

LazyRest4 完全指南

LazyRest4 完全指南

- [LazyRest4 完全指南](#)
 - [简介](#)
 - [安装](#)
 - [获取源文件](#)
 - [将apache指定到对应目录](#)
 - [初始化项目](#)
 - [使用composer安装PHP依赖](#)
 - [使用Bower安装JS依赖](#)
 - [修改URL重写规则](#)
 - [设置默认管理帐号](#)
 - [实战指南](#)
 - [需求](#)
 - [设计MySQL表](#)
 - [创建新项目](#)
 - [创建接口](#)
 - [生成和测试代码](#)
 - [测试接口](#)
 - [授权机制](#)
 - [其他预置接口](#)
 - [修改接口](#)
 - [列表接口](#)
 - [其他](#)

简介

LazyRest, 是一个基于Web的类Rest风格API生成器。LazyRest4（以下简称LR4）是为LazyPHP4专门定制的全新设计的版本，但只要简单的修改代码模板，就可以为其他的框架生成代码。

和之前的版本相比，LR4具有以下特性：

- 直接生成最终代码，更好调试、更高性能。
- 全新的交互界面，更漂亮和便捷。
- 按接口而非数据表设计界面，具有更大的灵活性。
- 本身不需要数据库，项目配置直接保存成文件，方便分享。

安装

获取源文件

下载安装包，并解压。或者直接Git Clone。

Repo地址： <https://github.com/easychen/LazyRest4>

将apache指定到对应目录

初始化项目

使用composer安装PHP依赖

```
composer install --no-dev
```

使用Bower安装JS依赖

```
bower install
```

修改URL重写规则

```
cp sample.htaccess .htaccess
```

设置默认管理帐号

打开 /config/app.php

修改以下内容

```
$GLOBALS['lpconfig']['admin_email'] = 'your email';  
$GLOBALS['lpconfig']['admin_password'] = 'your password';
```

如果配置正常，这时候访问Web根目录，LazyRest就已经可以使用了。

实战指南

为了更好的演示，我们选择一个真实需求来看看LR4到底怎么用。

需求

我们需要一个公司部门内使用的函数共享库。这个共享库位于公司内网，公司员工可以注册帐号，登入帐号以后可以添加、下载、修改和删除函数。添加函数时可以将其设置为私有，这样别人就无法浏览和下载该函数了。

设计MySQL表

我们使用PHPMyAdmin来设计数据表。新建一个库，coderemote。总共两张表，一张user表用来存放开发者帐号、领一张code表，用来存放函数。

user表结构如下：

	#	Name	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/>	1	<u>id</u>	int(11)			No	None	AUTO_INCREMENT
<input type="checkbox"/>	2	<u>email</u>	varchar(128)	utf8_general_ci		No	None	
<input type="checkbox"/>	3	<u>name</u>	varchar(32)	utf8_general_ci		No	None	
<input type="checkbox"/>	4	<u>password</u>	varchar(256)	utf8_general_ci		No	None	
<input type="checkbox"/>	5	<u>level</u>	tinyint(1)			No	1	

设计数据表时需要注意以下两点：

- 每个表需要有id字段，这个字段在查询接口中用于since_id，这个后文详细讲。
- 每个字段需要添加注释，注释的内容是字段的中文名。

code表结构如下：

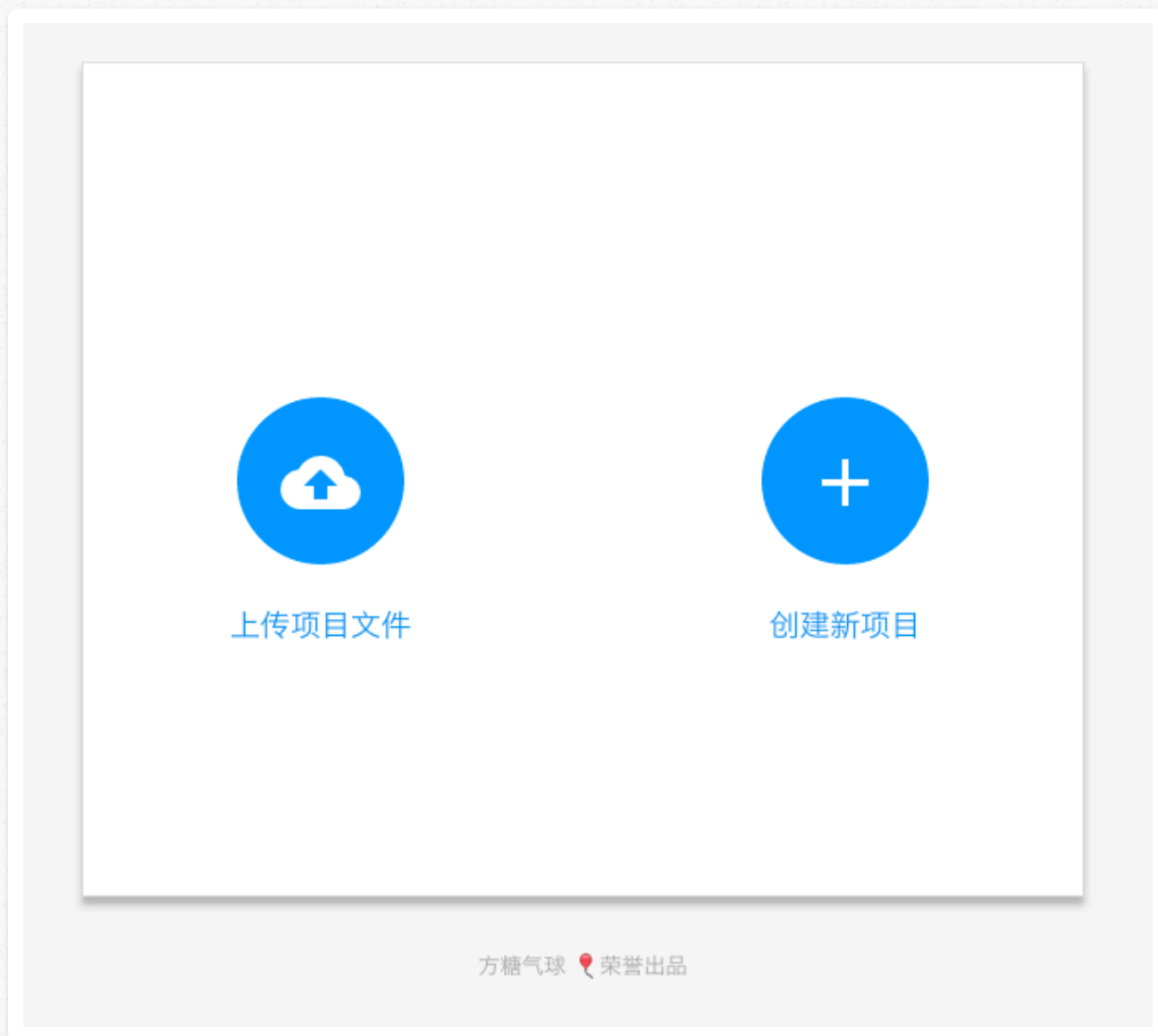
	#	Name	Type	Collation	Attributes	Null	Default	Extra
<input type="checkbox"/>	1	<u>id</u>	int(11)			No	None	AUTO_INCREMENT
<input type="checkbox"/>	2	<u>code</u>	text	utf8_general_ci		No	None	
<input type="checkbox"/>	3	<u>name</u>	varchar(64)	utf8_general_ci		No	None	
<input type="checkbox"/>	4	<u>uid</u>	int(11)			No	None	
<input type="checkbox"/>	5	<u>uname</u>	varchar(32)	utf8_general_ci		No	None	
<input type="checkbox"/>	6	<u>create_at</u>	datetime			No	None	
<input type="checkbox"/>	7	<u>private</u>	tinyint(1)			No	0	

为了方便显示，我们对账户名做了冗余（uname）字段，同时通过private字段来标识是否私有。

索引一般在整个API完成后，根据SQL来添加，这里就暂时跳过了。

创建新项目

LR4采用项目文件保存数据，所以使用方式类似软件。已有项目可以直接上传项目文件，然后继续，这里我们直接「创建新项目」。



在设计接口之前，我们需要把数据库配置下。这样在添加字段时，可以便捷的使用数据库的信息。点击顶导航右侧的⚙️图标，设置好Mysql信息。注意这个数据库可以是远程的，并不要一定和LR在一台机器上。



项目设置



数据库设置

Host 数据库地址，如：localhost、127.0.0.1

Port 数据库端口，如：3306

User 数据库用户名

Password 数据库密码

Database 数据库库名

保存设置

点击保存后，LR会连接数据库，并把数据表信息缓存起来。所以如果后续有数据库变动，请再到设置页面，点一遍保存来刷新数据表信息。

创建接口

然后我们来创建接口。首先是帐号注册接口。

接口设置

帐号注册

访问路径

方法 ☐ GET ☒ POST

开放 ☒ 是 ☐ 否

Auth - 授权控制

任意访问 ☐ 是 ☒ 否

Target - 目标数据表

一般我会把写操作设置为POST，把读操作设置为GET。这里我并不严格遵守RESTful的规范，因为觉得这样更简单一些。

目标数据表是我们生成SQL时需要操作的数据表，如果要使用联表操作，则目标数据表可以不选。

Input fields - 输入字段

+ 添加新字段

Input filter - 输入过滤代码

```
1 <?php
2 // 在自动化输入检查之后执行
3 // 可以直接使用input里边的变量
4 ?>
```

然后是输入字段和输入过滤代码。简单的说，「输入字段」部分，会帮你自动生成输入过滤代码，包括输入检查，类型转换等。而如果你觉得生成的代码不好用，或者没有把事情做完时，可以在「输入过滤代码」部分进行补充。

然后我们来添加字段，点击「添加新字段」，然后会出现字段设置界面。因为是LazyRest，所以当然不会让你挨个手工去填写，我们在右边做了快速输入。在「来自数据库」tab下，选择「user」表，然后选上Email字段，点「填至左侧」，LR会根据数据表信息自动填上一部分信息。

字段设置



英文名	email
所属table	user
中文名	电子邮件
检查函数	check_not_empty
过滤函数	可空，过滤函数，直接返回修正后的数据
可否为空	<input type="radio"/> 是 <input checked="" type="radio"/> 否

保存设置

删除

来自数据库

最近输入

user

id

email

name

password

level

填至左侧

因为email字段设置为了not null，所以检查函数默认启用了check_not_empty。但是则会这个函数不够强，我们需要检查是否为email，所以我们自己写一个，先把名字写上（注意LP4中的检查函数必须以check_开头，过滤函数则没有限制）：

字段设置



英文名	email
所属table	user
中文名	电子邮件
检查函数	check_mail_lr
过滤函数	可空，过滤函数，直接返回修正后的数据
可否为空	<input type="radio"/> 是 <input checked="" type="radio"/> 否

保存设置


删除

来自数据库

最近输入

请选择数据表

填至左侧

保存好。LR也直接提供了公共函数的在线编写界面，点击顶导航第二个图标，会弹出代码编辑图层。


```
1 <?php
2 // 公共函数写这里，例如检查函数、过滤函数、权限检查函数
3 function check_mail_lr( $email )
4 {
5     return filter_var( $email , FILTER_VALIDATE_EMAIL );
6 }
```

[保存代码](#)

保存后，点右上角关掉。接着再把其他帐号注册接口需要的字段添进来。我们不希望密码明文存储，所以我们在输入过滤代码中对它进行sha1编码（生产环境需要更强的密码策略）。

```
<?php
// 在自动化输入检查之后执行
// 可以直接使用input里边的变量
$password = sha1( $password );
?>
```

这样输入部分就完成。接下来看业务逻辑代码。LR预置了添加、修改、列表、删除四大内业务逻辑模板，这些模板你都可以在 `/codetpl/code` 目录下进行修改。

这里我们选添加，添加模板预置了唯一字段检查逻辑。在我们的需求里，email字段是唯一的。于是把email字段加上。

这样核心逻辑就完成了。为了方便使用，在预置逻辑里，添加、修改和删除都会返回对应的那条数据的信息（用于提示，某某数据已经删除之类）。

所以这里还用到输出字段和输出过滤代码。这输入很像，但记住password字段其实是不需要返回的。

选完以后我们就可以保存接口了。

生成和测试代码

点击顶导航右三图标，代码会生成到controller目录下的LazyRestController.php。为了调试和部署方便，LR目前所有代码都放到这个文件里。

这时候就可以开始测试了。

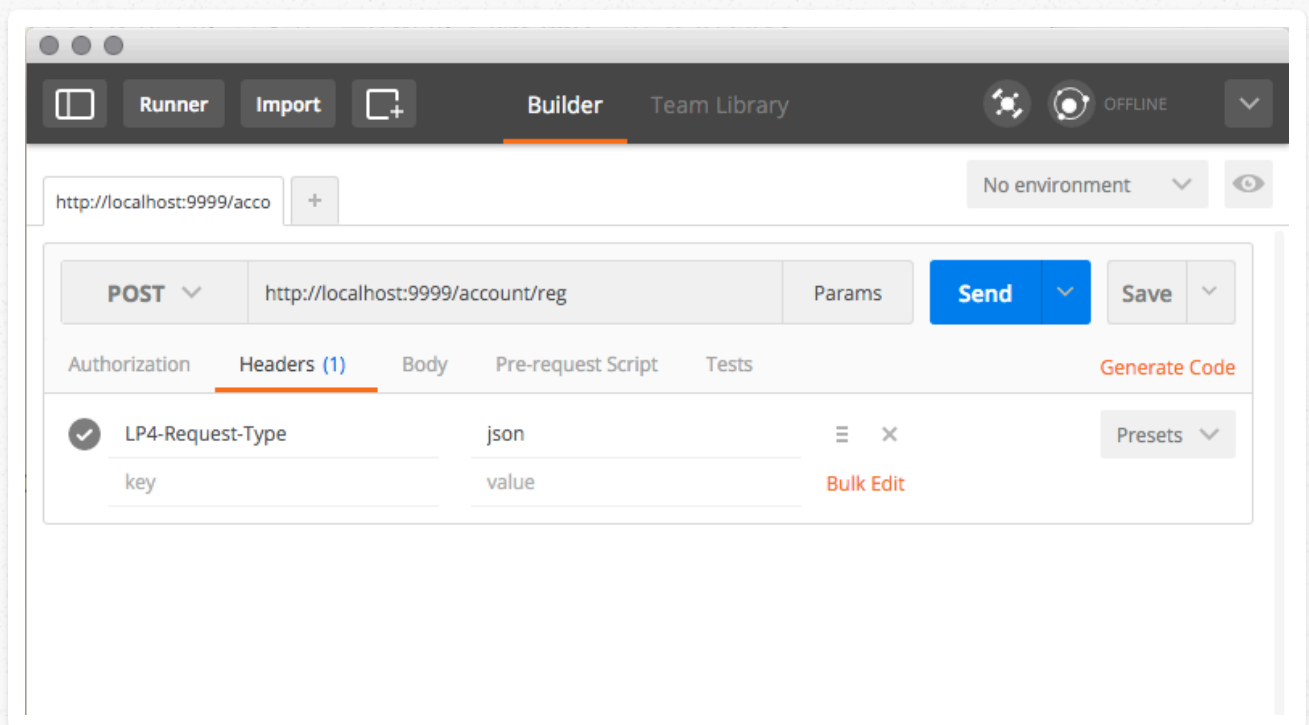
如果生成代码中有语法错，会导致LR整体报错，只需要删除生成的文件就好。

如果路由没有更新，请记得把LazyPHP4的自动重建路由开关打开。在/config/app.php中

```
$GLOBALS['lpconfig']['buildeverytime'] = true;
```

测试接口

我一般使用[PostMan](#)来测试API，所以以下以它为例。



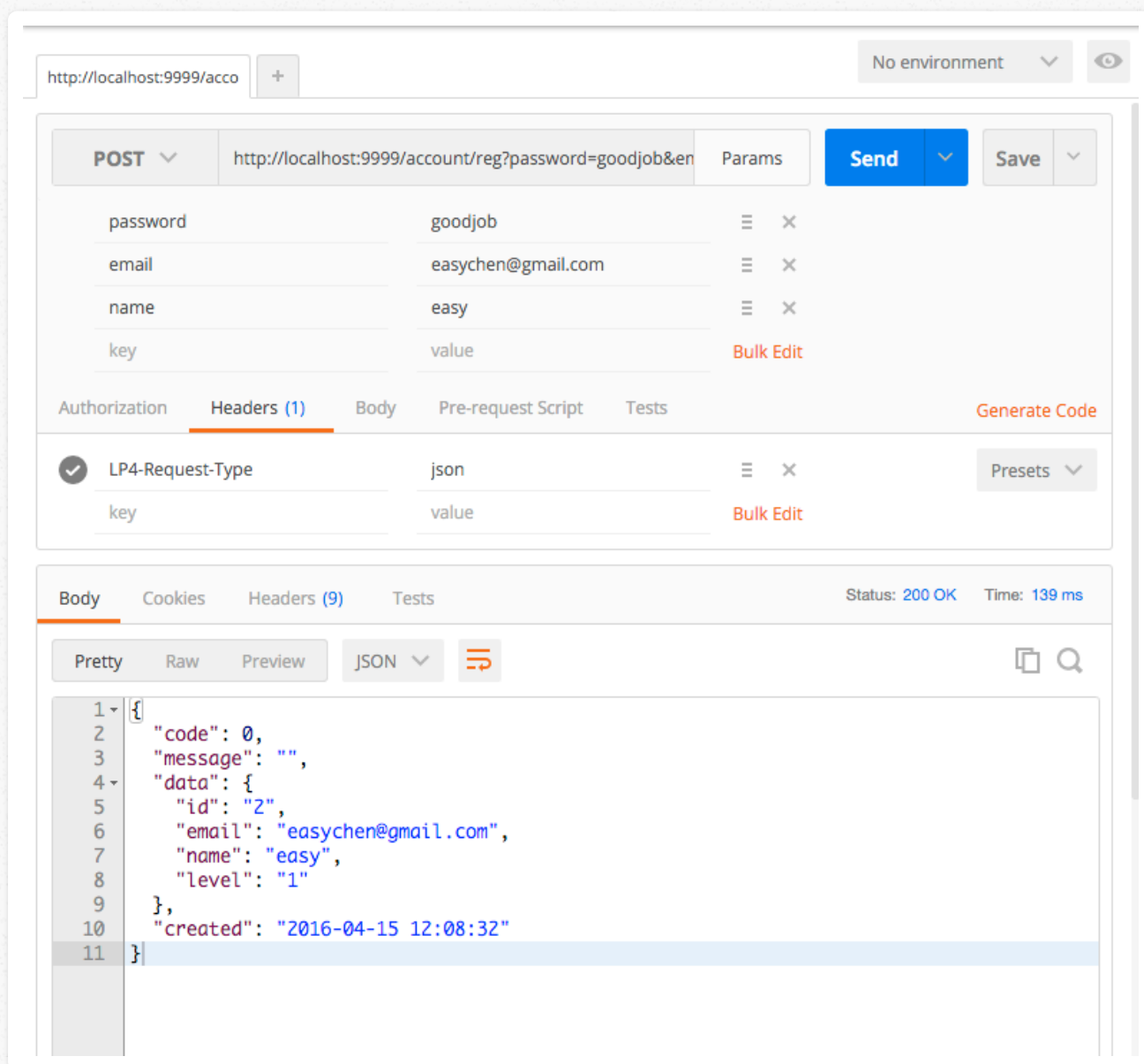
注意几个点：

- 我们设置了这个接口只支持POST，所以用GET会提示404

- LazyPHP4.5开始因为支持了Web界面，所以要在添加一个特殊的Header才能强制返回json。

```
'LP4-Request-Type': 'json'
```

填好参数，接口已经正常工作了。



授权机制

在这个项目里，我们采用token机制：通过email和密码换取一个token，需要授权的接口，带上这个token来访问。

首先我们来做这个接口，在LR中再新建一个。顺便说句，点击这个图标就可以返回接

口列表。



将email和password添加为输入字段，设置为必填。

Input fields - 输入字段

password

email

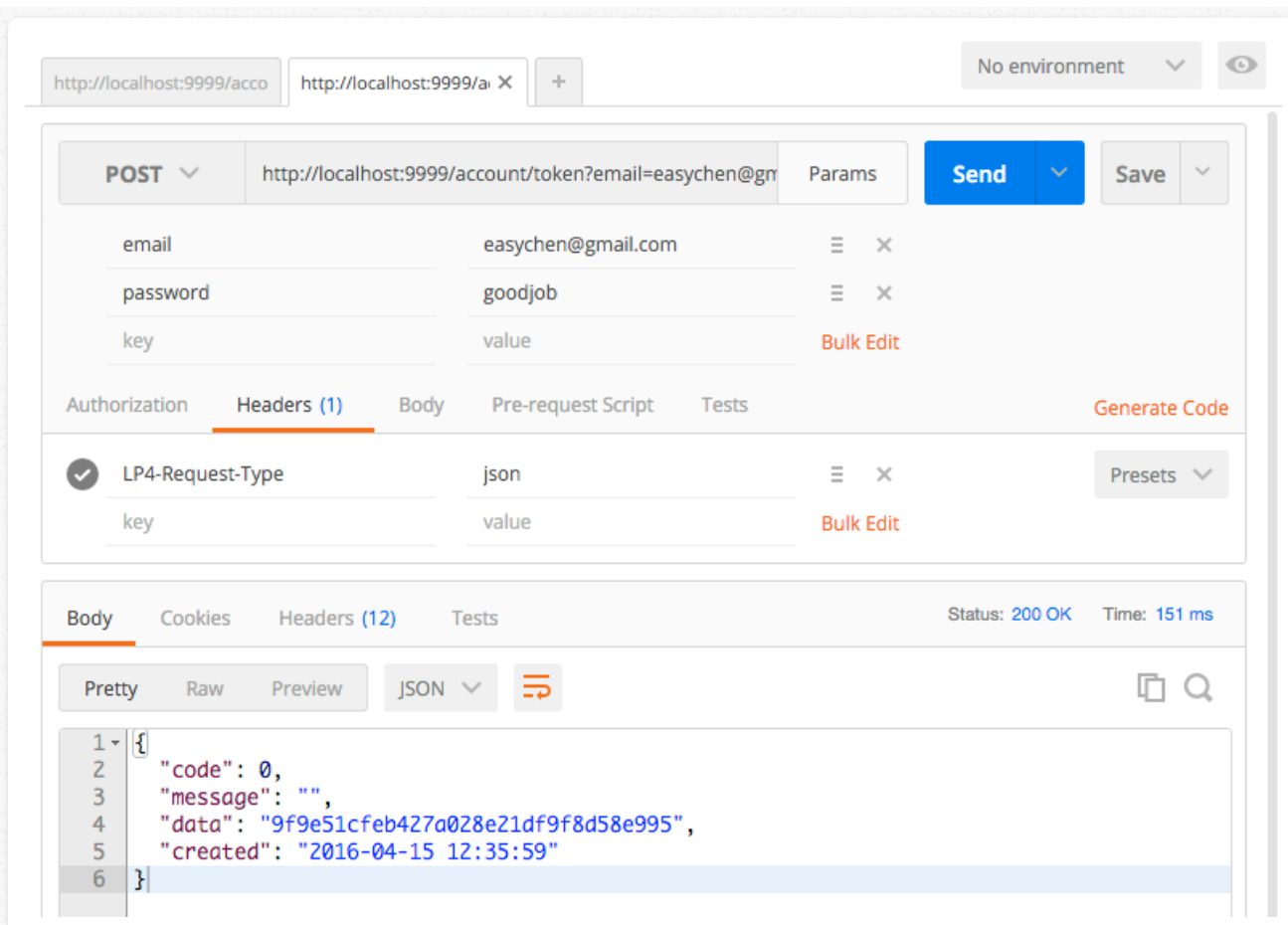
+ 添加新字段

然后我们这次直接自定义业务逻辑代码

```
<?php
// 首先取得用户信息
if( $user = get_line( "SELECT * FROM `user` WHERE `email` = '" . s( $email ) . "'" ) )
{
    if( $user['password'] == sha1( $password ) )
    {
        // 创建一个Session
        session_start();
        $_SESSION['uid'] = $user['id'];
        $_SESSION['uname'] = $user['name'];
        $_SESSION['level'] = $user['level'];

        return render_json( session_id() );
    }
    else
        return send_error( 'AUTH' , '错误的密码' );
}
else
    return send_error( 'INPUT' , '电子邮件不存在' );
```

Build一下代码，然后我们来测试。



这里我们直接把session_id当成token返回了。这样可以很好的利用上PHP本身的session机制。

然后我们来做一个需要权限的接口：添加代码，顺便把权限限制加上。

仍然是先设置输入字段。

Input fields - 输入字段

create_at

private

uid

code

name

+ 添加新字段

这里需要注意的是，create_at的值不是来自REQUEST，而是来自php本身的。

字段设置



英文名	<input type="text" value="create_at"/>
所属table	<input type="text" value="-phpfunction-"/>
中文名	<input type="text" value="创建时间"/>
检查函数	<input type="text" value="可空，输入检查函数，返回false则提示错误"/>
过滤函数	<input type="text" value="now"/>
可否为空	<input type="radio"/> 是 <input checked="" type="radio"/> 否

来自数据库

最近输入

code

id

code

name

uid

uname

create_at

private

填至左侧

保存设置

删除

将所属table写成「-phpfunction-」，并在过滤函数上填上函数名，LR就会把过滤函数的值返回给这个字段。

顺手在公共函数里把now实现了：

```
function now()
{
    return date("Y-m-d H:i:s");
}
```

uid和uname来自于也做类似处理，不过它们的值来自session。

```
function uid()
{
    return $_SESSION['uid'];
}

function uname()
{
    return $_SESSION['uname'];
}
```

最后我们来做权限控制。如果你查看一下生成好的代码，会发现在构造函数中有这么一

段:

```
public function __construct()
{
    $noauth[] = 'user_lcadd_1460691333694';
    $noauth[] = 'user_lccustom_1460694654897';

    if( function_exists('lazyrest_auth_check') )
    {
        if( !in_array( g('a') , $noauth ) )
        {
            $this->auth();
            lazyrest_auth_check();
        }
    }
}
```

可以看到，所有授权控制的「任意访问」设置为否的接口，都会通过 lazyrest_auth_check函数进行权限检查。但是lazyrest_auth_check函数默认是没定义的，所以默认不生效。我们只需要在公共函数中把这个函数实现了，就完成了权限控制。

先看看\$this->auth()的实现，在/controller/AuthedContrllor.php里边：

```
protected function auth()
{
    $token = t(v('_token'));
    if( strlen( $token ) > 0 )
    {
        session_id( $token );
    }

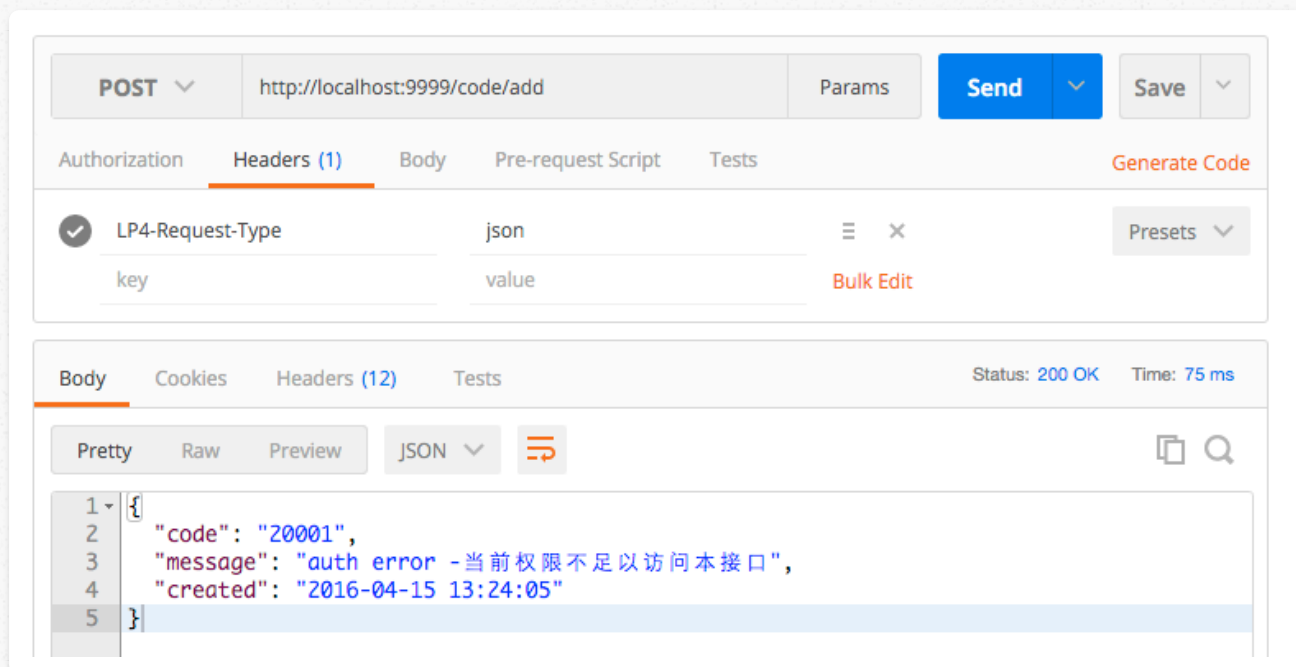
    session_start();
}
```

它接受_token参数，并把其作为session id，自动enable了session。所以我们只需要在 lazyrest_auth_check里边简单的检查下\$_SESSION数据是否正常就OK了。

```
function lazyrest_auth_check()
{
    if( intval( $_SESSION['level'] ) < 1 )
    {
        send_error( 'AUTH' , '当前权限不足以访问本接口' );
        exit;
    }
}
```

```
}  
}
```

同时，如果对auth函数的实现不满意，可以直接覆盖它的实现。
保存后Build下代码，就可以测试了：



带上token，已经可以访问了。

POST

http://localhost:9999/code/add?_token=9f9e51cf9b427a00

Params

Send

Save

_token

9f9e51cf9b427a0028e21df9f8d58e

key

value

Bulk Edit

Authorization

Headers (1)

Body

Pre-request Script

Tests

Generate Code

LP4-Request-Type

json

key

value

Bulk Edit

Presets

Body

Cookies

Headers (12)

Tests

Status: 200 OK

Time: 103 ms

Pretty

Raw

Preview

JSON

1

{

2

"code": "10001",

3

"message": "input error- 代码(code)未提供或格式不正确 via check_not_empty return ",

4

"created": "2016-04-15 13:27:51"

5

}

其他预置接口

修改接口

id和code作为输入字段；业务逻辑代码选「修改」，WHERE子句添加id，选完全匹配。

字段设置

英文名

id

所属table

code

匹配规则

☒ 全等匹配(=)

☐ 部分匹配(%)

保存设置

删除

来自数据库 最近输入

code

id

code

name

uid

uname

create_at

private

填至左侧

然后因为只有自己能修改自己的代码，所以在业务逻辑代码中加上权限检查：

```
if( $last['uid'] != uid() ) return send_error( 'AUTH' , '只能修改自己发布的函数' );
```

注意在UPDATE和DELETE预置模块中，会查询对应的记录，并保存到\$last中供使用。

列表接口

列表接口很简单，预置的业务逻辑也是WHERE匹配，和前文一样处理就好。

The screenshot shows a REST client interface. The top bar indicates a GET request to `http://localhost:9999/code/list`. Below the bar, the 'Headers' tab is active, showing a single header: `LP4-Request-Type: json`. The 'Body' tab is also visible. The response section shows a status of `200 OK` and a time of `78 ms`. The response body is displayed in JSON format, showing a list of two code entries.

```
{
  "code": 0,
  "message": "",
  "data": {
    "0": {
      "id": "2",
      "code": "我是私有代码",
      "name": "常用函数2",
      "uid": "2",
      "uname": "easy",
      "create_at": "2016-04-15 13:44:46",
      "private": "1"
    },
    "1": {
      "id": "1",
      "code": "我就是代码",
      "name": "常用函数",
      "uid": "2",
      "uname": "easy",
      "create_at": "2016-04-15 13:30:52",
      "private": "0"
    }
  }
}
```

下边是列表接口支持的参数：

- `_order:asc/dsec`。
- `_order_by`:字段名。
- `_since_id`:从某个id开始返回。

- `_count`:每次返回的数据条数，最多1000条。

但上文我们发现了一个问题，私有函数也被列出来。所以我们需要在业务逻辑处，进行调整，给where子句再加一个条件：

```
$add_where = " (`private` != 1 OR `uid` = '" . intval( $uid ) . "' ) ";  
  
if( strlen( t( $where_sql ) ) < 1 ) $where_sql = 'WHERE ' . $add_where;  
else $where_sql .= ' AND ( ' . $add_where . ' )';
```

要了解如何调整sql的拼接，直接查看生成好的代码是最好的办法。

另外，为了让查询的实现更简单点，LR预置的删除是直接删除，而非标记删除。

其他

本来只是做来给自己用的，但因为很多同学表示也想用，就放出来了。写文档解释如何使用什么的挺费事，所以如果LazyRest帮你节省了时间和精力，可以考虑打赏我一点零花钱 😊

微信



支付宝

