

## 一、loadPackageDefinition 转换grpc对象

## 二、credentials.createInsecure

## 三、Server类

模块划分

http\_proxy 代理 & 链接

load-balancer 负载均衡相关

## 一、loadPackageDefinition 转换grpc对象

```
1 // 加载proto文件
2 const packageDefinition = protoLoader.loadSync(PROTO_PATH, {
3   keepCase: true,
4   longs: String,
5   enums: String,
6   defaults: true,
7   oneofs: true,
8 })
9 // 将proto转换成 grpc对象
10 const hello_proto = grpc.loadPackageDefinition(packageDefinition)
11 // hello_proto 返回值如下
```

对比packageDefinition的返回值（在proto-loader文件），对象的层级结构发生了变化，将方法都放到了helloworld对象下，另外Greeter类下的方法都放到了service下

```

  > helloworld: {}
  > helloworld: {Greeter: f, HelloRequest: {}, AddRequest: {}, HelloReply: {}, AddReply: {}}
    > AddReply: {format: 'Protocol Buffer 3 DescriptorProto', type: {}, fileDescriptorProtos: Array(1)}
    > AddRequest: {format: 'Protocol Buffer 3 DescriptorProto', type: {}, fileDescriptorProtos: Array(1)}
  > Greeter: class ServiceClientImpl extends client_1.Client {
    > get arguments: f ()
    > set arguments: f ()
    > get caller: f ()
    > set caller: f ()
    length: 0
    name: 'ServiceClientImpl'
    > prototype: Client {SayHello: f, Add: f, add: f, constructor: f}
  > service: {SayHello: {}, Add: {}}
    > Add: {path: '/helloworld.Greeter/Add', requestStream: false, responseStream: false, requestSerialize: f, requestDeserialize: f, ...}
  > SayHello: {path: '/helloworld.Greeter/SayHello', requestStream: false, responseStream: false, requestSerialize: f, requestDeserialize: f, ...}
    > originalName: 'SayHello'
    > path: '/helloworld.Greeter/SayHello'
    > requestDeserialize: f deserialize(argBuf) {
      return cls.toObject(cls.decode(argBuf), options);\n
    }
    > requestSerialize: f serialize(arg) {
      const message = cls.fromObject(arg);\n
      return cls.encode(message).finish();\n
    }
    > requestStream: false
    > responseType: {format: 'Protocol Buffer 3 DescriptorProto', type: {}, fileDescriptorProtos: Array(1)}
    > responseDeserialize: f deserialize(argBuf) {
      return cls.toObject(cls.decode(argBuf), options);\n
    }
    > responseSerialize: f serialize(arg) {
      const message = cls.fromObject(arg);\n
      return cls.encode(message).finish();\n
    }
    > responseStream: false
    > responseType: {format: 'Protocol Buffer 3 DescriptorProto', type: {}, fileDescriptorProtos: Array(1)}
    > __proto__: Object
  > __proto__: Object
  > [[FunctionLocation]]: @ /Users/bj-51-60-1216/Documents/myself/mycode/grpc/grpc-demo-client/node_modules/@grpc/grpc-js/build/src/make-client.js:50
  > [[Scopes]]: Scopes[3]
  > __proto__: class Client {
    constructor(address, credentials, options = {}) {
      var _a, _b;
      options = Object.assign({}, options);
      this[INTERCEPTOR_SYMBOL] = (_a = options.interceptors) != null && _a != void 0 ? _a : [];
      delete options.interceptors;
      this[INTERCEPTOR_PROVIDER_SYMBOL] = (_b = options.interceptor_providers) != null && _b != void 0 ? _b : [];
      delete options.interceptor_providers;
      if (this[INTERCEPTOR_SYMBOL].length > 0 && this[INTERCEPTOR_PROVIDER_SYMBOL].length > 0) {
        throw new Error('Both interceptors and interceptor_providers were passed as options ' +
          'to the client constructor. Only one of these is allowed.');
```

## 二、credentials.createInsecure

返回新的 无证书的 ChannelCredentials

```

const packageDefinition = protoLoader.loadSync(PROTO_PATH, { packageDefinition = {helloworld.Greeter: {_, helloworld.HelloWorld: {
keepCase: true,
longs: true,
enums: true,
defaultValues: true,
oneofs: true,
}}});
// 转换 grpc
const helloProto = protoLoader.loadSync(PROTO_PATH, { packageDefinition: {helloworld.Greeter: {_, helloworld.HelloWorld: {
// 全局挂载
this.app.grpc.helloProto = helloProto;
this.app.grpc.helloProto = helloProto;
const insecure = grpc.credentials.createInsecure() insecure = InsecureChannelCredentialsImpl {callCredentials: EmptyCallCredentials}
this.app.grpcGreeter = new this.app.grpc.Greeter({ this = AppBootHook, grpc = {credentials: {_, Metadata: <accessor>, logVerbosity: <accessor>},
'localhost:50051',
insecure
}
console.log('insecure', insecure)

```

### 三、Server类

#### 1、addService

```

1 const server = new grpc.Server()
2 server.addService(hello_proto.Greeter.service, {
3   sayHello,
4   Add,
5 })

```

bindAsync