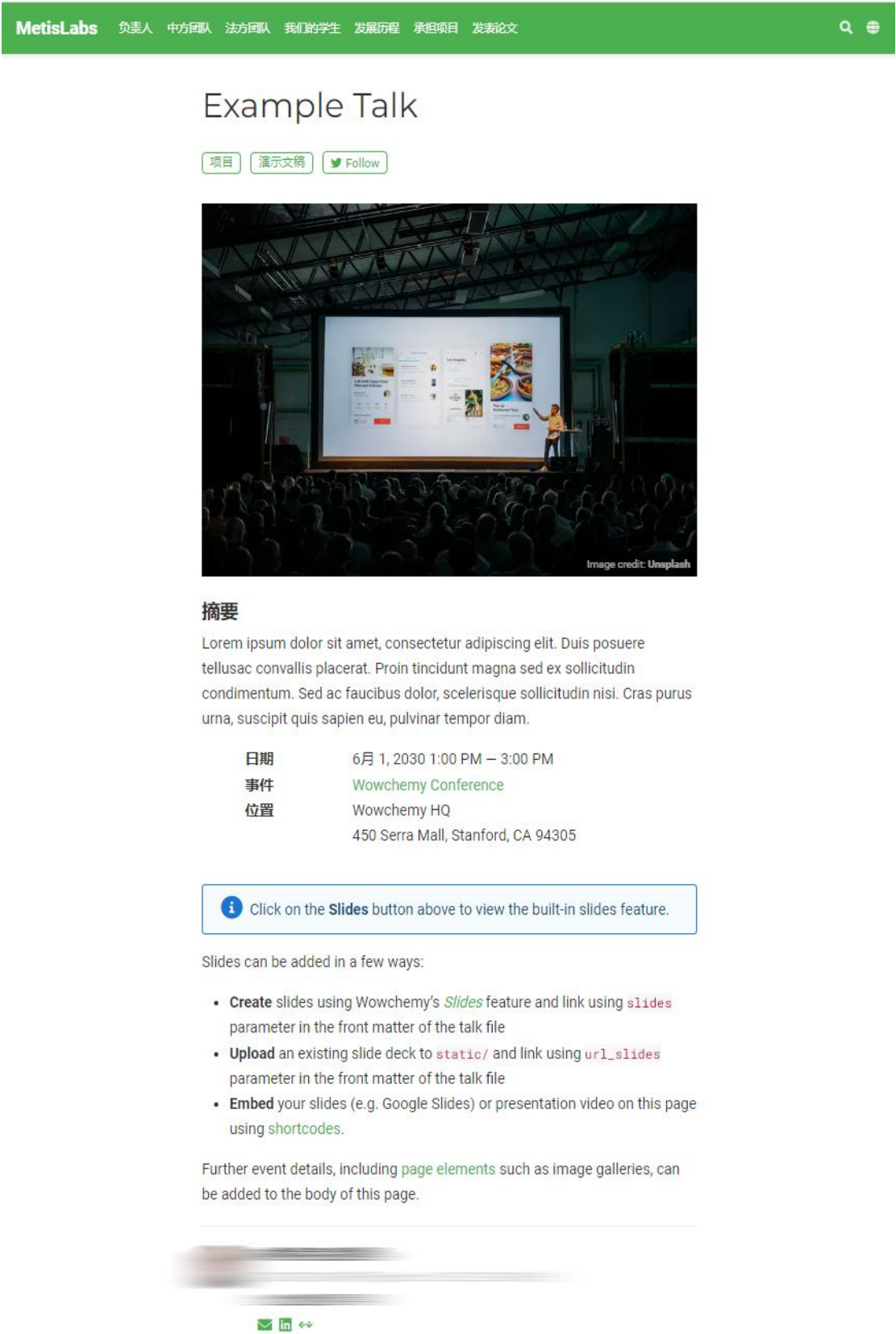


图 3.10 承担项目模块详情页

6. 发表论文模块：该模块用来展示实验室成员的科研成果和发表的相关论文，以使用户了解实验室的科研水平和研究方向。发表论文模块如图 3.11 所示：



图 3.11 发表论文模块

点击可进入相应论文的详情页，展示了论文的摘要、类型和出版物等，如图 3.12 所示：



图 3.12 发表论文模块详情页

4 系统实现

4.1 环境搭建

4.1.1 Hugo

本系统的静态网页生成依靠 Hugo 来实现，下面是 Hugo 的环境搭建步骤：

1. 安装 Hugo: Hugo 可以在 Windows、macOS 和 Linux 等操作系统上运行。在 Hugo 的官方网站 (<https://gohugo.io/getting-started/installing/>) 上下载适用于操作系统的 Hugo 安装包。安装包包括二进制文件和示例网站。

2. 验证安装: 安装完成后，您可以在命令行中输入 “hugo version” 命令来验证 Hugo 是否正确安装。如果看到 Hugo 的版本号，则说明 Hugo 安装成功。

3. 添加主题: 通过对系统需求的分析，本系统使用 wowchemy 模板为佳，从官方网站下载相应的主题。在系统网站的文件夹中新建 ‘modules’ 文件夹，将下载的主题解压缩到该目录之下。然后，新建配置文件 ‘config.yaml’，在其中设置主题名称：

```
1.   module:
2.   imports:
3.     - path: GitHub.com/wowchemy/wowchemy-hugo-modules/wowchemy-cms
      /v5
4.     - path: GitHub.com/wowchemy/wowchemy-hugo-modules/wowchemy/v5
```

4. 本地预览: 使用 Hugo 进行本地预览非常简单。在命令行中，进入网站目录，并输入 “hugo server -D” 命令，就可以启动 Hugo 的本地预览服务器，并在浏览器中打开网站。可以在浏览器中查看网站的效果，并进行调试和修改。

4.1.2 GitHub Actions

本系统使用 Git Actions 实现网页自动化构建，通过创建一个 yaml 文件，来定义工作流。本系统工作流包含一个名为 build-deploy 的作业，该作业在 Ubuntu 系统上运行，包含四个步骤：检出代码、设置 Hugo、构建代码和部署网页。

```
1.   name: main
2.   on:
3.     push:
4.       branches:
5.         - main
```

```

6.   jobs:
7.     build-deploy:
8.       runs-on: ubuntu-20.04 // 虚拟系统
9.       steps:
10.        - uses: Actions/checkout@main //检出代码
11.          with:
12.            submodules: true
13.        - name: Setup Hugo //设置 hugo
14.          uses: peaceiris/Actions-hugo@v2
15.          with:
16.            hugo-version: '0.98.0'
17.            extended: true
18.        - name: Build //构建代码
19.          run: hugo --minify --baseURL=https://metislab.github.io
20.        - name: Deploy //部署网页
21.          uses: peaceiris/actions-gh-pages@v3
22.          with:
23.            deploy_key: ${ secrets.actions_DEPLOY_KEY }
24.            publish_branch: gh-pages
25.            publish_dir: ./public
26.            allow_empty_commit: true

```

4.2 系统功能实现

回顾上文的需求分析，可以总结 METISLAB 网站的需求是：简单明了的网站结构；研究人员介绍清晰；实验室的研究成果和项目突出；提供多语言支持；适合实验室的技术和平台；遵守隐私政策和版权规定。

考虑到实验室人员的计算机技术，选择了简单、易上手的 Hugo 静态网站构建器，并采用 Wowchemy 这一专门针对学术领域的 Hugo 模板进行构建。在网站的部署，交给了 GitHub Actions 并直接部署至 GitHub Pages，实现自动化构建。

在隐私方面，本系统采用静态网站，静态网站相对于动态网站来说，没有后端服务器和数据库，通常不会涉及到用户的个人信息和隐私数据的存储和处理，因此在隐私问题上相对更安全。在版权方面，采用 MIT 许可，本软件和相关文档文件的副本，可以不受限制地处理本软件，并注明在 LICENSE.md 文件中。

针对其余的需求，将会在本章节进行详述。其中，网站结构为 4.2.1 章；研究人员、研究成果和项目皆属于网站内容，位于 4.2.2 章；多语言支持在 4.2.3 章进行介绍。

4.2.1 网站结构

网站结构如 3.2 章设计所示，分为导航栏、展示页、人员、发展历程、承担项目和发表论文这七个模块，其中人员分为：负责人、中方团队、法方团队和学生四个部分。通过点击导航栏中的标签可以跳转至相应模块。结构简单明了，可供网页浏览者精准定位。

网站的顶部栏和底部栏被定义在了`params.yaml`文件中，该文件是用于定义网站全局参数的文件。具体来说，`params.yaml`文件包含了一些全局参数，例如网站的标题、副标题、作者、描述、语言、主题等。

以下是`params.yaml`文件的部分内容，主要显示顶部栏和底部栏设置：

```

1.   # Appearance
2.   appearance:
3.     theme_day: Forest //符合实验室定位和 logo 设计，设置为相契合的绿色
4.     font: Minimal //字体
5.     font_size: L //字体大小
6.   # Site header
7.   header:
8.     navbar: //导航栏
9.       enable: true
10.    align: 1
11.    show_logo: true //logo 展示，使用代码链接到设计的 logo
12.    show_day_night: false
13.    show_search: true //搜索框
14.    highlight_active_link: true
15.  # Site footer
16.  footer:
17.    copyright:
18.      notice: '© {year} METISLAB.' //版权声明
19.      license:
20.        enable: false
21.        allow_derivatives: false
22.        share_alike: false

```

在上面的代码中，`params.yaml`文件定义了顶部栏和底部栏分别所要展示的内容，顶部栏展示 logo 和搜索栏，底部栏展示版权所有。

此外，本系统中，通过对名为`menus.yaml`的文件进行更改来定义网站菜单的参数。`menus.yaml`文件位于`config_default`目录下，用于定义网站的主菜单、底部菜单、语言菜单等。每个菜单项都可以包含一个链接、一个图标和一个名称。以下是中文页面的`menus.yaml`文件：


```
1.   main:
2.     - name: 负责人
3.       url: '#PI'
4.       weight: 14
5.     - name: 中方团队
6.       url: '#teamcn'
7.       weight: 20
8.     - name: 法方团队
9.       url: '#teamfr'
10.      weight: 22
11.    - name: 我们的学生
12.      url: '#students'
13.      weight: 25
14.    - name: 发展历程
15.      url: '#experience'
16.      weight: 28
17.    - name: 承担项目
18.      url: '#projects'
19.      weight: 50
20.    - name: 发表论文
21.      url: '#publications'
22.      weight: 60
```

在上面的文件中，`menu` 参数定义了一个菜单：`main`。菜单包含了网站的主要内容，例如负责人、中方团队、发展历程、承担项目、发表论文等。

其中，`weight` 是用于控制页面和内容的排序顺序的参数。具体来说，`weight` 参数是一个数字，用于指定页面和内容的排序顺序。较小的 `weight` 值排在较大的 `weight` 值之前。在数据文件中，可以在每个条目中添加 `weight` 参数，如先前代码所示。在 Markdown 文件中，在文件的开头添加 `weight` 参数，例如：

```
1.   widget: people
2.   active: true
3.   headless: true # This file represents a page section.
4.   weight: 20
5.   title: 中方团队
```

总之，通过调整 `weight` 参数，可以更改页面和内容的排序顺序，从而按照实验室需求顺序依次展示所需内容。

4.2.2 网站内容

在本系统建设的网页中,通过修改网站的配置文件,来添加和编辑网站内容。

1. “研究人员”模块:

首先,定义人员页面的 `html` 文件,这个文件的目的是为作者简介页提供一个统一的布局,并显示关于作者及其作品的相关信息。

```

1.  {{- define "main" -}}
2.  {{/* 作者简介页。 */}}
3.  {{/* 如果没有为该用户创建账户, 只需显示他们的名字作为标题。 */}}
4.  {{ if not .File }}
5.  <div class="universal-wrapper pt-3">
6.    <h1>{{ .Title }}</h1>
7.  </div>
8.  {{ end }}
9.  <section id="profile-page" class="pt-5">
10.    <div class="container">
11.      {{/* 如果该用户存在一个账户, 则显示 "关于 "小组件。 */}}
12.      {{ if .File }}
13.        {{ $widget := "widgets/about.html" }}
14.        {{ $username := (path.Base .File.Dir) }}
15.        {{ $params := dict "root" $ "page" . "author" $username }}
16.        {{ partial $widget $params }}
17.      {{end}}
18.      {{ $query := where .Pages ".IsNode" false }}
19.      {{ $count := len $query }}
20.      {{ if $count }}
21.        <div class="article-widget content-widget-hr">
22.          <h3>{{ i18n "user_profile_latest" | default "Latest" }}</h3>
23.          <ul>
24.            {{ range $query }}
25.              <li>
26.                <a href="{{ .RelPermalink }}">{{ .Title }}</a>
27.              </li>
28.            {{ end }}
29.          </ul>
30.        </div>
31.      {{ end }}
32.    </div>
33.  </section>
34.  {{- end -}}
```

在本系统中，定义了一个名为 `authors` 的变量，其中包含了所有人员的信息。在网页的 HTML 代码中，通过以下代码来访问 `authors` 变量：

```
1.    {{ range .Site.Data.authors }}
2.    <!-- 显示人员信息的代码 -->
3.    {{ end }}
```

以上代码会遍历 `authors` 变量中的所有人员信息，并在每个人员信息上执行一次循环。在循环中，通过遍历 `authors` 变量中的所有人员信息，并在每个人员信息上执行一次循环，可以显示每个人员的信息。这样，用户可以方便地添加和管理网站内容，同时也可以保证网站的一致性和可维护性。在循环中，使用以下代码来访问每个人员的信息：

```
1.    {{ range .Site.Data.authors }}
2.    <h2>{{ .name }}</h2>
3.    <p>{{ .bio }}</p>
4.    
5.    {{ end }}
```

这段代码会显示每个人员的姓名、简介和头像。在网站的配置文件中对 `authors` 变量进行定义，添加、删除或修改标签种类。

```
1.    - name: authors
2.    label: Authors
3.    label_singular: Author
4.
5.    # 文件存放目录与位置
6.    folder: 'content/authors'
7.    path: '{{slug}}/_index'
8.    filter: {field: "cms_exclude"}
9.    create: true # Allow users to create new documents in this collection
10.
11.    # 领域的定义
12.    fields: # The fields each document in this collection have
13.        - {label: "Display name (such as your full name)", name: "title", widget: "string"}
14.        - {label: "Position or tagline (such as Professor of AI)", name: "role", widget: "string", required: false}
15.
16.    # 头像的定义
17.        - label: "Avatar (upload an image named `avatar.jpg/png`)"
```

```

18.     name: "avatar_filename"
19.     widget: "image"
20.     default: "avatar"
21.     required: false
22.     media_library:
23.         config:
24.             multiple: false
25.
26.     # 标签元素的定义
27.     - {label: "Short biography (shown in author boxes)", name: "bio", widget: "string", required: false}
28.     - {label: "Full biography (shown in About widget)", name: "body", widget: "markdown", required: false}
29.     - label: "Interests (shown in About widget)"
30.     name: "interests"
31.     required: false
32.     widget: "list"

```

接着，新建研究人员文件夹，构建一个 index.md 文件用于存放人员信息和一张头像图片，并命名为 avatar.jpg。随后，在 `index.md` 文件中添加人员的信息。

在本系统中，添加了 50 多位研究人员及其学生的基本信息与相应头像照片，另外，系统提供中、英、法三种语言的版本，共计两百余个 Markdown 文件。

例如，要添加成员信息，在文件中添加以下代码（个人信息已进行模糊，仅作为格式示例）：

```

1.     # 显示姓名
2.     title: XXX
3.     # 角色/职位/标签词
4.     role: XX 大学 XXXX 学院院长 / 法国 XX 实验室“XX”中方主任
5.     # 在 “关于” 小部件中显示的组织/附属机构
6.     organizations:
7.         - name: XX 大学
8.           url: https://www.xx.edu.cn/
9.         - name: XX 大学 XXXX 学院
10.        url: https://xxxx.xx.edu.cn/
11.    social:
12.        - icon: envelope
13.          icon_pack: fas
14.          link: '/#contact'
15.        - icon: ciencia-vitae

```

```

16.     icon_pack: ai
17.     link: https://xxxx.shu.edu.cn/info/1124/2171.htm
18.     # 链接到简历/CV 的 PDF 文件。
19.     # 输入显示 Gravatar 的电子邮件（已在配置中启用 Gravatar）。
20.     email: 'XXX@xxx.edu.cn'
21.     # 在作者列表中突出显示作者
22.     highlight_name: true
23.
24.     user_groups:
25.     - XXXX 学院 (XX 大学)
26.     ---
27.     .....（简介）
28.
29.     ## 工作经历
30.     - 2004 年 9 月起，任 XXX 大学教授
31.     - 2001-2004 年，任 XXX 高级研究员
32.     - 1997-2001 年，任 XXX 高级工程师

```

其中，`title` 是人员的姓名，`role` 是人员的职称，`social` 是人员的社交媒体账号，包括但不限于邮箱、谷歌学术、领英账号等，`organizations` 是人员所处组织。

2. “发表论文” 模块：

在本系统中，“发表论文”模块提供了多位研究人员所发表的多篇论文，用 Go 模板语言编写显示出版物的 html 文件。

```

1.     {{- define "main" -}}
2.     {{ $pub_types := partial "functions/get_pub_types" $ }}
3.     {{ $pub_type_param := .Params.publication_types | default (slice 0) }}
4.     {{/* 将 "0"形式的字符串转换为 int (0) */}}
5.     {{ $pub_type := (int (index $pub_type_param 0)) | default 0 }}
6.     {{/* 如果定义了 Pub 类型，则验证该类型 */}}
7.     {{ if gt $pub_type (sub (len $pub_types) 1) }}
8.     {{ warnf "Unknown publication type in %s" .File.Path }}
9.     {{ $pub_type = 0 }}
10.    {{ end }}
11.    <div class="pub">
12.        {{ partial "page_header.html" . }}
13.        <div class="article-container">
14.            {{ if .Params.abstract }}
15.            <h3>{{ i18n "abstract" }}</h3>
16.            <p class="pub-abstract">{{ .Params.abstract | markdownify }}</p>

```

```

17.      {{ end }}
18.      { /* 如果该类型是未分类的，则隐藏该类型 */ }
19.      {{ if ne $pub_type 0 }}
20.      <div class="row">
21.          <div class="col-md-1"></div>
22.          <div class="col-md-10">
23.              <div class="row">
24.                  <div class="col-12 col-md-3 pub-row-heading">{{ i18n "publication_ty
pe" }}</div>
25.                  <div class="col-12 col-md-9">
26.                      <a href="{{ (site.GetPage "section" "publication").RelPermalink }}"#{{
{ $pub_type | anchorize }}">
27.                          {{ index $pub_types $pub_type }}
28.                      </a>
29.                  </div>
30.              </div>
31.          </div>
32.          <div class="col-md-1"></div>
33.      </div>
34.      <div class="d-md-none space-below"></div>
35.      {{ end }}
36.      { /* 显示标题和内容的出版物信息 */ }
37.      {{ if .Params.publication }}
38.      <div class="row">
39.          <div class="col-md-1"></div>
40.          <div class="col-md-10">
41.              <div class="row">
42.                  <div class="col-12 col-md-3 pub-row-heading">{{ i18n "publication" }}
</div>
43.                  <div class="col-12 col-md-9">{{ .Params.publication | markdownify }}
</div>
44.              </div>
45.          </div>
46.          <div class="col-md-1"></div>
47.      </div>
48.      <div class="d-md-none space-below"></div>
49.      {{ end }}
50.      <div class="space-below"></div>
51.      <div class="article-style">{{ .Content }}</div>
52.      {{ partial "page_footer" . }}
53.  </div>
54. </div>
55. {{- end -}}

```

代码通过调用一个部分函数，以获得可用的出版物类型和出版物前言中指定的出版物类型。将出版物类型从字符串转换为整数，并对其进行验证。如果出版物类型没有被识别，就会打印一个警告，并将该类型设置为 0。随后定义出版物页面的 HTML 结构。如果在前面的内容中指定了一个摘要，它就会被显示出来。如果出版物类型不是 0（未分类），就会显示出版物类型。如果在前言中指定了一个出版物，它将被显示。最后，出版物的内容与页脚一起被显示。

随后，建立每篇论文的 PDF 文件和 Markdown 文件。其中，PDF 文件提供论文，Markdown 文件定义了相应页面（个人信息已进行模糊，仅作为格式示例）：

```

1.    ---
2.    title: '论文题目'
3.
4.    authors:
5.      - 作者 A
6.      - XXX // (实验室人员)
7.      - 作者 B
8.
9.    date: '2011-03-10T00:00:00Z'
10.   doi: '10.1007/s00006-021-01120-z'
11.
12.   publication: In *XXXXXX 大学学报*
13.   publication_short: In *JHIT*
14.
15.   abstract:
16.     (论文摘要)
17.
18.   summary: (论文总结)
19.   tags: []
20.
21.   image:
22.     caption: 'Image credit: [**Unsplash**](https://unsplash.com/photos/pLCdAaMFLTE)'
23.     focal_point: "
24.     preview_only: false

```

其中，`title` 是论文的名称，`authors` 是论文的作者，点击即可进入相应作者的介绍，`date` 是论文发表时间，`publication` 是论文出版社，`abstract` 是论文摘要。

3. “承担项目”模块：

“承担项目”主要展示了实验室最近的项目信息，照片，参与人等等信息，其 html 文件定义如下：

```

1.      {{- define "main" -}}
2.      <article class="article article-project">
3.          {{ partial "page_header.html" . }}
4.          <div class="article-container">
5.              <div class="article-style">
6.                  {{ .Content }}
7.              </div>
8.              {{ partial "page_footer" . }}
9.              <div class="project-related-pages content-widget-hr">
10.                 {{ $page := . }}
11.                 {{ $project := .File.ContentBaseName }}
12.                 {{ $items := where (where site.RegularPages "Type" "post") ".Params.projects" "intersect" (slice $project) }}
13.                 {{ $count := len $items }}
14.                 {{/* 循环浏览项目，为每个项目渲染视图 */}}
15.                 {{ if ge $count 1 }}
16.                     <h2>{{ (i18n "posts") }}</h2>
17.                     {{ range $index, $item := $items }}
18.                         {{ partial "functions/render_view" (dict "page" $ "item" . "view" (site.Params.projects.post_view | default "compact") "index" $index) }}
19.                         {{ partial "functions/render_view" (dict "page" $ "item" . "view" (site.Params.projects.post_view | default "compact") "index" $index) }}
20.                     {{end}}
21.                 {{ end }}
22.                 {{/* 渲染与项目有关的出版物列表 */}}
23.                 {{ $items := where (where site.RegularPages "Type" "publication") ".Params.projects" "intersect" (slice $project) }}
24.                 {{ $pubs_len := len $items }}
25.                 {{ if ge $pubs_len 1 }}
26.                     <h2>{{ (i18n "publications") }}</h2>
27.                     {{ range $index, $item := $items }}
28.                         {{ partial "functions/render_view" (dict "page" $ "item" . "view" (site.Params.projects.publication_view | default "compact") "index" $index) }}
29.                     {{end}}
30.                 {{ end }}
31.                 {{/* 渲染具体项目页面 */}}
32.                 {{ $items := where (where site.RegularPages "Type" "event") ".Params.projects" "intersect" (slice $project) }}
33.                 {{ $talks_len := len $items }}
34.                 {{ if ge $talks_len 1 }}

```



```

35.      <h2>{{ (i18n "talks") }}</h2>
36.      {{ range $index, $item := $items }}
37.          {{ partial "functions/render_view" (dict "page" $ "item" . "view" (site.P
arams.projects.talk_view | default "compact") "index" $index) }}
38.      {{end}}
39.      {{ end }}
40.  </div>
41. </div>
42. </article>
43.  {{- end -}}

```

代码定义了一个具有 `article-project` 类的主文章元素，它包含一个页面标题、一个项目内容的容器和一个页面页脚。在项目内容容器内，模板使用 `.Content` 变量显示项目的内容。在项目内容下面，模板显示相关页面，包括与项目相关的帖子、出版物和事件。为了显示相关的帖子，首先将 `$page` 和 `$project` 变量分别设置为当前页面和项目的内容基础名称。然后，它使用 `where` 函数来过滤常规页面的列表，只包括那些具有与 `$project` 变量相交的 `project` 参数的 `"post"` 类型。由此产生的页面列表被存储在 `$items` 变量中，其长度被存储在 `$count` 变量中。如果 `$count` 变量大于或等于 1，模板会显示相关帖子部分的标题，然后是一个循环，在 `$items` 变量上迭代，并使用 `render_view` 部分显示每个帖子。

该模块每个项目需要添加图片和 Markdown 文件：

```

1.  ---
2.  title: Example Project
3.  summary: An example of using the in-built project page.
4.  tags:
5.    - Deep Learning
6.  date: '2016-04-27T00:00:00Z'
7.
8.  #项目的可选外部 URL（取代项目详情页）
9.  external_link: "
10. image:
11.   caption: Photo by rawpixel on Unsplash
12.   focal_point: Smart

```

其中，`'title'` 是项目的名称，`'summary'` 是论文的简介，`'tags'` 是文章标签，用于首页展示时进行筛选，`'date'` 是文章发表时间，`'image'` 是项目图片。

4.2.3 多语言支持

在本系统中，支持中文、英语和法语三种语言。

首先，对 `languages.yaml` 文件进行更改，在其中定义这三种语言。

```
1.  # 默认语言
2.  en:
3.    languageCode: en-us
4.    # 将英文内容移至`en`子文件夹。
5.    contentDir: content/en
6.
7.  #用第二/三种语言配置网站。
8.  zh:
9.    languageCode: zh-cn
10.   contentDir: content/zh
11.
12.  fr:
13.    languagecode: fr
14.    contentDir: content/fr
```

在上面的代码中，`contentDir` 参数指定了网站的内容文件夹为 `content`。

接着，手动添加语言支持，即在网站的每个页面中添加对应语言的翻译内容。通过在网站根目录下创建不同语言的文件夹来实现网页内容的多语言。具体来说，在网站根目录下创建一个名为 `content` 的文件夹，然后在该文件夹下为每种语言创建一中文、英语和法语的子文件夹，并在文件夹内添加相应的文档。这种方法比较繁琐，但可以灵活控制每个页面的翻译内容和样式：

```
content/
├── en/
├── fr/
└── zh/
```

最后，仅需在每个子文件夹中创建相应语言的网页内容就能够实现构建每种语言所对应的网页。在网站中，可以通过直接右上角切换语言，也可以使用 `url` 参数进行访问：

- 中文页面：`https://metislab.github.io/zh/`
- 英语页面：`https://metislab.github.io/en/`
- 法语页面：`https://metislab.github.io/fr/`

4.2.4 网站维护

对于实验室工作人员，他们可以通过一定的培训和指导，快速高效且保质保量地进行自行维护和更新实验室网站。**Hugo** 是一个简单易用的静态网站生成器，使用 **Markdown** 语言编写内容，非常适合非专业人员进行网站维护和更新。当需要对网站内容进行更新时，需要进行以下步骤：

1. 确定更新内容：首先需要确定要更新的内容。在实验室网站中，可以根据实验室的最新研究成果、项目进展、人员变动等方面进行更新。可以通过实验室的新闻稿、科研论文、项目报告等来获取更新内容的灵感。

2. 编辑更新内容：在确定了更新内容后，需要进行编辑。对于新闻稿、科研论文等文本内容，需要进行撰写和编辑 **Markdown** 文件；对于图片或视频等非文本内容，需要添加内容并更改相应路径。在编辑过程中，保证内容的准确性、简洁性和易读性，以便更好地向网站访问者传递信息。

3. 更新网站主题：如果需要更新网站主题，可以选择新的主题，并进行相关设置。通过更新网站主题，可以更改网站的整体风格和外观，包括颜色、字体、布局等，使网站更加美观和易于访问者使用。

4. 生成静态页面：在更新内容后，将更新后的内容添加到相应的文档中并使用 **Hugo** 重新生成静态页面。**Hugo** 的速度很出色，官方宣传每一个网页产生时间在 1ms 以内。

5. 测试更新内容：在更新内容后，需要进行测试，以确保网站的正常运行和页面显示效果。可以在本地使用“**hugo server -D**”命令进行测试，也可以发布到测试环境或临时页面中进行测试。

6. 发布更新内容：在测试通过后，只需“**git push**”将更新后的内容同步到 **GitHub** 网站上，即使用 **GitHub Actions** 自动化部署到 **GitHub Pages** 上进行发布。

通过培训和指导，实验室工作人员可以掌握 **Hugo** 的基本使用和网站维护和更新的技能，从而能够很好地进行自行维护和更新实验室网站。在进行更新内容时，需要注意更新的频率和质量。更新频率应该适当，不要过于频繁或过于稀少；更新质量应该高，以确保更新内容的准确性和有用性。同时，更新内容也需要与实验室的研究方向和目标相符合，以达到更好的宣传和推广效果。

5 总结与展望

5.1 本文总结

静态网站生成器将静态网站的简单性、速度、安全性和可扩展性与动态方法（代码重用和维护）相结合，实现了两全其美。SSG 技术易于学习，让开发人员完全掌控技术堆栈，而不会增加额外负担和复杂性。如果没有静态网站生成器，纯静态网站的维护将是一场噩梦。例如，如果想要在网站的导航中添加一个项目，必须手动在每个页面上进行相同的更改。使用静态网站生成器，只需在一个地方进行导航更改，然后重新生成站点，更改将反映在整个站点中。

静态网站生成器使得即使对于最广泛的网站，静态网站的工作也是可行的。静态网站一直存在，但现在成为了一种流行趋势，一是由于现代浏览器和前端技术的迅猛发展，简化了后端功能。加之，持续交付技术及其背后的自动化脚本技术的成熟，可以将部署交给诸如 GitHub Actions 这样的平台。此外，动态网页技术的安全问题的突显，也让静态网页优势明显。静态网站可以把整站都利用 CDN 进行加速，且对于服务器的负载较小，托管成本也较低。

传统的网站建设方式通常需要较长的开发周期和较高的维护成本，并且在更新和发布网站时需要手动操作，容易出现错误。SSG 技术的出现，提供了一种全新的静态网站建设方式，不仅可以极大地缩短网站建设的周期，还可以减少维护成本和出错的风险。而本研究的创新之处在于将 SSG 技术与实际需求结合，在保证网站建设效率的同时，还能满足实验室网站的具体需求。

其中，Hugo 作为当下热门的静态网站生成器之一，被广泛用于创建信息网站例如营销网站，博客，文档网站，或者投资组合。Hugo 因其高效的网站生成速度，是大型网站的绝佳选择。

在本次项目中，我承担了从需求分析、设计模块到选择模板并成功完成实验室官网搭建的任务。本系统共计涉及 1,119 个文件、509 个文件夹，涉及人员基本信息、论文、项目、头像照片等内容。以人员为例，该系统成功添加了 50 多位研究人员及其学生的基本信息与相应头像照片，并提供了中、英、法三种语言的版本，共计两百余个 Markdown 文件。每个模块都实现了其功能与效果，这为实验室官网的搭建奠定了坚实的基础。我的工作不仅仅是技术实现，更是对实验室官网的整体规划和设计。

本课题基于 hugo 这一静态网站生成器来建设实验室官网，为实验室进行宣传推广。Hugo 作为一个免费开源软件，被用作实验室网站开发充分发挥了其无需服务器配置、简单易用、高效快捷、操作门槛低的特点，具有切实的研究意义。

本研究的创新之处在于将 SSG 技术与实际需求结合，在保证网站建设效率的同时，还能满足实验室网站的具体需求。

在实际应用中，本研究提出的网站建设方案已被成功地应用于实验室网站的搭建和更新，得到了较好的效果。该方案不仅极大地提高了实验室网站建设的效率和质量，还能为类似的网站建设提供有益的借鉴和参考。因此，本研究的成果具有一定的创新价值和应用价值，可以促进网站建设技术的进一步发展和应用。

5.2 展望

未来，随着云计算技术的快速发展，发展趋势可能为通过云端部署网站，实现更高效、更灵活的网站建设。通过静态网站生成器建设的网站被自动化部署至云平台后，还可以结合云函数、云数据库、云存储、身份服务等。比如可以在静态托管的网站上使用云函数和云数据库实现评论、留言板功能等，或者可以把网站的内容管理从原来的静态文件部署变为动态内容管理等。

此外，SSG 也可能与 Jamstack 技术的结合，进一步提高网站的性能和安全性。Jamstack 是一种现代的 Web 开发架构，它的核心思想是使用 JavaScript、API 和预渲染技术来构建快速、安全和可扩展的 Web 应用程序。Jamstack 的名字来源于其三个核心组件：JavaScript、API 和 Markup（标记语言）。

Jamstack 的优点包括快速、安全、可扩展和易于维护等方面。同时，Jamstack 也提供了丰富的工具和服务，例如 API 服务、CDN 等，可以通过与 SSG 技术结合，方便地构建和部署现代化的 Web 应用程序。

于此同时，人工智能技术和机器学习技术的应用为网站建设带来了新的机遇，未来 SSG 也可能会更加注重这方面的应用。通过这些技术的准确应用，可以生成更加个性化的网站内容，并提高用户的体验感。对于实验室网站来说，这意味着更好地满足实验室的需求，提高实验室网站的影响力和竞争力。

虽然静态网站通常不支持服务器端脚本语言，但可以通过将 AI 模型部署在云端服务器中，然后通过 API 接口与静态网站进行交互来实现 AI 技术的应用。

比如，可以使用 JavaScript 代码调用 API 接口，将用户的输入传递给 AI 模型进行处理，然后将处理结果返回给用户。

通过这种方式，可以将 AI 技术应用于静态网站中，实现更加智能化、个性化的服务和内容，提高用户的体验感。同时，这种方式也可以避免在静态网站中使用服务器端脚本语言带来的安全问题，从而让网站更加安全可靠。

需要注意的是，将 AI 技术应用于静态网站中需要考虑到网站的性能和响应速度。由于静态网站不支持服务器端脚本语言，因此需要选择高效的 AI 模型和云端服务器，以保证网站的性能和响应速度。同时，也需要考虑到 AI 模型的训练和更新，以保证其准确性和可靠性。

总之，随着互联网技术的不断更新和应用，网站建设技术也将会不断地发展和创新，为互联网业务的发展提供更加可靠、高效、智能的支持。本研究所涉及的技术和思路，对于网站建设技术的发展具有一定的借鉴和参考意义。

参考文献

- [1] Diaz C. Minimal Computing with Progressive Web Apps[J]. DHQ: Digital Humanities Quarterly, 2022, 16(2).
- [2] 乔丛枫.静态网站的开发研究[J].赤子(上中旬),2015,No.339(05):262.
- [3] 毛德祥,罗荣阁.基于 ASP.net 技术的 Web 应用程序三层设计模型[J].微型电脑应用,2002(03):26-28+56-3.
- [4] 杜 闯 .PHP 在 动 态 网 站 开 发 中 的 优 势 [J]. 电 脑 知 识 与 技 术,2010,6(13):3342-3344.
- [5] 吕振刚.基于 Ajax 技术的 WEB 研究[J].石油知识,2020,No.203(04):50-51.
- [6] 王晓强. 基于 HTML5 的 CSRF 攻击与防御技术研究[D].电子科技大学,2013.
- [7] 畅玉洁.NET 与数据库技术在动态网站开发中的探讨[J].电脑与信息技术,2021,29(04):59-61.
- [8] 刘奇旭,陈文岗,靳泽,刘清越,李香龙,刘潮歌,谭儒. 一种 Node.js 代码安全检测方法 & 系统[P]. 北京市: CN114912110A,2022-08-16.
- [9] 白万民,吴夫丹,白小军.Web 应用中的网页静态化技术研究[J].西安工业大学学报,2014,34(01):34-37+43.
- [10] 刘耀钦.基于 Smarty 模板引擎的 Web 页静态化研究与性能分析[J].计算机与数字工程,2015,43(02):295-298+333.
- [11] 冯兴利,锁志海,徐墨.基于 PHP+MySQL 的 Web 系统安全防范及全站静态化[J].现代电子技术,2012,35(08):25-27.
- [12] 雷海卫,张萍.网页制作中静态页生成技术的研究[J].电脑开发与应用,2007,153(05):2-3.
- [13] 曾春华,江南雨.动态生成静态网页技术探索[J].科技信息(学术研究),2008,272(24):511-512.
- [14] 徐白,宋玲,吴昊.JSP 静态网页生成技术的研究[J].计算机技术与发展,2010,20(06):175-178.
- [15] 黄立冬.一种半静态化网站的构建方法[J].软件,2012,33(03):38-40.

- [16] 杨昕,邢海峰,苑明星. 一种全站静态化的方法和页面静态化的方法[P]. 北京市: CN109815432A,2019-05-28.
- [17] 郑丽敏. 网页内容静态化处理方法及系统[P]. 上海市: CN115758016A,2023-03-07.
- [18] Petersen H. From Static and Dynamic Websites to Static Site Generators[J]. university of TARTU, Institute of Computer Science, 2016.
- [19] Diaz C. Using static site generators for scholarly publications and open educational resources[J]. Code4Lib Journal, 2018 (42).
- [20] Williamson E P, Wikle O M, Becker D, et al. Using static web technologies and git-based workflows to re-design and maintain a library website (quickly) with non-technical staff[J]. College & Undergraduate Libraries, 2022,28(2): 129-147.
- [21] Utomo P. Building Serverless Website on GitHub Pages[C]//IOP Conference Series: Materials Science and Engineering. IOP Publishing, 2020,879(1): 012077.
- [22] Jiang WR, Yan JH. Implementation of Static Web-Pages Generator Using JavaScript.AMM,2010,39:588-591.
- [23] 刘佳,卢显良.小型高效模板引擎的设计与实现[J].计算机应用研究,2006(04):222-224.

致 谢

在此，我要向我的母校上海大学表达最诚挚的感谢和敬意。上海大学是我人生中最重要学习和成长场所之一。在这里，我受到了一流的教育和培养，学到了丰富的知识和技能，认识了许多优秀的老师和同学。他们的教诲和帮助，让我在学术、思想和人格等方面都得到了很大的提升和改进。

我要向上海大学的全体师生和工作人员致以最诚挚的感谢。上海大学的专业老师们，在课程的教授过程中他们倾囊相授，对我专业知识的收获和能力的提高提供了莫大的帮助，使我受益匪浅。此外，辅导员老师们的辛勤劳动和无微不至的关怀引领着我在积极健康的道路上前行。宿管老师的开朗体贴为我的大学时光提供了温暖的生活环境。四年时光里，我度过了无数个日日夜夜，经历了无数个喜怒哀乐，结交了无数个知心好友。这里，是我学习和成长的摇篮，是我追求梦想和实现目标的舞台，是我收获人生和感悟人生的宝库。

这四年，我受到了许多人的关爱和帮助，包括家人、老师、同学、朋友和校友等。他们的支持和鼓励，让我在困难和挫折面前坚持不懈，让我在成功和荣誉面前保持谦虚和感恩。他们的陪伴和启迪，让我在学术和思想方面都得到了很大的提升和改进。

我将永远怀念和珍惜这段为自己的未来和社会的发展而努力奋斗的时光。祝愿大学越来越好，祝愿老师和同学们前程似锦！

谨以此文献给所有关心、支持和帮助过我的人们。

朱馨宁
上海大学

