tmf 1.0.0 API

1.MVC 模式

<table>
<tr><td colspan="3">初始化框架,设置文件自动加载函数</td><td>Tmf::$tmfVersion
Tmf::$tmfPath</td></tr>
<tr><td rowspan="11">创建应用</td><td colspan="2">分析 http 头部
App::$http_request = new HttpRequest();
App::getProtectedDir()</td><td>App::$http_request</td></tr>
<tr><td colspan="2">保存应用根目录</td><td>Tmf::$appPath</td></tr>
<tr><td rowspan="6">分析配置文件</td><td>是否调试模式</td><td>App::$debug</td></tr>
<tr><td>出错控制器与行为</td><td>App::$errorConAct</td></tr>
<tr><td>受保护的代码的父文件夹名称</td><td>App::$protected_name</td></tr>
<tr><td>全局的配置文件</td><td>App::$app_config</td></tr>
<tr><td>默认控制器</td><td>App::$default_controller</td></tr>
<tr><td>初始化当前 url 的样式｛$action 如果不是控制器默认行为(即：$action 为空)，$action 此时就可以使用｝</td><td>App::$url_manager
App::$controller
App::$action</td></tr>
<tr><td>数据库操作对象</td><td>App::$db</td></tr>
<tr><td>当前用户</td><td>App::$user</td></tr>
<tr><td colspan="2">开启 session,自动登陆模块(from cookie)</td><td>$_SESSION</td></tr>
<tr><td rowspan="6">创建分发器，执行分发</td><td colspan="2">创建分发器</td><td>App::$dispatcher</td></tr>
<tr><td rowspan="5">执行分发</td><td>创建控制器</td><td>dispatch->createController()</td></tr>
<tr><td>设置默认动作
$action 若为默认行为，设置默认行为</td><td>App::$default_action
App::$action
dispatch->setDefaultAction()</td></tr>
<tr><td>对该动作进行过滤</td><td>dispatch->isFilter()</td></tr>
<tr><td>在开始动作前的操作</td><td>$controller->beforeAction()</td></tr>
<tr><td>执行动作</td><td>$this->doAction()</td></tr>
</table>

2.HttpRequest 对象

getRequestUri()	获取相对 Uri (如：/test/)
getHostName()	返回主机名称（如：localhost）
getReferrerURL()	返回链接/提交当前页的父页面 URL.
getUserAgent()	获取客户端信息
getUserIp()	获取访问者的 IP 地址
getAbsoluteScriptFile()	获取请求脚本的绝对文件路径（如：C:\www\test\index.php）
getAbsoluteBasePath()	获取应用根目录（如：C:\www\test）
getScriptURL()	获取请求脚本相对 URL（如：/test/index.php）

getServerPort()	获取服务器端，端口号
isHttps()	判断是否为 HTTPS 请求
getHostURL($schema=null)	获取主机 url（如：http://localhost）
getBaseURL($absolute=false)	获取相对路径 URL（如：/test）
getAbsoluteBaseUrl()	获取绝对路径 URL（如：http://localhost/test）

3.urlManager 对象（用于 url 优化）

createUrl($controller,$action,$data = null)	创建相对 URL
createAbsoluteUrl($controller,$action,$data = null)	创建绝对路径 URL
analyseUrl($url=null)	分析 URL 得出控制器，行为，get 参数

4.Db 对象（数据库操作对象，用于操作数据库）

getPdoIns()	通过配置文件获取 Pdo 实例对象
query($sql,$params= null)	查询 返回结果集
exec($sql,$params= null)	查询 返回受影响的行数
delete($table,$where)	删除
save($table,$data,$pk_name=null,$pk_value=null)	保存或添加
getError()	获取错误信息
getLastInsertId($pk_name=null)	返回最后插入行的 ID 或序列值

5.WebUser 对象（用户对象）

login(Identity $identity)	登陆
logout()	退出
IsLogin()	是否是登陆状态
loginFromCookie()	通过 cookie 进行登陆
hasSession($key)	判断$key 是否在 session 中
getSession($key)	获取$key 在 session 中的值，如果$key=’sid’,获得唯一用户标识符
setSession($key,$value)	设置$key 在 session 中值

6.Identity 对象（身份验证对象）

$username	登陆验证名称
$password	为加密后的密码
$autoLogin	是否自动登陆
__construct($un,$pwd,$autoLogin = true)	构造函数
authenticate()	返回 boolean,false-验证不通过，true-验证通过
getId()	返回用户唯一标识符

7.Dispatcher 对象(分发动作)

$filter_reason	过滤失败的原因
$controller	当前控制器对象
createController()	创建控制器
setDefaultAction()	设置默认动作
doAction()	执行相应的动作
isFilter(Controller $controller)	是否能访问该行为，false-不能，true-能
dispatch()	分发
toError()	跳至用户配置的出错页面

8.Controller 对象

$view_suffix	视图文件后缀（.php;.tpl,.html）
$filter	过滤器配置
$layout	是否加载布局文件
$defaultAction	默认行为
getCurrentActionFilterName()	获取当前行为的$filter 名称，一个数组
createUrl($base_info,$data = null)	创建一个相对路径
createAbsoluteUrl($base_info,$data=null)	创建一个绝对路径
beforeAction()	在开始行为前需要做的事情
render($base_info,$data=null,$return=false)	渲染视图
forward($base_info,$data=null)	请求其他控制器的方法资源
redirectByUrl($url,$terminate=true, $statusCode=302)	通过 url 跳转
redirect($base_info,$data=null,$absolute = true, $terminate = true, $statusCode = 302)	跳转到该域下

9.Model 对象

query($sql, $params = null)	查询
exec($sql, $params = null)	通过$sql 返回受影响的行数（不能执行 select）
save($datas, $pk_value = null)	保存、添加
delete($where=null)	删除
getLastInsertId($pk_name = null)	获取插入 ID
getError()	获取错误信息
find($sql, $param = null)	返回一个结果
deleteByPk($pk_v)	通过主键删除
findByPk($pk_v, $fields)	通过主键查找

10.配置文件

```
return array(
    'debug' => true,
    'defaultController' => 'user',
```

```
    'urlManager'=>'tmf\web\urlManager\SimplifyUrl',
    'errorHandler'=>'site/to404',
    /**
     * db
     */
    'db'=>array(
        'type' => 'mysql',
        'one'    => array(
                'connectionString' => 'mysql:host=127.0.0.1;dbname=exam',
                'username' => 'root',
                'password' => '123456',
                'charset' => 'utf8'
            )
    ),
    'identity'=>'code\components\NormalIdentity',
    'filter'=>array(
        'Provider'=>array(
            'code\components\filters\LoginFilter'=>'*',
            'test2'=>'index,Papers'
        )
    )
);
```

10.1.配置表

配置项	默认值	说明
debug	false	是否开启调试模式
defaultController	index	设置路由默认控制器，默认行为在具体控制器中设置
urlManager	tmf\web\urlManager\NormalUrl	设置 url 解析方式
errorHandler		设置，框架出错显示页面
db	必须有，没有就没有数据库操作对象	设置数据库操作对象
identity	code\components\NormalIdentity	设置验证对象，通过它，可以设置不同的验证方式
filter	没有就不过滤	设置各个控制器的过滤方式

10.2.filter 设置说明

在配置文件中(‘*’表示任何)

```
'filter'=>array(
        '*'=>array(
            '过滤器名称 1'=>'行为 3,行为 4',
            '过滤器名称 4'=>'*'
        ),
        '控制器名称 1'=>array(
```

```
            '过滤器名称 1'=>'行为 1,行为 2',
            '过滤器名称 2'=>'*'
            ),
        '控制器名称 2'=>array(
            '过滤器名称 1'=>'*',
            '过滤器名称 3'=>'行为 1,行为 2，行为 3'
            )
        )
```

控制器内部设置过滤器(‘*’表示任何)

```
public $filter=array(
        '过滤器名称 3'=>'行为 4,行为 5',
        '过滤器名称 5'=>'*'
        );
```

说明：如果存在同样的过滤器，后者会把前者的配置覆盖掉（优先级：配置文件中<控制器内部设置，前边设置<后边设置）

例如：

配置文件<控制器设置	当前控制为‘控制器 1’，‘过滤器 1’应该为'行为 1,行为 2';而不是'行为 1,行为 2,行为 3,行为 4'，或者'行为 3,行为 4'
前边设置<后边设置	当前控制为‘控制器 2’，‘过滤器 3’应该为'行为 4,行为 5';而不是'行为 1,行为 2,行为 3,行为 4,行为 5'，或者'行为 1,行为 2',行为 3'

11.App(公共对象)

$http_request	请求信息（解析$_SERVER）
$debug	是否是调试模式，非调试模式将关闭错误报告
$errorConAct	出错控制器与行为(如：site/error)
$protected_name	受保护的代码的父文件夹名称
$app_config	全局的配置文件
$default_controller	默认控制器
$url_manager	初始化当前 url 的样式
$controller	当前请求的控制器名称
$db	数据库操作对象
$user	当前用户
$dispatcher	当前分发器
$default_action	默认行为
$action	当前请求的行为名称
getProtectedDir()	获取代码受保护的文件夹路径（包含控制器，视图，模型层等）

12.Application(应用对象)

__construct($config)	应用初始化
run()	运行应用

end()	结束运行程序

13.BaseValid（验证类，这里面的函数全是静态函数）

required($var)	不为空
email($var)	输入字符串为 email
url($var,$params=null)	输入字符串是一个 url
boolean($var)	是一个 boolean 值
noScriptTags($str)	从字符串中去除 HTML 和 PHP 标记
string($var)	是一个字符串
minLength($var,$min)	是一个字符串，长度大于等于 mix
maxLength($var,$max)	是一个字符串，长度小于等于 max
rangeLength($var,$param)	是一个字符串，长度小于等于 max,大于等于 min
int($var)	是一个整数
intMin($var,$min)	是一个整数，值大于等于 min
intMax($var,$max)	是一个整数，值小于等于 max
intRange($var,$param)	是一个整数（范围（min,max））
number($var)	是一个数
Max($var,$max)	是一个数，值小于等于 max
Min($var,$min=0)	是一个数，值大于等于 min
Range($var,$param)	是一个数（范围（min,max））
regexp($var,$string)	通过正则表达式验证
nameInArr($name,$arr)	在数组 arr 中,是否存在键名为 name(name 可以为数组)的值。
validate($var,$params)	一个变量，多次验证
validates($arr,$params)	多个变量，同样的多次验证

14.BaseTool(公共工具类，函数全为静态函数)

getRandString($length)	获取$length 长度的随机字符
getRandShortString()	获取随机文件名
getDateDir($root_dir)	创建一个目录(年-月)

15.BaseFileUpload(文件上传类)

protected $filename	上传时的文件名称（<input> 中的 name 属性）
protected $typeList	允许上传文件类型 MIME(字符/数组)
protected $maxSize	允许上传的文件大小，单位：MB
public $error	验证文件时，失败的原因
public $path	保存文件的路径
__construct($name,$typeList=null,$size=null)	构造函数，生成基本的文件上传类
validate()	验证文件是否上传成功

saveByName($dir,$name)	通过名字保存文件

16.