ZoeeyPHP 用户手册

moxie(system128@gmail.com)

出版日期 2010-10-10

目录

项目介绍 iv
1. 快速入门 1
2. class ZeActiveString 3
    2.1. 类摘要 3
    2.2. put 5
    2.3. putAll 6
    2.4. active 7
    2.5. cancel 7
    2.6. update 8
    2.7. clear 9
    2.8. toString 10
3. class ZeLoader 12
    3.1. 类摘要 12
    3.2. isGet 15
    3.3. isPost 15
    3.4. isPut 16
    3.5. isDelete 16
    3.6. isTrace 16
    3.7. getGet 17
    3.8. fromGet 17
    3.9. getPost 19
    3.10. fromPost 20
    3.11. getCookie 21
    3.12. fromCookie 22
    3.13. getRequest 23
    3.14. fromRequest 24
    3.15. getServer 25
    3.16. fromServer 26
    3.17. getFile 27
    3.18. fromFile 28
    3.19. getSession 29
    3.20. fromSession 30
    3.21. getIp 31
    3.22. fromIp 32
    3.23. setValues 33
4. class ZePageSet 35
    4.1. 类摘要 35
    4.2. __construct 38
    4.3. setRecordCount 39
    4.4. getCurrent 40
    4.5. current 40
    4.6. getPageCount 40
    4.7. getPageSize 41
    4.8. getRecordCount 41
    4.9. getStartRow 41
    4.10. getOffset 42
    4.11. getEndRow 42
    4.12. hasPrev 42
    4.13. hasNext 43
    4.14. isLast 43
    4.15. isFirst 43
    4.16. isList 44

4.17. first 44
4.18. prev 44
4.19. next 45
4.20. last 45
4.21. all 45
4.22. siblings 46
4.23. compare 46
4.24. isCurrent 47
5. class ZeRecorder 48
5.1. 类摘要 48
5.2. __construct 50
5.3. query 51
5.4. getStmt 52
5.5. bind 52
5.6. fetch 53
5.7. getInt 54
5.8. fetchAll 55
5.9. execute 56
5.10. exec 56
5.11. lastId 57
5.12. affected 58
5.13. begin 58
5.14. rollback 59
5.15. commit 60
5.16. close 60
6. class ZeRouter 61
6.1. 类摘要 61
6.2. 参数规则 63
6.3. add 64
6.4. addRegexp 65
6.5. addArray 66
6.6. addParamCount 67
6.7. addAllRegexp 68
6.8. append 68
6.9. shiftSep 69
6.10. end 70
6.11. parse 71
7. class ZeStatus 72
7.1. 类摘要 72
7.2. __construct 73
7.3. getLabel 74
7.4. getBrief 74
7.5. getName 75

项目介绍

ZoeeyPHP 是PHP扩展实现的Web开发框架底层，特点是较大限度的将控制权释放给了程序员。

建议PHP版本 PHP 5 >= 5.2.0 。

要点介绍

1. 为经验丰富的开发人员设计。
2. 提供最基础的开发功能如字符串控制、客户端数据提取、数据分页、数据库访问、路由、状态跟踪。
3. 可用于保护源码。能二次开发编译后放入您的产品中。
4. 提供一个良好的结构实践范例，扩展框架本身并不锁定结构。
5. 它不是MVC框架，但您可以通过它快速设计出自己的MVC框架。
6. 不包含设置项，方便部署。
7. 为代码洁癖患者提供的静默式错误跟踪设计。

站点

http://code.google.com/p/zoeeyphp

下载

http://code.google.com/p/zoeeyphp/downloads/list

版本库

```
hg clone https://code.google.com/p/zoeeyphp
```

Hg/Mercurial 下载： http://mercurial.selenic.com

安装

```
[    /]# cd /php-src/ext/zoeey
[zoeey]# /path/of/php/bin/phpize
[zoeey]# ./configure --with-php-config=/path/of/php/bin/php-config
[zoeey]# make
[zoeey]# make install
[zoeey]# cp modules/zoeey.so /path/of/php/lib/php_zoeey.so (extension_dir=lib/)
[zoeey]# make clean
```

配置好php.ini重新启动http服务即可。

第 1 章 快速入门

安装好ZoeeyPHP后便可将示例项目放入http目录内测试。

第一次浏览示例项目会自动进行数据初始和目录权限检查等。

示例工程说明

浏览提示

1. 您需要一个强大的PHP IDE，例如：NetBeans http://netbeans.org/。
2. 增加源码目录： zoeey/api 、当前工程根目录（bootstrap.php中定义的DIR_ROOT目录）。

   示例项目优先考虑了IDE支持，对路径提示和代码提示支持较好。

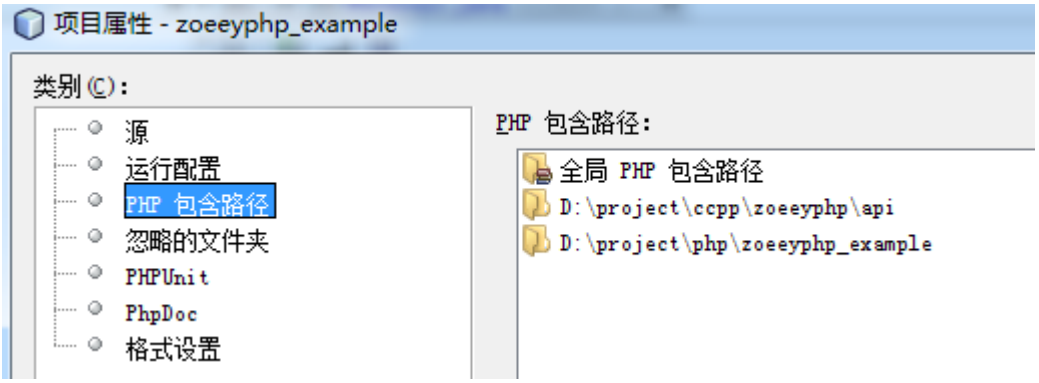

图 1.1. NetBeans 源码目录示意

目录结构

```
/zoeeyphp_example
.
|-- bootstrap.php            (启动文件，初始基础信息)
|-- config                   (配置文件目录)
|   |-- Config.php           (配置文件：数据库链接、模板缓存、调试状态、时区等)
|   `-- init.php             (初始化基础库、设定时区、SESSION存储方法)
|-- database
|   `-- zoeeyphp-example.db  (SQLite数据库，安装后自动生成)
|-- index.php                (新闻模块入口，本程序为多入口。含路由分配、模板分配等)
|-- install                  (系统安装)
|   `-- index.php
|-- lib                      (独立库)
|   |-- firephp              (FirePHP http://www.firephp.org/)
|   |-- main                 (自属库，与module/common的区别是该目录下的类可移植)
|   |   |-- Supervisor.php   (全局状态跟踪)
|   |   `-- funces.php       (基础函数：salted md5，307跳转等。)
|   `-- smarty
|-- module
|   |-- common               (专属库)
|   |   |-- Conn.php         (数据库连接、缓存服务连接等)
|   |   |-- Vali.php         (数据验证)
|   |   |-- PowerCenter.php  (权限分配)
|   |   |-- Tpl.php          (模板引擎封装)
|   |   `-- plugin           (通用模板函数)
|   |       |-- function.buildurl.php      (链接生成，简单示例)
|   |       |-- function.dateformat.php    (日期时间格式化，支持中文)
|   |       `-- function.dump.php          (调试工具，打印数据)
|   `-- news                     (新闻模块示例，数据模型由数组维护)
|       |-- News.php             (动作分配，业务逻辑)
|       |-- NewsPower.php        (权限装载)
|       |-- NewsRecorder.php     (数据持久)
|       |-- NewsValidator.php    (数据验证)
|       |-- news_category.php    (示例分类数据)
|       `-- plugin           (模板函数均在本目录下，可参见common)
|           |-- function.newsList.php   (新闻列表调用函数，非本模块模板也可调用)
|           `-- function.newsView.php   (新闻详情调用函数)
|-- script
|-- style
|-- temp          (临时文件目录)
|   |-- cache     (模板缓存)
|   |-- compile   (模板编译)
|   `-- lock      (安装锁，初始自动安装，可多次安装)
`-- tpl                      (模板目录)
    |-- common.header.html   (公共header,footer也可建立类似文件)
    |-- common.noti.html     (状态提示页，error或success级别的错误会跳转到此页)
    |-- news.edit.html       (新闻新增、编辑表单页)
    |-- news.list.html       (新闻列表页)
    `-- news.view.html       (新闻详情查看页，含一列表调用示意)
```

例 1.1. 目录结构

第 2 章 class ZeActiveString

2.1. 类摘要

以条件激活拼装的字符串。

注意

注意：字符串拼装顺序以条件“载入顺序”为准。

```
 ZeActiveString
{
  ZeActiveString put(
            string

            $condition
          ,

            string

            $val
          );

  ZeActiveString putAll(
            string

            $map
          );

  ZeActiveString active(
            string

            $condition
          );

  ZeActiveString cancel(
            string

            $condition
          );

  ZeActiveString update(
            string

            $oldCondition
          ,

            string

            $newCondition
          );

  ZeActiveString clear();

  ZeActiveString toString(
            string

            $sep
          );
}
```

例 2.1. ZeActiveString

ZeActiveString::put
    增加条件与结果项。

ZeActiveString::putAll
    增加多个条件与结果项。

ZeActiveString::active
    激活条件。

ZeActiveString::cancel
    取消条件。

ZeActiveString::update
    调换条件。

ZeActiveString::clear
    取消所有激活条件。

ZeActiveString::toString
    将结果拼接为字符串，并使用 $seq 作为间隔。

2.2. put

增加条件与结果项。

描述

```
ZeActiveString put(
        string

        $condition
      ,

        string

        $val
      );
```

在条件和结果项较为分散，或动态增加单项时使用。

参数

condition
    条件名。

val
    结果。

返回值

当前 ZeActiveString 对象。

范例

```
$activeStr = new ZeActiveString();
$activeStr->put('select', 'select id,title,content from article 1=1');
$activeStr->put('where.id', 'AND id = :id');
$activeStr->put('where.category', 'AND category = :category');
$activeStr->active('select');
$activeStr->active('where.category');
echo $activeStr->toString(' ');
/* select id,title,content from article where 1=1 AND category = :category */
```

例 2.2. 操作一个多重条件的SQL语句

2.3. putAll

增加多个条件与结果项。

描述

```
ZeActiveString putAll(
        array

        $map
      );
```

在条件和结果项集中的情况下使用。

参数

map
    数组键位条件，值为结果项。

返回值

当前 ZeActiveString 对象。

范例

```
$activeStr = new ZeActiveString();
$activeStr->putAll(array(
      'select' => 'select id,title,content from 1=1'
    , 'where.id' => 'AND id = :id'
    , 'where.category' => 'AND category = :category'
    , 'where.limit' => 'limit 1'
));
$activeStr->active('select');
$activeStr->active('where.id'); /* 根据具体情况进行激活，可换用 category 进行尝试 */
echo $activeStr->toString(' ');
/* select id,title,content from article where 1=1 AND id = :id */
```

例 2.3. 集中操作一个多重条件的SQL语句

2.4. active

激活条件。

描述

```
  ZeActiveString active(
          string

          $condition
        );
```

激活条件，取消前激活多次仅生效一次。

参数

condition
    需要激活的条件。

返回值

当前 ZeActiveString 对象。

范例

参见cancel范例。

2.5. cancel

取消条件。

描述

```
  ZeActiveString cancel(
```

```
          string

          $condition
        );
```

取消条件，取消已激活的条件。若取消的条件未激活则不进行任何操作。

参数

condition
    需要取消的条件。

返回值

当前 ZeActiveString 对象。

范例

```
$activeStr = new ZeActiveString();
$activeStr->putAll(array(
      'select' => 'select id,title,content from article where 1=1'
    , 'where.id' => 'AND id = :id'
    , 'where.category' => 'AND category = :category'
    , 'where.limit' => 'limit 1'
));
$activeStr->active('select');
$activeStr->active('where.id');
$activeStr->cancel('where.id');
echo $activeStr->toString(' ');
/* select id,title,content from article where 1=1   */
```

例 2.4. 在条件激活后取消

2.6. update

调换条件。

描述

```
  ZeActiveString update(
          string

          $oldCondition
        ,

          string

          $newCondition
        );
```

调换条件，若原条件不存在则直接添加激活条件。

参数

oldCondition
    需要取消的条件。

newCondition
    需要激活的条件。

返回值

当前 ZeActiveString 对象。

范例

```
$activeStr = new ZeActiveString();
$activeStr->putAll(array(
      'select' => 'select id,title,content from article where 1=1'
    , 'where.id' => 'AND id = :id'
    , 'where.category' => 'AND category = :category'
    , 'where.limit' => 'limit 1'
));
$activeStr->active('select');
$activeStr->active('where.id');
$activeStr->update('where.id', 'where.category');
echo $activeStr->toString(' ');
/* select id,title,content from article where 1=1 AND category = :category */
```

例 2.5. 更替一个条件

2.7. clear

取消所有激活条件。

描述

```
  ZeActiveString clear();
```

将已激活条件进行清空。

返回值

当前 ZeActiveString 对象。

范例

```
$activeStr = new ZeActiveString();
$activeStr->putAll(array(
      'select' => 'select id,title,content from article where 1=1'
    , 'where.id' => 'AND id = :id'
    , 'where.category' => 'AND category = :category'
    , 'where.limit' => 'limit 1'
));
$activeStr->active('select');
$activeStr->active('where.id');
$activeStr->active('where.category');
$activeStr->clear();
echo $activeStr->toString(' '); /* 输出长度为0的字符串 */
```

例 2.6. 清空所有条件

2.8. toString

将结果拼接为字符串，并使用 $seq 作为间隔。

描述

```
  ZeActiveString toString(
          string

          $sep
        );
```

依照条件填充顺序依次查看条件是否激活，并将以激活的结果值使用 $seq 作为间隔拼接起来。

参数

sep
    间隔符。

返回值

当前 ZeActiveString 对象。

范例

```
$activeStr = new ZeActiveString();
$activeStr->putAll(array(
      'select' => 'select id,title,content from article where 1=1'
    , 'where.id' => 'AND id = :id'
    , 'where.category' => 'AND category = :category'
    , 'where.limit' => 'limit 1'
));
$activeStr->active('select');
$activeStr->active('where.id');
$activeStr->active('where.category');
echo $activeStr->toString('++');
/* select id,title,content from article where 1=1++AND category = :category++AND id = :id*/
```

例 2.7. 更替一个条件

第 3 章 class ZeLoader

3.1. 类摘要

请求数据提取器。

```
 ZeLoader
{
```

```
bool isGet();
```

```
bool isPost();
```

```
bool isPut();
```

```
bool isDelete();
```

```
bool isTrace();
```

```
mixed getGet(
          string

          $key
        ,

          mixed

          $default

          = NULL
        );
```

```
void fromGet(
          object | array

          &$fields
        ,

          string | array

          $names

          = NULL
        ,

          mixed

          $default

          = NULL
        );
```

```
mixed getPost(
          string

          $key
        ,

          mixed

          $default

          = NULL
```

例 3.1. ZeLoader

ZeLoader::isGet
    检查请求方式是否为 GET。

ZeLoader::isPost
    检查请求方式是否为 POST。

ZeLoader::isPut
    检查请求方式是否为 PUT。

ZeLoader::isDelete
    检查请求方式是否为 DELETE。

ZeLoader::isTrace
    检查请求方式是否为 TRACE。

ZeLoader::getGet
    获取 GET 方式传递的数据。

ZeLoader::fromGet
    获取 GET 方式传递的数据，并填充到指定变量。

ZeLoader::getPost
    获取 POST 方式传递的数据。

ZeLoader::fromPost
    获取 POST 方式传递的数据，并填充到指定变量。

ZeLoader::getCookie
    获取 COOKIE 方式传递的数据。

ZeLoader::fromCookie
    获取 COOKIE 方式传递的数据，并填充到指定变量。

ZeLoader::getRequest
    获取 GET，POST 和 COOKIE 方式传递的数据。

ZeLoader::fromRequest
    获取 GET，POST 和 COOKIE 方式传递的数据，并填充到指定变量。

ZeLoader::getServer
    获取 $_SERVER 的数据。

ZeLoader::fromServer
    获取 $_SERVER 的数据，并填充到指定变量。

ZeLoader::getFile
    获取 FILE 传递的数据。

ZeLoader::fromFile
    获取 FILE 传递的数据，并填充到指定变量。

ZeLoader::getSession
    获取 SESSION 方式传递的数据。

ZeLoader::fromSession
    获取 SESSION 方式传递的数据，并填充到指定变量。

ZeLoader::getIp
    获取客户端IP。

ZeLoader::fromIp
    获取客户端IP，并填充到指定变量。

ZeLoader::setValues
    载入预设值。

3.2. isGet

检查请求方式是否为 GET。

描述

```
  bool isGet();
```

获取当前页面所使用的请求方法，并判断是否为 GET 方式。

返回值

仅当页面请求方法为 GET 时返回 true，否则为 false。

范例

```
$loader = new ZeLoader();

if ($loader->isGet()){
 /* 从数据库中取回现有数据，并递向view层 */
}
```

例 3.2. 编辑页面载入时，请求现有数据

3.3. isPost

检查请求方式是否为 POST。

描述

```
  bool isPost();
```

获取当前页面所使用的请求方法，并判断是否为 POST 方式。

返回值

仅当页面请求方法为 POST 时返回 true，否则为 false。

范例

```
$loader = new ZeLoader();

if ($loader->isPost()){
 /* 这里可以放数据验证，持久化等代码。 */
}
```

例 3.3. 判断表单是否提交

3.4. isPut

检查请求方式是否为 PUT。

描述

```
bool isPut();
```

获取当前页面所使用的请求方法，并判断是否为 PUT 方式。

返回值

仅当页面请求方法为 PUT 时返回 true，否则为 false。

范例

参见isPost范例。

3.5. isDelete

检查请求方式是否为 DELETE。

描述

```
bool isDelete();
```

获取当前页面所使用的请求方法，并判断是否为 DELETE 方式。

返回值

仅当页面请求方法为 DELETE 时返回 true，否则为 false。

范例

参见isPost范例。

3.6. isTrace

检查请求方式是否为 TRACE。

描述

```
bool isTrace();
```

获取当前页面所使用的请求方法，并判断是否为 TRACE 方式。

返回值

仅当页面请求方法为 TRACE 时返回 true，否则为 false。

范例

参见isPost范例。

3.7. getGet

获取 GET 方式传递的数据。

描述

```
mixed getGet(
        string

        $key
      ,

        mixed

        $default

        = NULL
      );
```

获取 GET 方式传递的数据，如字段不存在则返回默认值。

参数

key
    字段名。

default
    默认值。

返回值

请求字符串中存在所需字段则返回相应值，否则返回默认值。

范例

```
/* script.php?name=example_get */
$loader = new ZeLoader();

echo $loader->getGet('name'); /* example_get */
echo $loader->getGet('other','example_default'); /* example_default */
```

例 3.4. 字段值和默认值

3.8. fromGet

获取 GET 方式传递的数据，并填充到指定变量

描述

```
void fromGet(
        object | array

        &$fields
      ,

        string | array

        $names

        = NULL
      ,

        mixed

        $default

        = NULL
      );
```

批量将指定的字段，填充入数组或对象。

参数

fields
    字段数组或对象。

names
    字段名，数组或逗号分隔字段名的字符串。

注意

字段名验证规则为"[a-zA-Z0-9_]+"，除此之外的字符均被认为是间隔符。

default
    默认值。

返回值

无返回值，字段值将直接填充入 fields 。

范例

```
/* script.php?name=example_get&age=128_get */
$values = null;  /* fields 参数为 null 时，结果为数组。*/
$loader = new ZeLoader();
$loader->fromGet($values, 'name, age');
var_export($values); /* array('name' => 'example_get', 'age' => '128_get') */

$values = new stdClass();
$loader->fromGet($values, 'name, age');
/**
  * $values = new stdClass();
  * $values->name = 'example_get';
  * $values->age = '128_get';
  */
```

例 3.5. 对象或数组填充

3.9. getPost

获取 POST 方式传递的数据。

描述

```
mixed getPost(
        string

        $key
      ,

        mixed

        $default

        = NULL
      );
```

获取 POST 方式传递的数据，如字段不存在则返回默认值。

参数

key
    字段名。

default
    默认值。

返回值

请求字符串中存在所需字段则返回相应值，否则返回默认值。

范例

```
/* script.php?name=example_post */
$loader = new ZeLoader();

echo $loader->getPost('name'); /* example_post */
echo $loader->getPost('other','example_default'); /* example_default */
```

例 3.6. 字段值和默认值

3.10. fromPost

获取 POST 方式传递的数据，并填充到指定变量

描述

```
void fromPost(
        object | array

        &$fields
      ,

        string | array

        $names

        = NULL
      ,

        mixed

        $default

        = NULL
      );
```

批量将指定的字段，填充入数组或对象。

参数

fields
    字段数组或对象。

names
    字段名，数组或逗号分隔字段名的字符串。

注意

字段名验证规则为"[a-zA-Z0-9_]+"，除此之外的字符均被认为是间隔符。

default
    默认值。

返回值

无返回值，字段值将直接填充入 fields 。

范例

```
/* name=example_post&age=128_post */
$values = null;  /* fields 参数为 null 时，结果为数组。*/
$loader = new ZeLoader();
$loader->fromPost($values, 'name, age');
var_export($values); /* array('name' => 'example_post', 'age' => '128_post') */

$values = new stdClass();
$loader->fromPost($values, 'name, age');
/**
  * $values = new stdClass();
  * $values->name = 'example_post';
  * $values->age = '128_post';
  */
```

例 3.7. 对象或数组填充

3.11. getCookie

获取 COOKIE 方式传递的数据。

描述

```
  mixed getCookie(
          string

          $key
        ,

          mixed

          $default

          = NULL
        );
```

获取 COOKIE 方式传递的数据，如字段不存在则返回默认值。

参数

key
    字段名。

default
    默认值。

返回值

请求字符串中存在所需字段则返回相应值，否则返回默认值。

范例

```
/* name=example_cookie */
$loader = new ZeLoader();

echo $loader->getCookie('name'); /* example_cookie */
echo $loader->getCookie('other','example_default'); /* example_default */
```

例 3.8. 字段值和默认值

3.12. fromCookie

获取 COOKIE 方式传递的数据，并填充到指定变量

描述

```
void fromCookie(
        object | array

        &$fields
      ,
        string | array

        $names

        = NULL
      ,
        mixed

        $default

        = NULL
      );
```

批量将指定的字段，填充入数组或对象。

参数

fields
    字段数组或对象。

names
    字段名，数组或逗号分隔字段名的字符串。

注意

字段名验证规则为"[a-zA-Z0-9_]+"，除此之外的字符均被认为是间隔符。

default
    默认值。

返回值

无返回值，字段值将直接填充入 fields 。

范例

```
/* name=example_cookie;age=128_cookie */
$values = null;  /* fields 参数为 null 时，结果为数组。*/
$loader = new ZeLoader();
$loader->fromCookie($values, 'name, age');
var_export($values); /* array('name' => 'example_cookie', 'age' => '128_cookie') */

$values = new stdClass();
$loader->fromCookie($values, 'name, age');
/**
  * $values = new stdClass();
  * $values->name = 'example_cookie';
  * $values->age = '128_cookie';
  */
```

例 3.9. 对象或数组填充

3.13. getRequest

获取 GET，POST 方式传递的数据。

描述

```
mixed getRequest(
        string

        $key
      ,

        mixed

        $default

        = NULL
      );
```

获取 GET，POST 方式传递的数据，如字段不存在则返回默认值。

参数

key
    字段名。

default
    默认值。

返回值

请求字符串中存在所需字段则返回相应值，否则返回默认值。

范例

```
/* name=example_post */
$loader = new ZeLoader();

echo $loader->getRequest('name'); /* example_post */
echo $loader->getRequest('other','example_default'); /* example_default */
```

例 3.10. 字段值和默认值

3.14. fromRequest

获取 GET，POST 方式传递的数据，并填充到指定变量

描述

```
void fromRequest(
        object | array

        &$fields
      ,
        string | array

        $names

        = NULL
      ,
        mixed

        $default

        = NULL
      );
```

批量将指定的字段，填充入数组或对象。

参数

fields
    字段数组或对象。

names
    字段名，数组或逗号分隔字段名的字符串。

注意

字段名验证规则为"[a-zA-Z0-9_]+"，除此之外的字符均被认为是间隔符。

default
    默认值。

返回值

无返回值，字段值将直接填充入 fields 。

范例

```
/* name=example_post&age=128_post */
$values = null;  /* fields 参数为 null 时，结果为数组。*/
$loader = new ZeLoader();
$loader->fromRequest($values, 'name, age');
var_export($values); /* array('name' => 'example_post', 'age' => '128_post') */

$values = new stdClass();
$loader->fromRequest($values, 'name, age');
/**
  * $values = new stdClass();
  * $values->name = 'example_post';
  * $values->age = '128_post';
  */
```

例 3.11. 对象或数组填充

3.15. getServer

获取 $_SERVER 方式传递的数据。

描述

```
mixed getServer(
        string

        $key
      ,

        mixed

        $default

        = NULL
      );
```

获取 $_SERVER 的数据。

参数

key
    字段名。

default
    默认值。

返回值

请求字符串中存在所需字段则返回相应值，否则返回默认值。

范例

```
$loader = new ZeLoader();

echo $loader->getServer('SCRIPT_FILENAME'); /* file name */
echo $loader->getServer('other','example_default'); /* example_default */
```

例 3.12. 字段值和默认值

3.16. fromServer

获取 $_SERVER 的数据，并填充到指定变量。

描述

```
void fromServer(
        object | array

        &$fields
      ,

        string | array

        $names

        = NULL
      ,

        mixed

        $default

        = NULL
      );
```

批量将指定的字段，填充入数组或对象。

参数

fields
    字段数组或对象。

names
    字段名，数组或逗号分隔字段名的字符串。

注意

字段名验证规则为"[a-zA-Z0-9_]+"，除此之外的字符均被认为是间隔符。

default
    默认值。

返回值

无返回值，字段值将直接填充入 fields 。

范例

```
$values = null;  /* fields 参数为 null 时，结果为数组。*/
$loader = new ZeLoader();
$loader->fromServer($values, 'REQUEST_METHOD, SCRIPT_FILENAME');
var_export($values); /* array('REQUEST_METHOD' => 'POST', 'SCRIPT_FILENAME' => __FILE__) */

$values = new stdClass();
$loader->fromServer($values, 'REQUEST_METHOD, SCRIPT_FILENAME');
/**
  * $values = new stdClass();
  * $values->REQUEST_METHOD = 'POST';
  * $values->SCRIPT_FILENAME = __FILE__;
  */
```

例 3.13. 对象或数组填充

3.17. getFile

获取 FILE 传递的数据。

描述

```
  mixed getFile(
          string

          $key
        );
```

获取文件表单上传的数据。

参数

key
    字段名。

返回值

请求字符串中存在所需字段则返回相应值。

范例

```
$loader = new ZeLoader();

var_export($loader->getFile('file_a'));
```

例 3.14. 字段值和默认值

3.18. fromFile

获取 FILE 的数据，并填充到指定变量。

描述

```
void fromFile(
        object | array

        &$fields
      ,

        string | array

        $names

        = NULL
      );
```

批量获取文件表单上传的数据，填充入数组或对象。

参数

fields
    字段数组或对象。

names
    字段名，数组或逗号分隔字段名的字符串。

注意

字段名验证规则为"[a-zA-Z0-9_]+"，除此之外的字符均被认为是间隔符。

返回值

无返回值，字段值将直接填充入 fields 。

范例

```
$values = null;  /* fields 参数为 null 时，结果为数组。*/
$loader = new ZeLoader();
$loader->fromFile($values, 'file_a, file_b');

$values = new stdClass();
$loader->fromFile($values, 'file_a, file_b');
```

例 3.15. 对象或数组填充

3.19. getSession

获取 SESSION 方式传递的数据。

描述

```
mixed getSession(
        string

        $key
      ,

        mixed

        $default

        = NULL
      );
```

获取 SESSION 方式传递的数据，如字段不存在则返回默认值。

参数

key
    字段名。

default
    默认值。

返回值

请求字符串中存在所需字段则返回相应值，否则返回默认值。

范例

```
/* script.php?name=example_session */
$loader = new ZeLoader();

echo $loader->getSession('name'); /* example_session */
echo $loader->getSession('other','example_default'); /* example_default */
```

例 3.16. 字段值和默认值

3.20. fromSession

获取 SESSION 方式传递的数据，并填充到指定变量

描述

```
  void fromSession(
          object | array

          &$fields
        ,

          string | array

          $names

          = NULL
        ,

          mixed

          $default

          = NULL
        );
```

批量将指定的字段，填充入数组或对象。

参数

fields
    字段数组或对象。

names
    字段名，数组或逗号分隔字段名的字符串。

注意

字段名验证规则为"[a-zA-Z0-9_]+"，除此之外的字符均被认为是间隔符。

default
    默认值。

返回值

无返回值，字段值将直接填充入 fields 。

范例

```
/* name=example_session&age=128_session */
$values = null;  /* fields 参数为 null 时，结果为数组。*/
$loader = new ZeLoader();
$loader->fromSession($values, 'name, age');
var_export($values); /* array('name' => 'example_session', 'age' => '128_session') */

$values = new stdClass();
$loader->fromSession($values, 'name, age');
/**
  * $values = new stdClass();
  * $values->name = 'example_session';
  * $values->age = '128_session';
  */
```

例 3.17. 对象或数组填充

3.21. getIp

获取客户端IP。

描述

```
string | long getIp(
        bool

        $isFromat

        = FALSE
      );
```

获取当前请求者IP地址。

注意

当需要防止 HTTP_X_FORWARDED_FOR 欺诈时，请使用 getServer('REMOTE_ADDR')

参数

isFromat
    true 返回 点分十进制格式, false 返回有符长整型。

返回值

默认返回有符长整型，isFormat 为 true 则返回点分十进制格式。

范例

```
$loader = new ZeLoader();
var_export($loader->getIp());     // 2130706433
var_export($loader->getIp(true)); // 127.0.0.1
```

例 3.18. 字段值和默认值

3.22. fromIp

获取 FILE 的数据，并填充到指定变量。

描述

```
void fromIp(
        string | array

        &$fields
      ,

        string | array

        $names

        = NULL
      ,

        bool

        $isFromat

        = FALSE
      );
```

批量获取请求者IP地址，填充入数组或对象。

注意

当需要方式 HTTP_X_FORWARDED_FOR 欺诈时，请使用 fromServer 获取 REMOTE_ADDR。

参数

fields
    字段数组或对象。

names
    字段名，数组或逗号分隔字段名的字符串。

注意

字段名验证规则为"[a-zA-Z0-9_]+"，除此之外的字符均被认为是间隔符。

返回值

无返回值，字段值将直接填充入 fields 。

范例

```
$values = null;  /* fields 参数为 null 时，结果为数组。*/
$loader = new ZeLoader();
$loader->fromIp($values, 'ip_a, ip_b');

$values = new stdClass();
$loader->fromIp($values, 'ip_a, ip_b');
```

例 3.19. 对象或数组填充

3.23. setValues

载入预设值。

描述

```
void setValues(
        array

        $values
      );
```

载入预设值，如ZeRouter的参数结果，使用 GET 或 REQUEST 获取。优先级低于常规GET传参。

参数

values
    预设值。

返回值

无返回值 。

范例

```
/* @var $router Router */
$router = new ZeRouter();
$router->add('/:action');
$router->addArray('action', array('add'), '{/}');
$router->addArray('action', array('list', 'search'), '/:page')->append('{/}', '-');
$router->addArray('action', array('view', 'edit', 'del'), '/:id');
$values = $router->parse(getenv('PATH_INFO'));

/* @var $loader Loader global */
$loader = new ZeLoader();
$loader->setValues($values);
```

例 3.20. 对象或数组填充

第 4 章 class ZePageSet

4.1. 类摘要

获取常用的分页信息。

```
ZePageSet
{
```

```
__construct(
        int

        $pageSize
      ,

        int

        $page
      );
```

```
int setRecordCount(
        int

        $recordCount
      );
```

```
int getCurrent();
```

```
int current();
```

```
int getPageCount();
```

```
int getPageSize();
```

```
int getRecordCount();
```

```
int getStartRow();
```

```
int getOffset();
```

```
int getEndRow();
```

```
bool hasPrev();
```

```
bool hasNext();
```

```
bool isLast();
```

```
bool isFirst();
```

```
bool isList();
```

```
int first();
```

例 4.1. ZePageSet

```
int prev();
```

```
int next();
```

ZePageSet::__construct
    获取分页信息。

ZePageSet::setRecordCount
    设置记录数，分页信息将会重新计算。

ZePageSet::getCurrent
    获取当前页页码。

ZePageSet::current
    获取当前页页码。

ZePageSet::getPageCount
    获取页面总数。

ZePageSet::getPageSize
    获取单页显示条数。

ZePageSet::getRecordCount
    获取记录总数。

ZePageSet::getStartRow
    获取起始行行号。

ZePageSet::getOffset
    获取起始偏移量。

ZePageSet::getEndRow
    获取当前页结束行行数。

ZePageSet::hasPrev
    是否有前一页。

ZePageSet::hasNext
    是否有后一页页码。

ZePageSet::isLast
    是否是最后一页页码。

ZePageSet::isFirst
    是否是第一页页码。

ZePageSet::isList
    是否有列表当记录数为0时则返回false。

ZePageSet::first
    获取第一页页码。

ZePageSet::prev
    获取前一页页码。

ZePageSet::next
    获取后一页页码。

ZePageSet::last
    获取最后页页码。

ZePageSet::all
    获取所有页面列表。

ZePageSet::siblings
    获取临近页面列表。

ZePageSet::compare
    对比页面索引和当前页并显示相应内容。

ZePageSet::isCurrent
    判断是否为当前页。

4.2. __construct

描述

```
__construct(
        int

        $pageSize
      ,

        int

        $page
      );
```

本类用于获取分页时常需信息。

参数

pageSize
    单页信息数。

page
    当前页。

范例

```
/* 每页3条，共有30条信息， 当前为第5页 */
$pageSet = new ZePageSet(3, 5);
$pageSet->setRecordCount(30);

$pageSet->isFirst(); /* false */
$pageSet->isLast(); /* false */
$pageSet->hasList(); /* true */
$pageSet->hasNext(); /* true */
$pageSet->hasPrev(); /* true */
$pageSet->current(); /* 5 */
$pageSet->first(); /* 1 */
$pageSet->getOffset(); /* 12 */
$pageSet->getStartRow(); /* 13 */
$pageSet->getEndRow(); /* 15 */
$pageSet->getRecordCount(); /* 30 */
$pageSet->siblings(1); /* array(5) */
$pageSet->siblings(5); /* array(3, 4, 5, 6, 7) */
$pageSet->siblings(6); /* array(3, 4, 5, 6, 7, 8) */
$pageSet->all(); /* array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10) */
```

例 4.2. ZePageSet完整范例

```
MySQL 中的使用
[LIMIT {[offset,] row_count | row_count OFFSET offset}]

sql = "LIMIT :offset,:pageSize "
offset = $pageset->getOffset();
pageSize = $pageset->getPageSize();

sql = "BETWEEN :startRow and :endRow "
startRow = $pageset->getStartRow();
endRow = $pageset->getEndRow();


H2database中的使用
sql = "LIMIT :limit OFFSET :offset"
limit = $pageset->getPageSize();
offset = $pageset->getOffset();
```

例 4.3. ZePageSet数据库操作范例

4.3. setRecordCount

设置记录数，分页信息将会重新计算。

描述

```
int setRecordCount(
```

```
        int

        $recordCount
    );
```

每次设置记录总数后，分页信息将会被重新计算。

参数

recordCount
    记录总数。

范例

参见ZePageSet完整范例。

4.4. getCurrent

获取当前页页码。

描述

```
int getCurrent();
```

获取当前页码，这是经过计算后的当前页码，并非在构造函数中递入的页码。

返回值

经过计算后的当前页页码。

范例

参见ZePageSet完整范例。

4.5. current

获取当前页页码，本函数与 getCurrent 相同。

4.6. getPageCount

获取页面总数

描述

```
int getPageCount();
```

获取页面总数。

返回值

返回页面总数。

范例

参见ZePageSet完整范例。

4.7. getPageSize

获取单页面信息项数量

描述

```
int getPageSize();
```

获取单页面信息项数量。

返回值

单页面信息项数量。

范例

参见ZePageSet数据库操作范例。

4.8. getRecordCount

获取记录总数。

描述

```
int getRecordCount();
```

获取记录总数，与 setRecordCount 设置相同。

返回值

记录总数。

范例

参见ZePageSet完整范例。

4.9. getStartRow

获取起始行行号。

描述

```
int getStartRow();
```

获取起始行行号。

返回值

起始行号。

范例

参见ZePageSet数据库操作范例。

4.10. getOffset

获取起始偏移量。

描述

```
int getOffset();
```

获取起始偏移量。

返回值

起始偏移量。

范例

参见ZePageSet数据库操作范例。

4.11. getEndRow

获取当前页结束行行号。

描述

```
int getEndRow();
```

获取当前页结束行行号。

返回值

结束行行好。

范例

参见ZePageSet数据库操作范例。

4.12. hasPrev

是否有前一页。

描述

```
bool hasPrev();
```

是否有前一页。

返回值

当前页非第一页则返回 true 否则为 false。

范例

参见ZePageSet完整范例。

4.13. hasNext

是否有后一页页码。

描述

```
bool hasNext();
```

是否有后一页页码。

返回值

当前页非最后一页则返回 true 否则为 false

范例

参见ZePageSet完整范例。

4.14. isLast

是否是最后一页页码。

描述

```
bool isLast();
```

判断当前页是否为最后一页。

返回值

当前页为最后一页时返回 true 否则为 false。

范例

参见ZePageSet完整范例。

4.15. isFirst

是否是第一页页码。

描述

```
bool isFirst();
```

是否是第一页页码。

返回值

当前页为第一页则返回 true 否则为 false。

范例

参见ZePageSet完整范例。

4.16. isList

检查分页列表是否为空。

描述

```
bool isList();
```

检查分页列表是否为空。

返回值

仅当记录数为0时则返回false。

范例

参见ZePageSet完整范例。

4.17. first

获取第一页页码。

描述

```
int first();
```

获取第一页页码。

返回值

始终返回 1，仅为统一格式提供。

范例

参见ZePageSet完整范例。

4.18. prev

获取前一页页码。

描述

```
int prev();
```

获取前一页页码。

返回值

前一页/上一页 页码。

范例

参见ZePageSet完整范例。

4.19. next

获取后一页页码。

描述

```
int next();
```

获取后一页页码。

返回值

后一页/下一页 页码。

范例

参见ZePageSet完整范例。

4.20. last

获取最后页页码。

描述

```
int last();
```

获取最后页页码。

返回值

获取最后页页码。

范例

参见ZePageSet完整范例。

4.21. all

获取所有页码列表。

描述

```
array all();
```

获取所有页码列表。

返回值

所有页码组成的数组。

范例

参见ZePageSet完整范例。

4.22. siblings

获取临近页面列表。

描述

```
array siblings(
        int

        $count

        = 5
      );
```

获取临近页面列表。

返回值

临近$count页的页码列表，$count为页面数最大值。

范例

参见ZePageSet完整范例。

4.23. compare

对比页面索引和当前页并显示相应内容。

描述

```
mixed compare(
        int

        $page
      ,

        mixed

        $return
      ,

        mixed
```

```
        $elseReturn
    );
```

对比页面索引和当前页并显示相应内容。

返回值

页码与当前页“相同”返回 $return 的值，不同则返回 $elseReturn 的值。

范例

参见ZePageSet完整范例。

4.24. isCurrent

判断是否为当前页。

描述

```
bool isCurrent(
        int

        $page
    );
```

判断是否为当前页。

返回值

$page 与当前页码相同时返回 true。

范例

参见ZePageSet完整范例。

第 5 章 class ZeRecorder

5.1. 类摘要

PDO辅助类。

注意

与PDO参数绑定有所不同的是，同一名称可出现多次。

```
      ZeRecorder
     {
```

```
__construct(
          PDO

          $conn
        );
```

```
ZeRecorder query(
          string

          $query
        );
```

```
ZeRecorder getStmt();
```

```
ZeRecorder bind(
          array | object

          $fields
        ,

          array | object

          $types

          = array()
        );
```

```
ZeRecorder fetch(
          int

          $style

          = PDO::FETCH_ASSOC
        );
```

```
ZeRecorder getInt();
```

```
ZeRecorder fetchAll(
          int

          $style

          = PDO::FETCH_ASSOC
        );
```

```
bool execute();
```

例 5.1. ZeRecorder

```
int exec();
```

```
int lastId(
          string

          $name
```

ZeRecorder::__construct
    PDO辅助类。

ZeRecorder::query
    预处理语句。

ZeRecorder::getStmt
    获取 PDOStatement 对象。

ZeRecorder::bind
    绑定参数。

ZeRecorder::fetch
    获取单行记录。

ZeRecorder::getInt
    获取数量（获取第一列的值，并转换为int）。

ZeRecorder::fetchAll
    获取多行记录。

ZeRecorder::execute
    执行请求。

ZeRecorder::exec
    执行请求，并返回受影响的列数。

ZeRecorder::lastId
    最后插入数据的ID。

ZeRecorder::affected
    受影响的列数。

ZeRecorder::begin
    初始化事务。

ZeRecorder::rollback
    回滚事务。

ZeRecorder::commit
    提交事务。

ZeRecorder::close
    关闭连接（默认会自动关闭）。

5.2. __construct

PDO 辅助类。

描述

```
__construct(
        PDO

        $conn
      );
```

载入PDO实例，初始化PDO辅助对象。

参数

conn
    PDO连接实例。

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');
$recorder = new ZeRecorder($conn);
$value = $recorder->query('select :first `first`,:second `second` ')
        ->bind(array(
                    'first' => 11
                  , 'second' => 12
              ));
        ->fetch();

var_export($value); /* array('first' => 11, 'second' => 12) */
```

例 5.2. 简单示例

5.3. query

预处理语句。

描述

```
ZeRecorder query(
        string

        $query
      );
```

预处理SQL语句，参数以冒号":"开头可包含 "a-zA-Z0-9_"，同一参数可出现在多个位置。

参数

query
    SQL语句。

返回值

当前 ZeRecorder 对象。

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');
$recorder = new ZeRecorder($conn);
$value = $recorder->query('select :first `first`,:second `second` ')
        ->bind(array(
                    'first' => 11
                  , 'second' => 12
              ));
        ->fetch();

var_export($value); /* array('first' => 11, 'second' => 12) */
```

例 5.3. 简单示例

5.4. getStmt

获取 PDOStatement 对象。

描述

```
ZeRecorder getStmt();
```

得到 PDO::prepare 的返回值。

返回值

PDOStatement 对象。

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');
$recorder = new ZeRecorder($conn);
$recorder->query('select :first `first`,:second `second` ');

$stmt = $recorder->getStmt();
```

例 5.4. 简单示例

5.5. bind

绑定参数。

描述

```
ZeRecorder bind(
        array | object
```

```
          $fields
        ,
          array | object
          $types
          = array()
        );
```

参数名不需要冒号前缀，可包含"a-zA-Z0-9_"。同一参数在SQL中出现多次，不需要重复绑定。

参数

fields
: 参数值。参数名与值的键值对。

types
: 参数类型。参数名与类型的键值对。

返回值

当前 ZeRecorder 对象。

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');
$recorder = new ZeRecorder($conn);
$value = $recorder->query('select :first `first`,:second `second` ')
                  ->bind(array(
                               'first' => 11
                             , 'second' => 12
                        )
                        , array(
                                'first' => PDO::PARAM_INT
                              , 'second' => PDO::PARAM_STR
                        )
                    )
                  ->fetch();

var_export($value); /* array('first' => 11, 'second' => 12) */
```

例 5.5. 简单示例

5.6. fetch

获取单行记录。

描述

```
  ZeRecorder fetch(
```

```
        int

        $style

        = PDO::FETCH_ASSOC
      );
```

获取单条记录。

参数

style
    返回列形式。

返回值

各列的值。

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');
$recorder = new ZeRecorder($conn);
$value = $recorder->query('select :first `first`,:second `second` ')
                  ->bind(array(
                              'first' => 11
                            , 'second' => 12
                       )
                       , array(
                              'first' => PDO::PARAM_INT
                            , 'second' => PDO::PARAM_STR
                       )
                    )
                  ->fetch();

var_export($value); /* array('first' => 11, 'second' => 12) */
```

例 5.6. 简单示例

5.7. getInt

获取数量。

描述

```
ZeRecorder fetch(
        int

        $style

        = PDO::FETCH_ASSOC
      );
```

获取第一列的值，并转换为int，本方法常用于获取信息数量。

返回值

第一列转换为整数的值。

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');

$recorder = new ZeRecorder($conn);
$value = $recorder->query('select 321 `key` ')
                  ->getInt();

var_export($value); /* 321 */
```

例 5.7. 简单示例

5.8. fetchAll

获取多行记录。

描述

```
ZeRecorder fetchAll(
        int

        $style

        = PDO::FETCH_ASSOC
      );
```

获取结果集。

参数

style
    返回列形式。

返回值

包含多行值的数组。

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');
$recorder = new ZeRecorder($conn);
$value = $recorder->query('select :first `first`,:second `second` ')
                  ->bind(array(
                               'first' => 11
                             , 'second' => 12
                       )
                       , array(
                               'first' => PDO::PARAM_INT
                             , 'second' => PDO::PARAM_STR
                       )
                    )
                  ->fetchAll();

var_export($value); /* array(array('first' => 11, 'second' => 12)) */
```

例 5.8. 简单示例

5.9. execute

执行请求。

描述

```
  bool execute();
```

执行请求。

返回值

执行成功与否。

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');
$recorder = new ZeRecorder($conn);
$value = $recorder->query('select false ')
                  ->execute();

var_dump($value); /* true */
```

例 5.9. 简单示例

5.10. exec

执行请求，并返回受影响的列数。

描述

```
int exec();
```

执行语句成功后，获取受影响的列数。

返回值

受影响列数，语句执行失败则返回 0 。

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');
$recorder = new ZeRecorder($conn);
$value = $recorder->query('select 123 ')
                  ->exec();

var_dump($value); /* 1 */
```

例 5.10. 简单示例

5.11. lastId

最后插入数据的ID。

描述

```
int lastId(
        string

        $name

        = NULL
      );
```

获取最后新增数据的编号或续列等。

参数

name
    序列名。

返回值

ID

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');
$recorder = new ZeRecorder($conn);
$recorder->query('INSERT INTO `article` (`id`,`title`,`content`) VALUES (null, :title, :content)')
         ->bind(array(
                   'title' => 'blah~blah~blah~'
                  , 'content' => 12
            )
         )
         ->execute();
$value = $recorder->lastId();
var_dump($value); /* 1 */
```

例 5.11. 简单示例

5.12. affected

受影响的列数。

描述

```
  int affected();
```

获取受上一操作影响的列数。

返回值

受影响列数

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');
$recorder = new ZeRecorder($conn);
$recorder->query('INSERT INTO `article` (`id`,`title`,`content`) VALUES (null, :title, :content)')
         ->bind(array(
                   'title' => 'blah~blah~blah~'
                  , 'content' => 12
            )
         )
         ->execute();
$value = $recorder->affected();
var_dump($value); /* 1 */
```

例 5.12. 简单示例

5.13. begin

初始化事务。

注意

使用前请确认已将 autocommit 关闭。

描述

```
bool begin();
```

初始化事务。

返回值

执行是否成功。

范例

```
$conn = new PDO('mysql:host=127.0.0.1;port=3308;dbname=test', 'test', 'testtest');
$recorder = new ZeRecorder($conn);
$recorder->begin();
$recorder->query('INSERT INTO `article` (`id`,`title`,`content`) VALUES (null, :title, :content)')
         ->bind(array(
                    'title' => 'blah~blah~blah~'
                  , 'content' => 12
            )
         )
         ->execute();
$value = $recorder->lastId();

var_dump($value); /* 1 */
if (!$value){
    $recorder->rollback();
}
/* todo other thing */
$recorder->commit();
```

例 5.13. 简单示例

5.14. rollback

回滚事务。

描述

```
bool rollback();
```

回滚事务。

返回值

执行是否成功。

范例

查看 ZeRecorder::begin 范例

5.15. commit

提交事务。

描述

```
bool commit();
```

提交事务。

返回值

执行是否成功。

范例

查看 ZeRecorder::begin 范例

5.16. close

关闭连接（默认会自动关闭）。

描述

```
void close();
```

该方法会在构析函数中自动执行。

第 6 章 class ZeRouter

6.1. 类摘要

路由参数分析器。

```
 ZeRouter
{
```

```
ZeRouter add(
          string

          $pattern
        );
```

```
ZeRouter addRegexp(
          string

          $varName
        ,
          string

          $regexp
        ,
          string

          $pattern
        );
```

```
ZeRouter addArray(
          string

          $varName
        ,
          array

          $strs
        ,
          string

          $pattern
        );
```

```
ZeRouter addParamCount(
          int

          $count
        ,
          string

          $pattern
        );
```

例 6.1. ZeRouter

```
ZeRouter addAllRegexp(
          string

          $regexp
        ,
```

ZeRouter::add
    普通规则。

ZeRouter::addRegexp
    正则匹配已出现的变量。

ZeRouter::addArray
    已出现变量在某集合内。

ZeRouter::addParamCount
    参数个数是某值。

ZeRouter::addAllRegexp
    全请求字符串正则匹配。

ZeRouter::append
    追加规则，当满足上一条规则，则追加此规则。

ZeRouter::shiftSep
    切换分隔符（默认分隔符为 "/"）。

ZeRouter::end
    满足上一条则终止匹配。

ZeRouter::parse
    分析请求字符串。

6.2. 参数规则

普通规则 “/:title”

前缀为分隔符（前一规则规定的，默认为”/”），可以省略。

变量名以冒号”:”起始，名称不包含分隔符即可，但建议仅使用字母和下划线。

数组 “/:ids[,]“

与普通规则不同的是，后追加了”[,]“。

其中逗号”,”为值分隔符，可使用其他字符。

如值为 /1,2,3,4 结果为 array(1,2,3,4)。

但当规则为”/:ids[-]” 值便是array(“1,2,3,4")。

Map “/:others{/}”

与普通规则不同的是，后追加了”{/}”。

其中斜杠”/”为键值分隔符，可使用其他字符。

如值为 /label/123/title/mytitle 结果为：

array(‘others’ => array(‘label’ => ’ 123', ‘title’ => ‘mytitle’))

在实践过程中我们通常不需要变量名，规则为”/{/}”或 “{/}”结果为：

array(‘label’ => ’ 123', ‘title’ => ‘mytitle’)

```
规则 $router->addArray('action', array('list'), '/:page')->append('{/}', '-');

链接 /list/1/label/123/title/mytitle

结果 array ( 'action' => 'list', 'page' => '1', 'label' => '123', 'title' => 'mytitle', )
```

例 6.2. 不定参数示例

注意

在使用Map值时、分隔符应与键值分隔符相区分。

实践示例

```
// /ZoeeyCMS/article.php/add/label/123
$router = new ZeRouter();
$router->add('/:action');
$router->addArray('action', array('add'), '{/}', '-');
$router->addArray('action', array('list', 'search'), '/:page')->append('{/}', '-');
$router->addArray('action', array('view'), '/:id/:title');
$router->addArray('action', array('edit'), '/:id');
$values = $router->parse($_SERVER['PATH_INFO']);
```

例 6.3. 单页面实现全crud操作所需的Router规则。

6.3. add

普通规则。

描述

```
ZeRouter add(
        string

        $pattern
      );
```

添加一个无条件激活的参数规则。

参数

pattern
    参数规则。

返回值

当前 ZeRouter 对象。

范例

```
$router = new ZeRouter();
$router->add('/:action');

$router->parse('/add'); /* araay('action' => 'add') */
```

例 6.4. 基本规则

6.4. addRegexp

正则匹配已出现的变量。

描述

```
ZeRouter addRegexp(
        string

        $varName
      ,

        string

        $regexp
      ,

        string

        $pattern
      );
```

当参数匹配指定正则表达式时激活参数规则。

参数

varName
    变量名称。

regexp
    正则表达式。

pattern
    参数规则。

返回值

当前 ZeRouter 对象。

范例

```
$router = new ZeRouter();
$router->add('/:action');
$router->add('action', '/(list)/i', '/:page/:label');
$router->add('action', '/(view|edit)/i', '/:id/:title'); /* 本示例可由更为高效的 addArray 代替，在此仅作实例 */

$router->parse('/view/123/my-title'); /* array('action' => 'view','id' => '123', 'title' => 'my-title') */
```

例 6.5. 正则匹配则激活

6.5. addArray

已出现变量在某集合内。

描述

```
ZeRouter addArray(
        string

        $varName
      ,

        array

        $strs
      ,

        string

        $pattern
      );
```

当参数存在于指定集合时激活参数规则。

参数

varName
    变量名称。

strs
    数据集合。

pattern
    参数规则。

返回值

当前 ZeRouter 对象。

范例

```
$router = new ZeRouter();
$router->add('/:action');
$router->addArray('action', array('view', 'display'), '/:id/:title');

$router->parse('/view/123/my-title'); /* array('action' => 'view','id' => '123', 'title' => 'my-title') */
```

例 6.6. 值存在于集合内则激活

6.6. addParamCount

参数个数是某值。

描述

```
ZeRouter addParamCount(
        int

        $count
      ,

        string

        $pattern
      );
```

当参数个数和指定数量吻合时激活参数规则。

参数

count
    参数数量。

pattern
    参数规则。

返回值

当前 ZeRouter 对象。

范例

```
$router = new ZeRouter();
$router->addParamCount(5, '/:module/:action/:id/:title/:query');

$router->parse('/article/view/123321/title-of-article/?query');
/* array('module' => 'article', 'action' => 'view', 'id' => '123321', 'title' => 'title-of-article', 'query' => '?
```

例 6.7. 参数数量吻合则激活

6.7. addAllRegexp

全请求字符串正则匹配。

描述

```
ZeRouter addAllRegexp(
        string

        $regexp
      ,

        array

        $keys
      );
```

当整个请求参数匹配指定正则表达式时激活参数规则。

参数

regexp
    正则表达式。

keys
    参数名称列表。

返回值

当前 ZeRouter 对象。

范例

```
$router = new ZeRouter();
$router->addAllRegexp('/([a-z]+)-([a-z]+):([\d]+)\-(.*)/i', array('module', 'action', 'id', 'title'));

$router->parse('/article-view:123321-title-of-article');
/* array('module' => 'article', 'action' => 'view', 'id' => '123321', 'title' => 'title-of-article') */
```

例 6.8. 整个请求字符串匹配正则表达式

6.8. append

追加规则。

描述

```
ZeRouter append(
        string
```

```
          $pattern
        ,

          string

          $sep
        );
```

当满足上一条规则，则追加此规则。

参数

regexp
    正则表达式。

sep
    分隔符。

返回值

当前 ZeRouter 对象。

范例

```
$router = new ZeRouter();
$router->add('/:action');
$router->addArray('action', array('list'), '/:page')
        ->append('{/}', '-');
$values = $router->parse('/list/1/label/2/search/mytitle');
/* array('action' => 'list', 'page' => '1', 'label' => '2', 'search' => 'mytitle') */
```

例 6.9. 识别动态参数

6.9. shiftSep

切换分隔符。

描述

```
  ZeRouter shiftSep(
          string

          $separator
        );
```

切换分隔符后，之后解析的参数将以此分隔。

参数

separator
    分隔符（可为字符串）。

返回值

当前 ZeRouter 对象。

范例

```
$router = new ZeRouter();
$router->add(':module/:action')->shiftSep('-');
$router->add(':id')->shiftSep('/');
$router->add(':title');

$values = $router->parse('/article/view/123321-title-of-article');
/* array('module' => 'article', 'action' => 'view', 'id' => '123321', 'title' => 'title-of-article') */
```

例 6.10. 切换分隔符

6.10. end

终止匹配。

描述

```
  ZeRouter shiftSep(
          string

          $separator
        );
```

满足上一条则终止匹配。

参数

separator
    分隔符（可为字符串）。

返回值

当前 ZeRouter 对象。

范例

```
$router = new ZeRouter();
$router->add('/:action');
$router->addArray('action', array('view'), '/:id')->end()
       ->addRegexp('action', '/^[0-9a-z]+$/i', '/:subaction');

$router->parse('/view/123/my-title'); /* array('action' => 'view', 'id' => '123') */
$router->parse('/steup/init_database'); /* array('action' => 'steup', 'subaction' => 'init_database') */
```

例 6.11. 条件性终止

6.11. parse

分析请求字符串。

描述

```
array parse(
        string

        $query
      );
```

分。

参数

query
    请求字符串，如：getenv('PATH_INFO')。

返回值

结果数组，若参数规则中的变量没有找到，则值为null。

范例

```
$router = new ZeRouter();
$router->add('/:action');
$router->addArray('action', array('view'), '/:id')->end()
       ->addRegexp('action', '/^[0-9a-z]+$/i', '/:subaction');

$router->parse(getenv('PATH_INFO'));
```

例 6.12. 分析请求

第 7 章 class ZeStatus

7.1. 类摘要

状态跟踪。

```
        ZeStatus
       {
  __construct(
            string | int

            $label
          ,

            string | int

            $name
          ,

            string | array | object

            $brief
          );

  ZeStatus getLabel();

  ZeStatus getBrief(
            string

            $lang
          );

  ZeStatus getName();

}
```

例 7.1. ZeStatus

ZeStatus::__construct
    状态跟踪。

ZeStatus::getLabel
    类别。

ZeStatus::getBrief
    描述。

ZeStatus::getName
    名称。

7.2. __construct

状态跟踪。

描述

```
__construct(
        string | int

        $label
      ,

        string | int

        $name
      ,

        string | array

        $brief
      );
```

为状态跟踪提供分类和描述。

参数

label
    类别。

name
    名称。

brief
    描述。

返回值

当前 ZeStatus 对象。

范例

```
define('ZE_ERROR','error');

$status = new ZeStatus(ZE_ERROR, 'error.title_unvalied'
                    , array('en-US' => 'need title for article.'
                          , 'zh-CN' => '请认真填写文章标题。'
                    ));

$status->getLabel(); /* 'error' */
$status->getName();  /* 'error.title_unvalied' */
$status->getBrief('en-US');  /* 'need title for article.' */
$status->getBrief('zh-CN');  /* '请认真填写文章标题。' */
```

例 7.2. 全功能示例

7.3. getLabel

类别。

描述

```
ZeStatus getLabel();
```

状态分类主要用于区分处理流程。

返回值

类别信息。

范例

```
define('ZE_INIT', 'init');

$status = new ZeStatus(ZE_INIT);

$status->getLabel(); /* 'init' */
```

例 7.3. 初始状态示例

7.4. getBrief

描述。

描述

```
ZeStatus getBrief(
        string

        $lang
      );
```

取得可阅读的状态描述，支持多语言。这种直接将语言文字写入代码的设计是为了方便集中管理。也可以使用 getName 将描述信息抽离至其他层面进行重组。

参数

lang
: 语言，当描述为数组时，这里填写需要返回的项。

返回值

类别信息。

范例

```
define('ZE_ERROR','error');

$status = new ZeStatus(ZE_ERROR, 'error.title_unvalied'
                    , array('en-US' => 'need title for article.'
                          , 'zh-CN' => '请认真填写文章标题。'
                    ));

$status->getBrief('zh-CN');  /* '请认真填写文章标题。' */
```

例 7.4. 状态描述

7.5. getName

名称。

描述

```
  ZeStatus getName();
```

此名称是对状态的详细标注。可用在细节逻辑上的判断，也可以作为稳定的标注信息，传递到界面中供Js使用。

返回值

名称。

范例

```
define('ZE_ERROR','error');

$status = new ZeStatus(ZE_ERROR, 'error.title_unvalied');

-- browser --

$status->getName();  /* 'error.title_unvalied' */
```

例 7.5. 状态名称