axio

axios源码分析 part2

这一章来研究一下上一章说先放一放文件, 分开一段段理解。

```
'use strict';

var utils = require('./utils');
var bind = require('./helpers/bind');
var Axios = require('./core/Axios');
var mergeConfig = require('./core/mergeConfig');
var defaults = require('./defaults');


```

1. 首先加载需要的文件
2. utils - 工具函数比如 判断是否对象类型等等辅助用工具函数
3. mergeConfig - 融合多个配置文件的函数
4. defaults - 创建实例的默认配置文件（上一章节已经提及）

```
/**
 * Create an instance of Axios
 *
 * @param {Object} defaultConfig The default config for the instance
 * @return {Axios} A new instance of Axios
 */
function createInstance(defaultConfig) {
  var context = new Axios(defaultConfig);
  var instance = bind(Axios.prototype.request, context);

  // Copy axios.prototype to instance
  utils.extend(instance, Axios.prototype, context);

  // Copy context to instance
  utils.extend(instance, context);

  // Factory for creating new instances
  instance.create = function create(instanceConfig) {
    return createInstance(mergeConfig(defaultConfig, instanceConfig));
  };

  return instance;
}

```

```
// Create the default instance to be exported
var axios = createInstance(defaults);
```

1. createInstance ， 接受一个参数，参数就是配置信息。
2. 创建Axios 实例，把defaultConfig 传入作为用户配置信息
3. 使用bind函数让request函数调用的上下文为Axios。这样设计request 就可以使用上下文提供的功能比如interceptor。返回的也是一个函数。
4. 使用工具函数extend, 让axios原型函数全部添加在instance，也在添加的时候，对函数的运行上下文做了处理，使其上下文总是在context。
5. 创建好的函数添加create属性。create属性的值是一个函数，接受一个参数，参数就是配置信息。其内部实现就是返回createIsntance函数。设计的目的就是可以让开发者也可以创建自己想要的实例根据开发者的配置信息。
6. 然后调用函数，传入defaults。defaults 就是默认的位置文件。

```
// Expose Cancel & CancelToken
axios.CanceledError = require('./cancel/CanceledError');
axios.CancelToken = require('./cancel/CancelToken');
axios.isCancel = require('./cancel/isCancel');
axios.VERSION = require('./env/data').version;
axios.toFormData = require('./helpers/toFormData');
```

1. 都是取消操作相关，这里就不做展开，会在后面的章节专门讲解

```
// Expose all/spread
axios.all = function all(promises) {
  return Promise.all(promises);
};
```

1. 添加all属性，就是一个promise all 的包装。

```
axios.spread = require('./helpers/spread');

module.exports = function spread(callback) {
  return function wrap(arr) {
    return callback.apply(null, arr);
  };
};
```

1. 其内部就是一个柯理函数，spread是一个函数接受一个参数，参数是一个函数，spread运行返回一个函数，这么做就可以扩展作为参数传入的函数，让它可以有机会提供更多参数。也可以进行懒加载运行。

```
// Expose isAxiosError
axios.isAxiosError = require('./helpers/isAxiosError');



module.exports = function isAxiosError(payload) {
  return utils.isObject(payload) && (payload.isAxiosError === true);
};

```

1. 检测payload是否有属性isAxiosError。 如果有，错误由axios抛出。现在理解有点抽象，等后面使用了会深入讲解。