tp3.2 配置文件 配置主从

配置一、

端口一致进行读写分离

```
'DB_TYPE' =>  'mysql',      // 数据库类型
'DB_HOST'               =>  '127.0.0.1,127.0.0.1', // 服务器地址
'DB_NAME'               =>  'lianxi',           // 数据库名
'DB_USER'               =>  'root,root',       // 用户名
'DB_PWD'                =>  'root,root',           // 密码
'DB_PORT'               =>  '3307',         // 端口
'DB_PREFIX'             =>  '',     // 数据库表前缀
'DB_PARAMS'           =>  array(), // 数据库连接参数
'DB_DEBUG'      =>  TRUE, // 数据库调试模式 开启后可以记录SQL日志
'DB_FIELDS_CACHE'       =>  true,         // 启用字段缓存
'DB_CHARSET'            =>  'utf8',       // 数据库编码默认采用utf8
'DB_DEPLOY_TYPE'        =>  1, // 数据库部署方式:0 集中式(单一服务器),1 分布式(主从服务器)
'DB_RW_SEPARATE'        =>  true,        // 数据库读写是否分离 主从式有效
'DB_MASTER_NUM'         =>  1, // 读写分离后 主服务器数量
'DB_SLAVE_NO'           =>  '154', // 指定从服务器序号
```

配置二、(注意：1、在M()的第三个参数中标名使用的那个数据库配置

2、在Model中需要定义数据库连接属性 `protected $connection = 'Read'`)

```
'Read'=>array('DB_TYPE' =>  'mysql',      // 数据库类型
    'DB_HOST'               =>  '127.0.0.1', // 服务器地址
    'DB_NAME'               =>  'lianxi',           // 数据库名
    'DB_USER'               =>  'root',       // 用户名
    'DB_PWD'                =>  'root',           // 密码
    'DB_PORT'               =>  '3307',         // 端口
    'DB_PREFIX'             =>  '',     // 数据库表前缀
    // 'DB_PARAMS'           =>  array(), // 数据库连接参数
    // 'DB_DEBUG'      =>  TRUE, // 数据库调试模式 开启后可以记录SQL日志
    // 'DB_FIELDS_CACHE'       =>  true,         // 启用字段缓存
    'DB_CHARSET'            =>  'utf8',       // 数据库编码默认采用utf8
    // 'DB_DEPLOY_TYPE'        =>  1, // 数据库部署方式:0 集中式(单一服务器),1 分布式(主从服务器)
    // 'DB_RW_SEPARATE'        =>  true,        // 数据库读写是否分离 主从式有效
    // 'DB_MASTER_NUM'         =>  1, // 读写分离后 主服务器数量
    // 'DB_SLAVE_NO'           =>  '154', // 指定从服务器序号
    ),
    'Write' =>array('DB_TYPE' =>  'mysql',      // 数据库类型
    'DB_HOST'               =>  '127.0.0.1', // 服务器地址
    'DB_NAME'               =>  'lianxi',           // 数据库名
    'DB_USER'               =>  'root',       // 用户名
    'DB_PWD'                =>  'root',           // 密码
    'DB_PORT'               =>  '3306',         // 端口
    'DB_PREFIX'             =>  '',     // 数据库表前缀
    // 'DB_PARAMS'           =>  array(), // 数据库连接参数
    // 'DB_DEBUG'      =>  TRUE, // 数据库调试模式 开启后可以记录SQL日志
    // 'DB_FIELDS_CACHE'       =>  true,         // 启用字段缓存
    'DB_CHARSET'            =>  'utf8',       // 数据库编码默认采用utf8
    // 'DB_DEPLOY_TYPE'        =>  1, // 数据库部署方式:0 集中式(单一服务器),1 分布式(主从服务器)
    // 'DB_RW_SEPARATE'        =>  true,        // 数据库读写是否分离 主从式有效
    // 'DB_MASTER_NUM'         =>  1, // 读写分离后 主服务器数量
    // 'DB_SLAVE_NO'           =>  '154', // 指定从服务器序号
    ),
```

注意事项：

1、首先得配置好MySQL主从，可以多主多从

2、在读写分离的情况下，默认第一个数据库配置是主服务器的配置信息，负责写入数据，如果设置了DB_MASTER_NUM参数，则可以支持多个主服务器写入。其它的都是从数据库的配置信息，负责读取数据，数量不限制。每次连接从服务器并且进行读取操作的时候，系统会随机进行在从服务器中选择

3、调用模型的CURD操作的话，系统会自动判断当前执行的方法的读操作还是写操作，如果你用的是原生SQL，那么需要注意系统的默认规则： 写操作必须用模型的execute方法，读操作必须用模型的query方法，否则会发生主从读写错乱的情况

图片配置

```
<?php
return array(
  // '配置项'=>'配置值'

  //缓存
  // 'DATA_CACHE_TYPE'=>'Memcache',
   // 'MEMCACHE_HOST'   =>'tcp://192.168.0.227:11211',
   // 'DATA_CACHE_TIME'=>'3600',

    'DEFAULT_CONTROLLER'    =>    'Index',// 默认控制器名称
    'DEFAULT_ACTION'        =>    'index',// 默认操作名称

  //数据库配置 beruiuser
    'DB_CONFIG1'=> array(
      'db_type'  =>  'mysql',
      'db_user'  =>  'root',
      'db_pwd'   =>  'aisijkl81',
      'db_host'  =>  '192.168.0.111',
      'db_port'  =>  '3306',
      'db_name'  =>  '2222',
  ),//里面存在跨库操作

  //数据库配置 br_esfmanager
    'DB_CONFIG2'=> array(
      'db_type'  =>  'mysql',
      'db_user'  =>  'root',
      'db_pwd'   =>  'aisijkl8*1',
      'db_host'  =>  '1111111',
      'db_port'  =>  '3306',
      'db_name'  =>  '444',
  ),//里面存在跨库操作

    'domain'=>  'demo',//这里上线后应该为api
```

代码里使用

```
$data=M()->db(2,"DB_CONFIG2")->table("mn_r_house_web")->where($where)->find();
```

mode使用案例

```
<?php
namespace Home\Model;
use Think\Model;
class BisaiModel extends Model
{
  //查找
  protected $connection = 'Read';
   public function s()
   {
    // $this->connection="Read";
    return $this->db(2,"Read")->select();
   }
}
```

M用法

```php
namespace Home\Controller;
use Think\Controller;
class IndexController extends Controller {

    public function index(){
        // print_r(M("bisai","","Read")->select());die;
        // print_r(M("address")->select());
        // $arr=['name'=>222];
         var_dump(M("bisai","","Write")->add($arr));
        print_r(D("Bisai")->s());
    }
}
```

M流程：

调用commen下的functions.php下面的M(表名，表前缀，调用那个数据库配置)方法

```php
/**
 * 实例化一个没有模型文件的Model
 * @param string $name Model名称 支持指定基础模型 例如 MongoModel:User
 * @param string $tablePrefix 表前缀
 * @param mixed $connection 数据库连接信息
 * @return Think\Model
 */
function M($name='', $tablePrefix='',$connection='') {
    static $_model  = array();
    if(strpos($name, needle: ':')) {
        list($class,$name)    =  explode( delimiter: ':',$name);
    }else{
        $class       =    'Think\\Model';
    }
//    echo $tablePrefix;die;
    $guid           =    (is_array($connection)?implode( glue: '',$connection):$connection).$tabl
//    echo $class;die;
    if (!isset($_model[$guid]))
        $_model[$guid] = new $class($name,$tablePrefix,$connection);
//    var_dump($_model);die;
    return $_model[$guid];
}
```

实例化 Model.class.php

```php
    /**
     * 架构函数
     * 取得DB类的实例对象 字段检查
     * @access public
     * @param string $name 模型名称
     * @param string $tablePrefix 表前缀
     * @param mixed $connection 数据库连接信息
     */
    public function __construct($name='',$tablePrefix='',$connection='') {
        // 模型初始化
        $this->_initialize();
        // 获取模型名称
        if(!empty($name)) {
            if(strpos($name, needle: '.')) { // 支持 数据库名.模型名的 定义
                list($this->dbName,$this->name) = explode( delimiter: '.',$name);
            }else{
                $this->name   =  $name;
            }
        }elseif(empty($this->name)){
            $this->name =   $this->getModelName();
        }
        // 设置表前缀
        if(is_null($tablePrefix)) {// 前缀为Null表示没有前缀
            $this->tablePrefix = '';
        }elseif('' != $tablePrefix) {
            $this->tablePrefix = $tablePrefix;
        }elseif(!isset($this->tablePrefix)){
            $this->tablePrefix = C('DB_PREFIX');
        }
//      echo $connection;die;
        // 数据库初始化操作
        // 获取数据库操作对象
        // 当前模型有独立的数据库连接信息
        $this->db( linkNum: 0, config: empty($this->connection)?$connection:$this->connection, force: true);
    }
```

调用DB方法

```php
    /**
     * 切换当前的数据库连接
     * @access public
     * @param integer $linkNum  连接序号
     * @param mixed $config  数据库连接信息
     * @param boolean $force 强制重新连接
     * @return Model
     */
    public function db($linkNum='',$config='',$force=false) {
//        echo $config;die;
        if('' === $linkNum && $this->db) {
            return $this->db;
        }

        if(!isset($this->_db[$linkNum]) || $force ) {
            // 创建一个新的实例
            if(!empty($config) && is_string($config) && false === strpos($config, needle: '/')) { // 支持读取
                $config  =  C($config);
            }
            $this->_db[$linkNum]            =    Db::getInstance($config);
        }elseif(NULL === $config){
            $this->_db[$linkNum]->close(); // 关闭数据库连接
            unset($this->_db[$linkNum]);
            return ;
        }

        // 切换数据库连接
        $this->db    =    $this->_db[$linkNum];
        $this->_after_db();
        // 字段检测
        if(!empty($this->name) && $this->autoCheckFields)    $this->_checkTableInfo();
        return $this;
    }
    // 数据库切换后回调方法
```