微擎执行主流程介绍

首先对微擎的工作原理做简单描述, 微擎使用规则和模块的机制来处理公众平台的请求数据并返回响应的结果.

执行流程描述为: 粉丝用户与公众号码进行对话或交互, 而后公众平台将粉丝用户的请求消息(当前包括: 文本, 图片, 位置, 链接, 事件. 请参阅消息类型)传递给微擎系统, 微擎系统按照消息类型和对应的公众号所设定的规则列表匹配到合适的规则(请参阅消息路由), 规则定义中包括处理此消息所使用的模块和此模块处理消息时所需要的其他附加数据(请参阅模块定义), 而后模块将会按照请求的消息数据和模块附加数据进行相关业务处理并返回处理结果(请参阅响应类型), 微擎系统将处理结果返回给公众平台, 而后经公众平台返回给粉丝用户.

微擎的主处理流程实现定义于: engine.php 中, 如有需要请参阅源码.

消息类型

消息类型指粉丝用户通过公众平台与微擎进行交互时所提供的消息数据的类型. 消息类型基于微信公众平台提供的数据接口(http://mp.weixin.qq.com/wiki/index.php?title=%E6%B6%88%E6%81%AF%E6%8E%A5%E5%8F%A3%E6%8C%
微擎在其基础上进行包装处理, 方便PHP开发者进行业务处理. 保存消息的基本对象结构定义为:

```
$message => array(
    'from' => '',  //string: 发送消息方, 代表一个粉丝用户(使用OpenID表示)
    'to' => '',    //string: 消息接收方, 对应当前的公众号(使用OpenID表示)
    'time' => '',  //int: 消息发送时间, 使用Unix时间戳表示
    'type' => '',  //string: 消息类型, 用于区分不同类型的消息, 请参阅下文
    'msgid' => ''  //int: 消息ID, 公众平台系统用于唯一标识一条请求消息
);
```

消息类型同公众平台官方不同之处在于将event类型拆分开为独立的消息类型, 避免了重复判断. 根据消息类型不同, 消息对象结构还存在不同的附加数据, 按照类型定义如下:

文本消息

粉丝用户向公众号发送了一条普通文本消息(包括包含表情的消息, 或者纯表情消息), 处理文本消息可以实现简单的文本对话, 结合使用文本上下文(请参阅上下文处理)可以实现调查, 测试等复杂的交互.

```
$message => array(
    //....全局数据
    'type' => 'text',                   //string: 代表当前消息为文本消息
    'content' => ''                     //string: 文本消息内容
);
```

图片消息

粉丝用户向公众号发送了一张图片, 处理图片消息可以实现分享用户图片的相关功能

```
$message => array(
    //....全局数据
    'type' => 'image',                  //string: 代表当前消息为图片消息
    'url' => 'http://same.im/image.jpg' //string: 用户所发送的图片地址
);
```

地理位置消息

粉丝用户向公众号发送了一条地理位置, 处理地理位置消息可以实现lbs相关功能(参阅LBS方案)

```
$message => array(
    //....全局数据
    'type' => 'location',               //string: 代表当前消息为位置消息
    'location_x' => '',                 //float: 代表位置经度
    'location_y' => '',                 //float: 代表位置纬度
    'scale' => '',                      //int: 表示地图缩放倍数
    'label' => ''                       //float: 表示地点描述
);
```

链接消息

粉丝用户向公众号发送了一条链接消息, 处理链接消息可以实现好友分享等社交功能

```
$message => array(
    //....全局数据
    'type' => 'link',                       //string: 代表当前消息为链接消息
    'title' => '',                          //float: 代表链接标题
    'description' => '',                    //float: 代表链接描述信息
    'scale' => 'url'                        //int: 表示链接的URL
);
```

> ### 关注消息
>
> 粉丝用户关注当前公众号后将会获得此消息, 处理此消息可以实现欢迎信息和粉丝增长统计

```
$message => array(
    //....全局数据
    'type' => 'subscribe'                   //string: 代表当前消息为关注消息
);
```

> ### 取消关注消息
>
> 粉丝用户取消关注当前公众号后将会获得此消息, 处理此消息可以实现粉丝数量增长分析

```
$message => array(
    //....全局数据
    'type' => 'unsubscribe'                 //string: 代表当前消息为取消关注消息
);
```

> ### 菜单点击消息
>
> 粉丝用户点击自定菜单后, 如果菜单设置为消息回复, 那么将会获得此消息, 处理此消息能实现自定义菜单的特定回复

```
$message => array(
    //....全局数据
    'type' => 'CLICK',                      //string: 代表当前消息为菜单点击消息
    'eventkey' => ''                        //string: 菜单点击附加的菜单数据信息
);
```

消息路由

消息路由是指粉丝用户经公众平台发送消息内容至微擎时, 微擎系统查找对应的规则记录, 并将消息分配至合适的模块处理的过程. 微擎系统按照不同的消息类型, 进行不同的处理. 处理方式如下:

上下文消息路由

微擎支持上下文操作, 通过上下文支持微擎可将用户对话锁定至特定的模块, 如果当前消息是上下文对话的消息, 那么将会自动路由至上下文锁定的模块. (请参阅 上下文处理)

'

文本消息规则匹配(重要)

针对文本消息, 微擎使用文本匹配来选择合适的规则和模块, 规则是指针对特定消息的处理方式. 微擎选择规则的方式包括:

```
* 关键字包含        指粉丝用户发送的消息内容含有指定的关键字就指派到特定规则.
* 内容等价          指粉丝用户发送的消息内容完全等于指定的内容才指派到特定规则.
* 正则表达式        指粉丝用户发送的消息类型符合指定正则表达式定义的模式时指派到特定规则.(高级模式, 需要有编程经验)
```

其他类型消息路由规则

图片消息, 位置消息, 链接消息等其他类型请求消息的路由支持正在紧张开发中

模块定义

模块是微擎用于处理用户请求消息的核心组件. 其作用是分析用户请求消息, 并根据业务需要来返回处理结果. 微擎将处理模块设计为开放的, 可扩展的模块式插件. 通过设计模块可以快速准确处理微信运营中的各种业务需求. 降低企业开发成本, 更快速有效的为营销提供服务. 定义模块需要继承 WeModule 和 WeModuleProcessor 这两个类的特定函数成员来实现.

WeModule 介绍

WeModule 用户定义一个独立的功能模块定义, 微擎系统将在用户管理界面使用 WeModule 的派生类来管理, 配置和显示此模块的相关功能. WeModule类的定义及派生时要实现的成员描述如下:

```
abstract class WeModule {
    /**
     * 需要附加至规则表单的字段内容, 编辑规则时如果模块类型为当前模块, 则调用此方法将返回内容附加至规则表单之后
```

```
     * @param int $rid 如果操作为更新规则, 则此参数传递为规则编号, 如果为新建此参数为 0
     * @return string 要附加的内容(html格式)
     */
    public function fieldsFormDisplay($rid = 0) {
        return '';
    }

    /**
     * 验证附加至规则表单的字段内容, 编辑规则时如果模块类型为当前模块, 则在保存规则之前调用此方法验证附加字段的有效性
     * @param int $rid 如果操作为更新规则, 则此参数传递为规则编号, 如果为新建此参数为 0
     * @return string 验证的结果, 如果为空字符串则表示验证成功, 否则返回验证失败的提示信息
     */
    public function fieldsFormValidate($rid = 0) {
        return '';
    }

    /**
     * 编辑规则时如果类型为当前模型, 则在提交成功后调用此方法
     * @param int $rid 规则编号
     * @return void
     */
    public function fieldsFormSubmit($rid) {
        //
    }

    /**
     * 在列表中删除规则如果类型为当前模型, 则在删除成功后调用此方法, 做一些删除清理工作。
     * @param int $rid 规则id,必填值
     * @return <boolean | error> 如果操作成功则返回true,否则返回error信息
     */
    public function ruleDeleted($rid) {
        return true;
    }

    /**
     * 如果模块需要配置附加的参数, 请在此方法内部输出配置项的表单元素, 系统将在表单 post 提交之后保存所有的的 post 字段
     * @param array $settings 已保存的配置项数据
     * @return void
     */
    public function settingsFormDisplay($settings = array()) {
        //
    }

    /**
     * 展示特定模板内容
     * @param string $filename 模板名称, 如: settings 将会展示本模块定义下的 template/settings.html 模板文件, 请参阅 "模板机制"
```

```
     * @return void
     */
    public function template($filename, $flag = TEMPLATE_INC
LUDEPATH) {}
}
```

WeModuleProcessor 介绍

WeModuleProcessor 用户定义的模块处理程序, 当匹配此模块的消息到来时, 使用此对象处理对象并返回处理结果.

WeModuleProcessor 执行流程描述如下:
- 系统根据上一步匹配到的模块创建对应的 Processor对象
- 系统上一步获得的模块附加数据初始化 $message, $inContext, $rule, $module 成员
- 系统调用 respond 方法, 获取响应内容并返回给微信接口
- 调用结束

WeModuleProcessor 的成员及作用描述如下:

```
abstract class WeModuleProcessor {
    // array: 本次请求消息, 此属性由系统初始化, 消息格式请参阅 "消息类型"
    public $message;

    // bool: 本次对话是否为上下文响应对话, 如果当前对话是由上下文锁定而路由到的. 此值为 true, 否则为 false
    public $inContext;

    // int: 本次请求所匹配的规则编号, 此属性由系统初始化, 如果为上下文对话, 那么此值为 -1
    public $rule;

    // array: 本次请求所匹配的处理模块, 此属性由系统初始化
    public $module;

    // void: 开始上下文对话, 附加的参数说明超时
    protected function beginContext($expire = 3600);

    // void: 结束上下文对话
    protected function endContext();

    /**
     * 应答此条请求. 如果响应内容为空. 将会调用更低级优先的模块, 直到默认回复为止
     * @return string
     */
    abstract function respond();
}
```

上下文处理

微擎现已支持上下文锁定对话, 可以将粉丝用户的对话锁定至特定模块. 用以实现在线调查, 在线测试等类似的功能. 微擎的上下文操作使用 $_SESSION 来实现, 在上下文对话中, 可以使用 $_SESSION 来保存会话数据. 要实现上下文操作主要使用 Processor 里的内定方法:

public inContext;
本次对话是否为上下文响应对话, 如果当前对话是由上下文锁定而路由到的. 此值为 true, 否则为 false

protected function beginContext($expire = 3600);
请在模块处理程序中调用此函数已开始一个新的上下文对话.
附加的参数 $expire 说明本次会话的失效时间. 例如 $this->beginContext(1800) 就说明启动一次上下文会话锁定, 并且本次会话将会于30分钟后释放, 如果不固定超时的话, 请在每次请求时调用 beginContext

protected function endContext();
在模块处理程序中调用 endContext 来结束一次会话, 并销毁会话中保存的所有数据(当前为 $_SESSION)

下面的示例将演示根据血型查性格的简单例子:

```
class BloodTestModuleProcessor extends WeModuleProcessor {
    //void: 所有处理程序必须实现虚函数 respond. 用以响应消息
    public function respond() {
        if(!$this->inContext) {
            $reply = '请输入你的血型(A, B, O, AB), 来分析你今年的运程. ';
            $this->beginContext();
            // 如果是按照规则触发到本模块, 那么先输出提示问题语句, 并启动上下文来锁定会话, 以保证下次回复依然执行到本模块
        } else {
            $btypes = array('a', 'b', 'o', 'ab');
            $b = strtolowwer($this->message['content']);
            // 如果当前会话在上下文中, 那么表示当前回复是用户回答提示问题的答案.
            if(in_array($b, $btypes)) {
                switch($b) {
                    case 'a':
                       $reply = 'A型血今年.....';
                       break;
                    case 'b':
                       $reply = 'B型血今年.....';
                       break;
                    case 'o':
                       $reply = 'O型血今年.....';
                       break;
                    case 'ab':
                       $reply = 'AB型血今年.....';
                       break;
                }
                $this->endContext();
                // 如果当前回答符合答案格式, 那么进行保存并进行下
```

```
一个问题. (可以保存至 SESSION 中)
                    // 直到最后一个问题回答完成, 输出测试结果给用户,
并结束对话锁定. 以保证用户其他对话能正常路由.
                    // 本示例只有一个问题, 因此不保存答案, 直接输出测
试结果.
                    // 如果对话默认的超时不够, 那么可以在每次提出下一
个问题的时候重新调用 beginContext 来顺延超时.
                } else {
                    $reply = '请输入正确的血型(A, B, O, AB). ';
                    // 回答不符合答案格式, 那么重新显示当前问题.
                }
            }

            return $this->respText($reply);
            // 返回至系统
        }

        private function respText($content) {
            $response = array();
            $response['FromUserName'] = $this->message['to'];
            $response['ToUserName'] = $this->message['from'];
            $response['MsgType'] = 'text';
            $response['Content'] = htmlspecialchars_decode($cont
ent);
            return $response;
        }
}
```

LBS解决方案

基于位置的数据解决方案正在紧张开发中