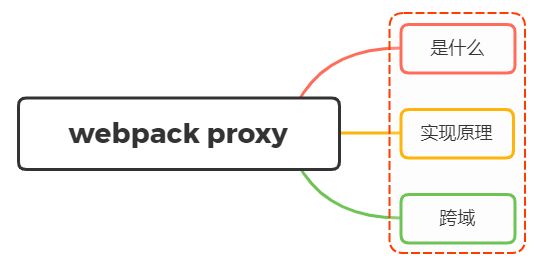
# 面试官：说说webpack proxy工作原理？为什么能解决跨域?



## 一、是什么

webpack proxy，即webpack提供的代理服务

基本行为就是接收客户端发送的请求后转发给其他服务器

其目的是为了便于开发者在开发模式下解决跨域问题（浏览器安全策略限制）

想要实现代理首先需要一个中间服务器，webpack中提供服务器的工具为webpack-dev-server

#### webpack-dev-server

webpack-dev-server是 webpack 官方推出的一款开发工具，将自动编译和自动刷新浏览器等一系列对开发友好的功能全部集成在了一起

目的是为了提高开发者日常的开发效率，**只适用在开发阶段**

关于配置方面，在webpack配置对象属性中通过devServer属性提供，如下：

// ./webpack.config.js  
const path = require('path')  
  
module.exports = {  
 // ...  
 devServer: {  
 contentBase: path.join(\_\_dirname, 'dist'),  
 compress: true,  
 port: 9000,  
 proxy: {  
 '/api': {  
 target: 'https://api.github.com'  
 }  
 }  
 // ...  
 }  
}

devServetr里面proxy则是关于代理的配置，该属性为对象的形式，对象中每一个属性就是一个代理的规则匹配

属性的名称是需要被代理的请求路径前缀，一般为了辨别都会设置前缀为/api，值为对应的代理匹配规则，对应如下：

* target：表示的是代理到的目标地址
* pathRewrite：默认情况下，我们的 /api-hy 也会被写入到URL中，如果希望删除，可以使用pathRewrite
* secure：默认情况下不接收转发到https的服务器上，如果希望支持，可以设置为false
* changeOrigin：它表示是否更新代理后请求的 headers 中host地址

## 二、工作原理

proxy工作原理实质上是利用http-proxy-middleware 这个http代理中间件，实现请求转发给其他服务器

举个例子：

在开发阶段，本地地址为http://localhost:3000，该浏览器发送一个前缀带有/api标识的请求到服务端获取数据，但响应这个请求的服务器只是将请求转发到另一台服务器中

const express = require('express');  
const proxy = require('http-proxy-middleware');  
  
const app = express();  
  
app.use('/api', proxy({target: 'http://www.example.org', changeOrigin: true}));  
app.listen(3000);  
  
// http://localhost:3000/api/foo/bar -> http://www.example.org/api/foo/bar

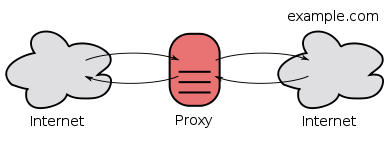
## 三、跨域

在开发阶段， webpack-dev-server 会启动一个本地开发服务器，所以我们的应用在开发阶段是独立运行在 localhost的一个端口上，而后端服务又是运行在另外一个地址上

所以在开发阶段中，由于浏览器同源策略的原因，当本地访问后端就会出现跨域请求的问题

通过设置webpack proxy实现代理请求后，相当于浏览器与服务端中添加一个代理者

当本地发送请求的时候，代理服务器响应该请求，并将请求转发到目标服务器，目标服务器响应数据后再将数据返回给代理服务器，最终再由代理服务器将数据响应给本地



在代理服务器传递数据给本地浏览器的过程中，两者同源，并不存在跨域行为，这时候浏览器就能正常接收数据

注意：**服务器与服务器之间请求数据并不会存在跨域行为，跨域行为是浏览器安全策略限制**

## 参考文献

* https://webpack.docschina.org/configuration/dev-server/#devserverproxy