一面 6：开发环境相关知识点与高频考题解析

工程师的开发环境决定其开发效率，常用的开发环境配置也是面试考查点之一。

知识点梳理

- IDE
- Git
- Linux 基础命令
- 前端构建工具
- 调试方法

本小节会重点介绍 Git 的基本用法、代码部署和开发中常用的 Linux 命令，然后以 webpack 为例介绍下前端构建工具，最后介绍怎么抓包解决线上问题。这些都是日常开发和面试中常用到的知识。

IDE

> 题目：你平时都使用什么 IDE 编程？有何提高效率的方法？

前端最常用的 IDE 有 Webstorm、Sublime、Atom 和 VSCode，我们可以分别去它们的官网看一下。

Webstorm 是最强大的编辑器，因为它拥有各种强大的插件和功能，但是我没有用过，因为它收费。不是我舍不得花钱，而是因为我觉得免费的 Sublime 已经够我用了。跟面试官聊到 Webstorm 的时候，没用过没事儿，但一定要知道它：第一，强大；第二，收费。

Sublime 是我日常用的编辑器，第一它免费，第二它轻量、高效，第三它插件非常多。用 Sublime 一定要安装各种插件配合使用，可以去网上搜一下“sublime”常用插件的安装以及用法，还有它的各种快捷键，并且亲自使用它。这里就不一一演示了，网上的教程也很傻瓜式。

慢，卡一下才打开。当然总体来说也是很好用的，只是个人习惯问题。

VSCode 是微软出品的轻量级（相对于 Visual Studio 来说）编辑器，微软做 IDE 那是出了名的好，出了名的大而全，因此 VSCode 也有上述 Sublime 和 Atom 的各种优点，但是我也是因为个人习惯问题（本人不愿意尝试没有新意的新东西），用过几次就不用了。

总结一下：

- 如果你要走大牛、大咖、逼格的路线，就用 Webstorm
- 如果你走普通、屌丝、低调路线，就用 Sublime
- 如果你走小清新、个性路线，就用 VSCode 或者 Atom
- 如果你面试，最好有一个用的熟悉，其他都会一点

最后注意：千万不要说你使用 Dreamweaver 或者 notepad++ 写前端代码，会被人鄙视的。如果你不做 .NET 也不要用 Visual Studio ，不做 Java 也不要用 Eclipse。

Git

你此前做过的项目一定要用过 Git，而且必须是命令行，如果没用过，你自己也得恶补一下。对 Git 的基本应用比较熟悉的同学，可以跳过这一部分了。macOS 自带 Git，Windows 需要安装 Git 客户端，去 Git 官网 下载即可。

国内比较好的 Git 服务商有 coding.net，国外有大名鼎鼎的 GitHub，但是有时会有网络问题，因此建议大家注册一个 coding.net 账号然后创建项目，来练练手。

> 题目：常用的 Git 命令有哪些？如何使用 Git 多人协作开发？

常用的 Git 命令

首先，通过 `git clone <项目远程地址>` 下载下来最新的代码，例如 `git clone git@git.coding.net:username/project-name.git` ，默认会下载 `master` 分支。

然后修改代码，修改过程中可以通过 `git status` 看到自己的修改情况，通过 `git diff <文件名>` 可查阅单个文件的差异。

最后，将修改的内容提交到远程服务器，做如下操作

```
git push origin master
```

如果别人也提交了代码，你想同步别人提交的内容，执行 `git pull origin master` 即可。

如何多人协作开发

多人协作开发，就不能使用 `master` 分支了，而是要每个开发者单独拉一个分支，使用 `git checkout -b <branchname>` ，运行 `git branch` 可以看到本地所有的分支名称。

自己的分支，如果想同步 `master` 分支的内容，可运行 `git merge master` 。切换分支可使用 `git checkout <branchname>` 。

在自己的分支上修改了内容，可以将自己的分支提交到远程服务器

```shell
git add .
git commit -m "xxx"
git push origin <branchname>
```

最后，待代码测试没问题，再将自己分支的内容合并到 `master` 分支，然后提交到远程服务器。

```shell
git checkout master
git merge <branchname>
git push origin master
```

关于 SVN

关于 SVN 笔者的态度和针对 IE 低版本浏览器的态度一样，你只需要查询资料简单了解一下。面试的时候可能会问到，但你只要熟悉了 Git 的操作，面试官不会因为你不熟悉 SVN 而难为你。前提是你要知道一点 SVN 的基本命令，自己上网一查就行。

不过 SVN 和 Git 的区别你得了解。SVN 是每一步操作都离不开服务器，创建分支、提交代码都需要连接服务器。而 Git 就不一样了，你可以在本地创建分支、提交代码，最后再一起 push 到服务器上。因此，Git 拥有 SVN 的所有功能，但是却比 SVN 强大得多。（Git 是 Linux 的创始人 Linus 发明的东西，因此也倍得推崇。）

目前互联网公司的线上服务器都使用 Linux 系统，测试环境为了保证和线上一致，肯定也是使用 Linux 系统，而且都是命令行的，没有桌面，不能用鼠标操作。因此，掌握基础的 Linux 命令是非常必要的。下面总结一些最常用的 Linux 命令，建议大家在真实的 Linux 系统下亲自试一下。

关于如何得到 Linux 系统，有两种选择：第一，在自己电脑的虚拟机中安装一个 Linux 系统，例如 Ubuntu/CentOS 等，下载这些都不用花钱；第二，花钱去阿里云等云服务商租一个最便宜的 Linux 虚拟机。推荐第二种。一般正式入职之后，公司都会给你分配开发机或者测试机，给你账号和密码，你自己可以远程登录。

> 题目：常见 linux 命令有哪些？

登录

入职之后，一般会有现有的用户名和密码给你，你拿来之后直接登录就行。运行 `ssh name@server` 然后输入密码即可登录。

目录操作

- 创建目录 `mkdir <目录名称>`
- 删除目录 `rm <目录名称>`
- 定位目录 `cd <目录名称>`
- 查看目录文件 `ls  ll`
- 修改目录名 `mv <目录名称> <新目录名称>`
- 拷贝目录 `cp <目录名称> <新目录名称>`

文件操作

- 创建文件 `touch <文件名称>  vi <文件名称>`
- 删除文件 `rm <文件名称>`
- 修改文件名 `mv <文件名称> <新文件名称>`
- 拷贝文件 `cp <文件名称> <新文件名称>`

文件内容操作

- 查看文件 `cat <文件名称>  head <文件名称>  tail <文件名称>`
- 编辑文件内容 `vi <文件名称>`

前端构建工具

构建工具是前端工程化中不可缺少的一环，非常重要，而在面试中却有其特殊性 —— **面试官会通过询问构建工具的作用、目的来询问你对构建工具的了解，只要这些你都知道，不会再追问细节**。因为，在实际工作中，真正能让你编写构建工具配置文件的机会非常少，一个项目就配置一次，后面就很少改动了。而且，如果是大众使用的框架（如 React、Vue 等），还会直接有现成的脚手架工具，一键创建开发环境，不用手动配置。

> 题目：前端为何要使用构建工具？它解决了什么问题？

何为构建工具

“构建”也可理解为“编译”，就是将开发环境的代码转换成运行环境代码的过程。**开发环境的代码是为了更好地阅读，而运行环境的代码是为了更快地执行，两者目的不一样，因此代码形式也不一样**。例如，开发环境写的 JS 代码，要通过混淆压缩之后才能放在线上运行，因为这样代码体积更小，而且对代码执行不会有任何影响。总结一下需要构建工具处理的几种情况：

- **处理模块化**：CSS 和 JS 的模块化语法，目前都无法被浏览器兼容。因此，开发环境可以使用既定的模块化语法，但是需要构建工具将模块化语法编译为浏览器可识别形式。例如，使用 webpack、Rollup 等处理 JS 模块化。
- **编译语法**：编写 CSS 时使用 Less、Sass，编写 JS 时使用 ES6、TypeScript 等。这些标准目前也都无法被浏览器兼容，因此需要构建工具编译，例如使用 Babel 编译 ES6 语法。
- **代码压缩**：将 CSS、JS 代码混淆压缩，为了让代码体积更小，加载更快。

构建工具介绍

最早普及使用的构建工具是 Grunt ，不久又被 Gulp 给追赶上。Gulp 因其简单的配置以及高效的性能而被大家所接受，也是笔者个人比较推荐的构建工具之一。如果你做一些简单的 JS 开发，可以考虑使用。

如果你的项目比较复杂，而且是多人开发，那么你就需要掌握目前构建工具届的神器 —— webpack 。不过神器也有一个缺点，就是学习成本比较高，需要拿出专门的时间来专心学

webpack 演示

接下来我们演示一下 webpack 处理模块化和混淆压缩代码这两个基本功能。

首先，你需要安装 Node.js，没有安装的可以去 Node.js 官网 下载并安装。安装完成后运行如下命令来验证是否安装成功。

```shell
node -v
npm -v
```

然后，新建一个目录，进入该目录，运行 `npm init` ，按照提示输入名称、版本、描述等信息。完成之后，该目录下出现了一个 `package.json` 文件，是一个 JSON 文件。

接下来，安装 wepback，运行 `npm i --save-dev webpack` ，网络原因需要耐心等待几分钟。

接下来，编写源代码，在该目录下创建 `src` 文件夹，并在其中创建 `app.js` 和 `dt.js` 两个文件，文件内容分别是：

```js
// dt.js 内容
module.exports = {
    getDateNow: function () {
        return Date.now()
    }
}

// app.js 内容
var dt = require('./dt.js')
alert(dt.getDateNow())
```

然后，再返回上一层目录，新建 `index.html` 文件（该文件和 `src` 属于同一层级），内容是

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>test</title>
</head>
```

```
    <script src='./dist/bundle.js'></script>
</body>
</html>
```

然后，编写 webpack 配置文件，新建 `webpack.config.js` ，内容是

```js
const path = require('path');
const webpack = require('webpack');
module.exports = {
  context: path.resolve(__dirname, './src'),
  entry: {
    app: './app.js',
  },
  output: {
    path: path.resolve(__dirname, './dist'),
    filename: 'bundle.js',
  },
  plugins: [
    new webpack.optimize.UglifyJsPlugin({
        compress: {
          //supresses warnings, usually from module minification
          warnings: false
        }
    }),
  ]
};
```

总结一下，目前项目的文件目录是：

```
src
  +-- app.js
  +-- dt.js
index.html
package.json
webpack.config.js
```

接下来，打开 `package.json` ，然后修改其中 `scripts` 的内容为：

```json
  "scripts": {
    "start": "webpack"
  }
```

总结

最后再次强调，**深刻理解构建工具存在的价值，比你多会一些配置代码更加有意义，特别是对于应对面试来说。**

调试方法

调试方法这块被考查最多的就是如何进行抓包。

> 题目：如何抓取数据？如何使用工具来配置代理？

PC 端的网页，我们可以通过 Chrome、Firefox 等浏览器自带的开发者工具来查看网页的所有网络请求，以帮助排查 bug。这种监听、查看网络请求的操作称为**抓包**。

针对移动端的抓包工具，Mac 系统下推荐使用 Charles 这个工具，首先 下载 并安装，打开。Windows 系统推荐使用 Fiddler，下载安装打开。两者使用基本一致，下面以 Charles 为例介绍。

接下来，将安装好 Charles 的电脑和要抓包的手机，连接到同一个网络（一般为公司统一提供的内网，由专业网络工程师搭建），保证 IP 段相同。然后，将手机设置网络代理（每种不同手机如何设置网络代理，网上都有傻瓜式教程），代理的 IP 为电脑的 IP，代理的端口为 `8888` 。然后，Charles 可能会有一个弹框提示是否允许连接代理，这里选择“允许”即可。这样，使用手机端访问的网页或者联网的请求，Charles 就能监听到了。

在开发过程中，经常用到抓包工具来做代理，将线上的地址代理到测试环境，Charles 和 Fiddler 都可实现这个功能。以 Charles 为例，点击菜单栏中 Tools 菜单，然后二级菜单中点击 Map Remote，会弹出配置框。首先，选中 Enable Map Remote 复选框，然后点击 Add 按钮，添加一个代理项。例如，如果要将线上的 `https://www.aaa.com/api/getuser?name=xxx` 这个地址代理到测试地址 `http://168.1.1.100:8080/api/getuser?name=xxx` ，配置如下图

小结

本小节总结了前端开发环境常考查的知识，这些知识也是前端程序员必须掌握的，否则会影响开发效率。

留言

写下你的留言

稻谷的谷 学生/前端
最后的调试方法长见识了

▲ 0　1条评论　2月前

L。同学 前端工程师 @ 北京莱熙科技