关于 service

service 一旦被定义后就可以在任何地方通过依赖的方式调用且可以保存数据，直到应用结束，比如

```
controller(['service', function(service){}]);
```

而 controller 则一旦路由发生变化 controller 就会失效，下次打开页面就需要重新加载保存数据

factory

factory 是 angularjs 中创建 service 最简单的一个方法

创建 factory

通过使用 factory 我们只需要返回一个包含数据，方法的对象就可以了，如下

```
angular.module('MyApplication')
.factory('user', ['$http', function($http){
    var loginUrl = 'http://localhost/api/v1/login';
    return {
        login: function(username, password) {
            return $http.post(loginUrl, {
                username: username,
                password: password
            });
        },
    };
}]);
```

调用 factory

通过以下方式即可调用上述 service

```
angular.module('MyApplication')
.controller(['$scope', 'user', function($scope, user){
    user.login('name', 'password').then(function(response){
        if(response.data.err_code == 0) {
            console.log('登录成功')
        } else {
            console.log('登录失败')
        }
    }, function(response){
        console.log('网络请求出错了', response)
    });
}]);
```

应用场景

在 service 中，如果我们仅仅需要的是一些方法或数据集合，则可以通过 factory 简单的创建一个 service 来满足需求

注：如果需要在 config 中配置 service 则不能使用 factory 来创建

service

创建 service

service 通过构造函数来创建 service，具体实现同 factory，示例如下

```
angular.module('MyApplication')
.service('user', ['$http', function($http){
    var loginUrl = 'http://localhost/api/v1/login';
    this.login: function(username, password) {
        return $http.post(loginUrl, {
            username: username,
            password: password
        });
    };
}]);
```

调用方式

其调用方式与 factory 同样

```
angular.module('MyApplication')
.controller(['$scope', 'user', function($scope, user){
    user.login('name', 'password').then(function(response){
        if(response.data.err_code == 0) {
            console.log('登录成功')
        } else {
            console.log('登录失败')
        }
    }, function(response){
        console.log('网络请求出错了', response)
    });
}]);
```

应用场景

可以看到，这里的 login 使用了 this. 的方式来创建的，在此场景中可以写更多的业务逻辑来控制数据

注：如需在 config 中配置 service 的话，除了 factory 不能用之外， service 也不能用

provider

provider 是最底层的创建 service 的方法，可以在 config 中被调用和配置

创建 provider

创建 provider，与 factory、service 不同的是，provider 需要使用 this,$get 来返回方法和数据

```
angular.module('MyApplication')
.provider('user', ['$http', function($http){
    this.loginUrl = 'http://localhost/api/v1/login';
    this.setLoginUrl = function(url){
        this.loginUrl = url;
    }
    this.$get = function($http) {
        return {
            login: function(username, password) {
                return $http.post(this.loginUrl, {
                    username: username,
                    password: password
                });
            },
        };
    }
}]);
```

调用方式

与 factory、service 不同的是， provider 可以在 config 中调用，示例如

```
angular.module('MyApplication')
.config(['user', function(user){
    user.setLoginUrl('http://localhost/api/v2/login');
}]);
```

普通调用则与 factory, service 的调用方式一样

应用场景

当需要通过配置指定特殊的数据源时就可以使用 provider 来实现，比如开发环境和生产环境的数据交互地址的域名可能不一致，就可以通过这种方式来实现