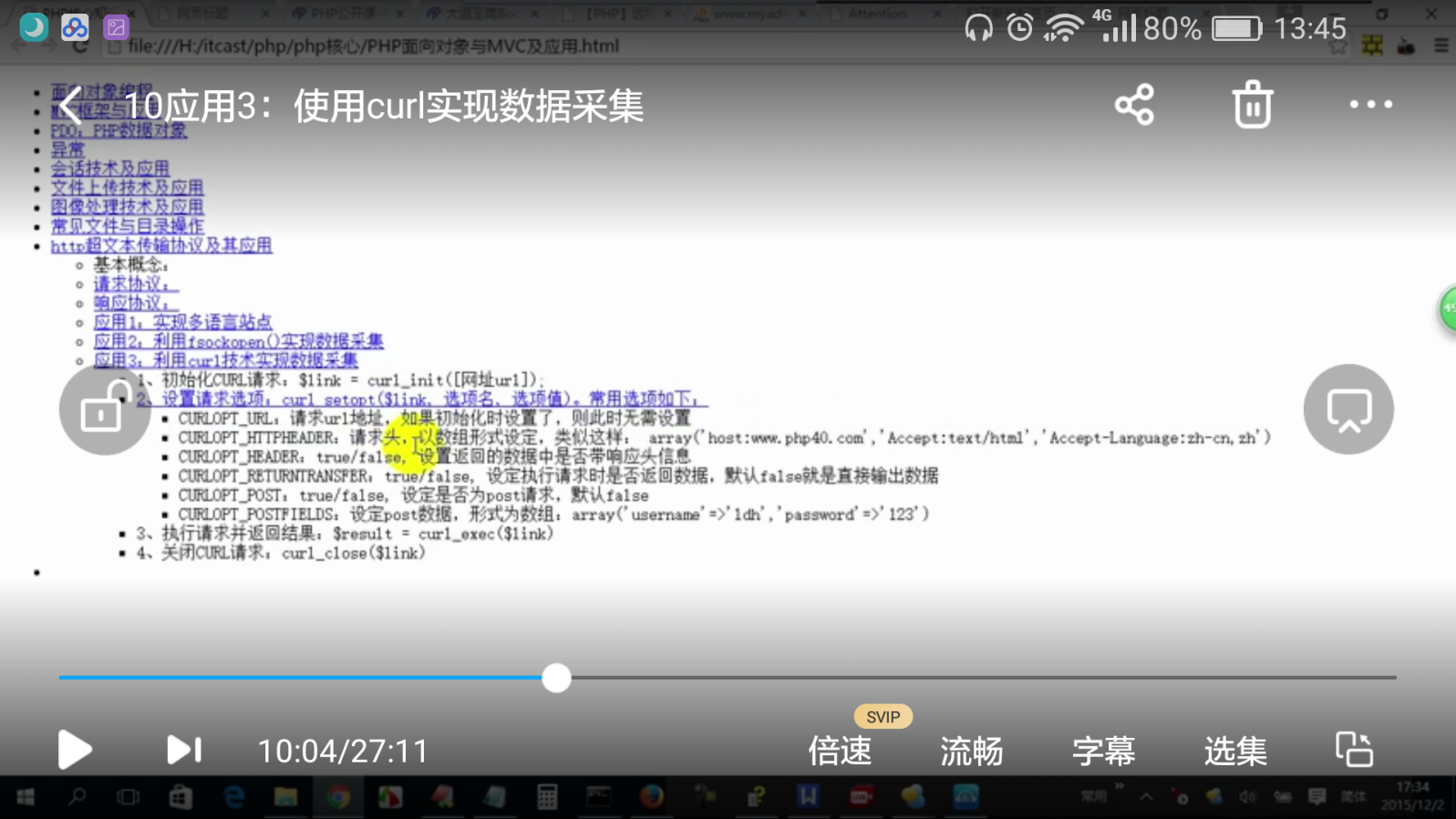
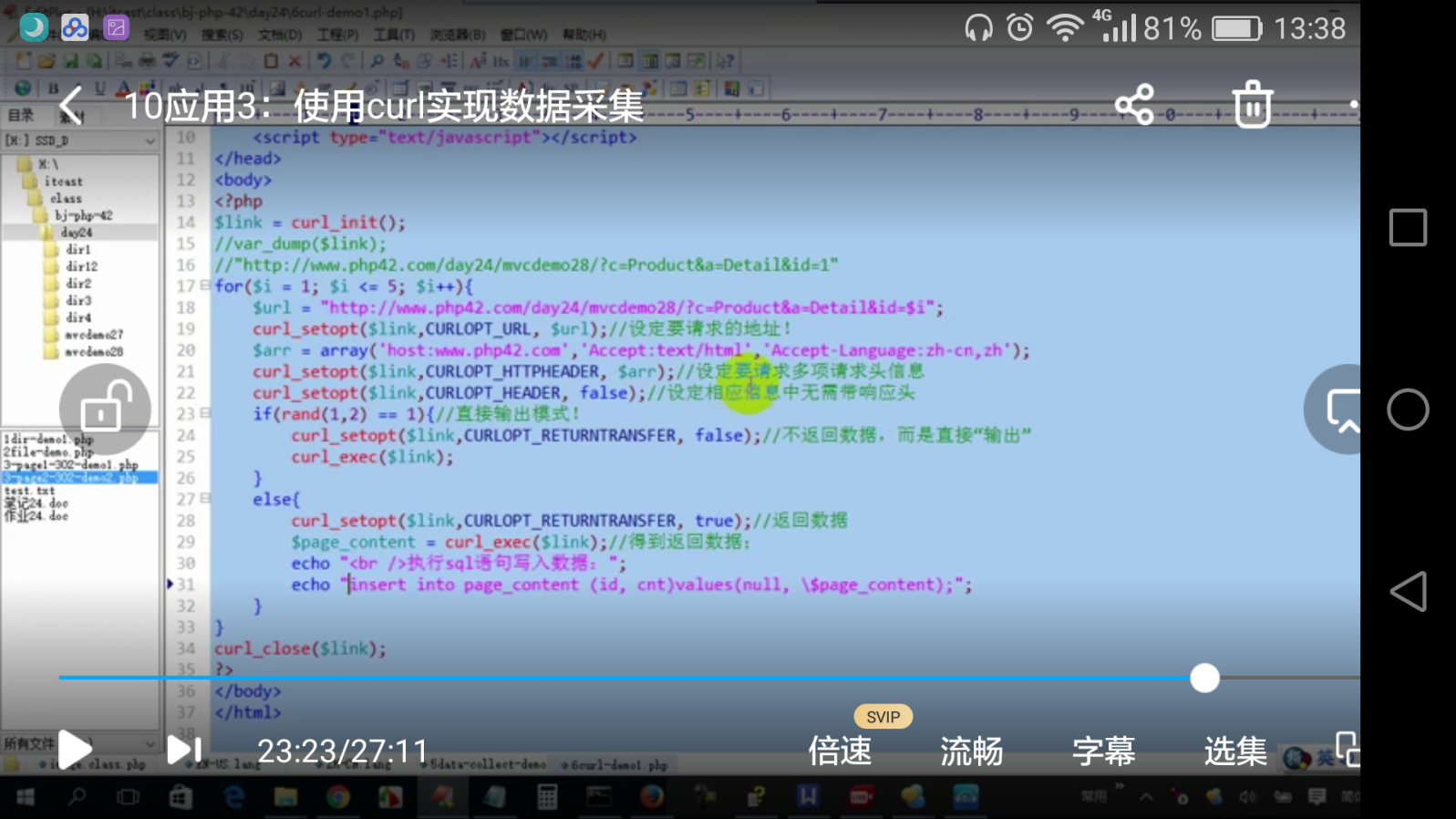
### 使用curl实现数据采集





如果curl\_init()函数报错，则尝试将php语言包中的libeay32.dll和ssleay32.dll文件copy到window目录中。

你可以通过curl\_setopt灵活地设置请求选项，这里面有很多的可选项，具体可以参考：<http://cn2.php.net/manual/zh/function.curl-setopt.php>

## **三、错误处理**

在上述代码中，你也可以增加错误处理的代码：

$response = curl\_exec($ch);

if ($response === FALSE) {

echo "cURL 具体出错信息: " . curl\_error($ch);

}

注意了，在做上述判断时务必要使用===，因为请求的回复可能是空字符串，curl在请求出错的情况下回返回FALSE值，所以我们必须使用===，而不是==。

## **四、获取curl请求的具体信息**

在执行一个cURL请求后，你也可以使用curl\_getinfo获取该请求的具体信息:

curl\_exec($ch);

$curl\_info= curl\_getinfo($ch);

echo "收到的http回复的code为： {$curl\_info['http\_code']}";

上述$curl\_info是一个关联数组，可以从中获取很多的具体请求信息。参考http://cn2.php.net/manual/zh/function.curl-getinfo.php

## **五、使用curl发送post请求**

我们在前面说过，在向某个url发送get请求的话，没有必要使用cURL来发送get请求，可以使用比较便捷的file\_get\_contents函数来完成请求。但是，一般地，我们在提交某个表单的时候，数据是通过post请求的内容区域来提交的，而不是通过url参数来传递的, 这种情况下，我们应该使用灵活的cURL来模拟发送post请求。

现在，让我们使用cURL来模拟发送一个post请求到post.php脚本，提交几个数据到post.php，然后在post.php中输出post请求中的数据。示例代码如下:

$url = "http://www.52fhy.me/post.php";

$post\_data = array (

"blog\_name" => "52fhy",

"blog\_url" => "http://www.52fhy.com",

"action" => "Submit"

);

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL, $url);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

// 设置请求为post类型

curl\_setopt($ch, CURLOPT\_POST, 1);

// 添加post数据到请求中

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $post\_data);

// 执行post请求，获得回复

$response= curl\_exec($ch);

curl\_close($ch);

echo $response;

以上请求发送到post.php中后，通过print\_r($\_POST)输出后，以上示例代码会输出如下回复：

Array

(

[blog\_name] => 52fhy

[blog\_url] => http://www.52fhy.com

[action] => Submit

)

正如我们看到的，cURL成功发送post请求到post.php，提交了一些数据，并且收到了相应的来自post.php的回复，最后输出回复。上例虽然简单，但是充分演示了cURL发送post请求的便捷及强大之处，你可以在curl\_setopt上做文章。

## **六、文件上传**

下面来看下如果通过cURL发送post请求来实现文件上传。就拿深入浅出PHP下的文件上传中的文件上传例子来演示，在深入浅出php下的文件上传中，是通过表单的提交来实现文件上传的，那么通过cURL怎么来实现呢?

$url = "http://www.52fhy.me/upload.php";

$post\_data = array (

"attachment" => "@E:/jackblog/boy.jpg"

);

//初始化cURL会话

$ch = curl\_init();

//设置请求的url

curl\_setopt($ch, CURLOPT\_URL, $url);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

//设置为post请求类型

curl\_setopt($ch, CURLOPT\_POST, 1);

//设置具体的post数据

curl\_setopt($ch, CURLOPT\_POSTFIELDS, $post\_data);

$response = curl\_exec($ch);

curl\_close($ch);

print\_r($response);

通过以上示例代码，可以将我本地机器上的boy.jpg上传到本地服务器的upload.php中，如果在upload.php输出上传的具体信息的话，以上示例代码最后的输出的回复为：

Array

(

[attachment] => Array

(

[name] => boy.jpg

[type] => application/octet-stream

[tmp\_name] => D:\xampp\tmp\phpF27D.tmp

[error] => 0

[size] => 11490

)

)

由此可见，如果你要通过cURL来上传文件的话，只需要将上传的文件路径作为post数据设置到curl请求中，[并且在路径前面加上@符合](mailto:%E5%B9%B6%E4%B8%94%E5%9C%A8%E8%B7%AF%E5%BE%84%E5%89%8D%E9%9D%A2%E5%8A%A0%E4%B8%8A@%E7%AC%A6%E5%90%88)。

## **七、文件下载**

上述将了文件上传，同样的也可以使用curl来自动地完成文件的下载以及保存。有一点要补充下，在执行一个curl请求时，如果你需要获取返回的内容，而不是直接输出返回的内容的话，别忘记使用下面的代码设置，因为curl的默认是输出请求的回复内容:

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

假如在52fhy的服务器根目录下面有一个test.zip文件，我们需要将其下载下来，并且保存到本地文件中，就可以尝试使用下面的代码来实现：

//设置请求的下载文件的url

$url = 'http://www.52fhy.com/test.zip';

//保存到本地的文件路径

$path = 'local/path/to/test.zip';

//初始化请求，设置请求，获取回复,关闭会话

$ch = curl\_init($url);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, true);

$data = curl\_exec($ch);

curl\_close($ch);

//将文件内容写入本地文件

file\_put\_contents($path, $data);

注意：我以上省略了错误处理方面的代码，只是简单做个示例, 在实际中，你还需要通过curl\_getinfo函数来进行错误处理!

上述代码对于下载比较大型的文件是不适用的，因为需要先将文件读取到内存中，等所有内容都读取完毕，然后再写入到本地硬盘中。即使php中设置的 memory limit非常大，这种情况对性能的影响也是很大的。所以，我们对于大型文件的下载，应该让curl来接管这个任务，实现边下载，边写入的处理，这样的 话，就没什么问题了。请看下述代码:

$url = 'http://www.52fhy.com/test.zip';

$path = 'local/path/to/test.zip';

// 打开本地文件

$fp = fopen($path, 'w');

// 告诉curl本地文件句柄

$ch = curl\_init($url);

curl\_setopt($ch, CURLOPT\_FILE, $fp);

curl\_exec($ch);

curl\_close($ch);

fclose($fp);

在上述代码中，我们先打开个本地文件，并将文件句柄设置到curl中，然后让curl一边读取远程数据，一边写入到本地文件中。因为我们不需要在程序中获取远程回复的内容了，所以只要执行请求就可以。

## **八、http 验证**

如果服务器端需要验证请求，可以通过类似一下示例代码来实现:

$url = "http://www.52fhy.com/users/";

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL, $url);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

// 设置用户名以及密码

curl\_setopt($ch, CURLOPT\_USERPWD, "username:password");

// 设置重导向

curl\_setopt($ch, CURLOPT\_FOLLOWLOCATION, 1);

curl\_setopt($ch, CURLOPT\_UNRESTRICTED\_AUTH, 1);

$response = curl\_exec($ch);

curl\_close($ch);

## **九、通过代理发送请求**

cURL还可以通过代理服务器来向发送请求，请看一下示例代码:

$ch = curl\_init();

curl\_setopt($ch, CURLOPT\_URL,'http://www.52fhy.com');

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

// 设置代理ip地址

curl\_setopt($ch, CURLOPT\_PROXY, '222.73.173.50:8080');

// 要验证的话，这里设置用户名以及密码

curl\_setopt($ch, CURLOPT\_PROXYUSERPWD,'username:password');

$response = curl\_exec($ch);

curl\_close ($ch);

## **十、发送json数据**

最后，我们来看下通过cURL来想服务器端发送json数据。具体的代码如下：

$url = 'http://www.52fhy.me/json.php';

// 建立json字符串

$data = array('site' => '52fhy', 'url' => 'http://www.52fhy.com','email'=>'52fhy@gmail.com');

$json\_string = json\_encode($data);

$ch=curl\_init($url);

curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1);

// 通过post请求发送上述json字符串

curl\_setopt($ch, CURLOPT\_CUSTOMREQUEST, "POST");

curl\_setopt($ch, CURLOPT\_POSTFIELDS, array('data'=>$json\_string));

$response = curl\_exec($ch);

curl\_close($ch);

echo $response;

大家可以看到，上述请求是发送到我的本地服务器的json.php下，我在该文件中使用json\_decode来将接受到的json字符串转换为对象，然后输出其中的email字段，代码如下:

$json\_data = json\_decode($\_POST['data']);

echo $json\_data->email;

在上述代码中接受的json字符串为:

'{"site":"52fhy","url":"http:\/\/www.52fhy.com","email":"52fhy@gmail.com"}'

经过json\_decode以后，就转换为php中的数据格式，成为了一个对象，所以可以通过$json\_data->email来访问其中email字段的值，[最后也就是输出52fhy@gmail.com](mailto:%E6%9C%80%E5%90%8E%E4%B9%9F%E5%B0%B1%E6%98%AF%E8%BE%93%E5%87%BA52fhy@gmail.com)。你可以使用上述代码测试一下。

如果通过以下php数组生成json字符串的话:

$data = array('52fhy', 'http://www.52fhy.com', '52fhy@gmail.com');

所生成的json字符串如下：

'["52fhy","http:\/\/www.52fhy.com","52fhy@gmail.com"]'

上述json字符串在经过json\_decode处理后，就会变成php中的数组格式，如果要获取email的话，就可以通过$json\_data[2]来访问。

## **十一、cURL批处理（multi cURL）**

cURL还有一个高级特性——批处理句柄（handle）。这一特性允许你同时或异步地打开多个URL连接。

下面是来自来自php.net的示例代码：

// 创建两个cURL资源

$ch1 = curl\_init();

$ch2 = curl\_init();

// 指定URL和适当的参数

curl\_setopt($ch1, CURLOPT\_URL, "http://lxr.php.net/");

curl\_setopt($ch1, CURLOPT\_HEADER, 0);

curl\_setopt($ch2, CURLOPT\_URL, "http://www.php.net/");

curl\_setopt($ch2, CURLOPT\_HEADER, 0);

// 创建cURL批处理句柄

$mh = curl\_multi\_init();

// 加上前面两个资源句柄

curl\_multi\_add\_handle($mh,$ch1);

curl\_multi\_add\_handle($mh,$ch2);

// 预定义一个状态变量

$active = null;

// 执行批处理

do {

$mrc = curl\_multi\_exec($mh, $active);

} while ($mrc == CURLM\_CALL\_MULTI\_PERFORM);

while ($active && $mrc == CURLM\_OK) {

if (curl\_multi\_select($mh) != -1) {

do {

$mrc = curl\_multi\_exec($mh, $active);

} while ($mrc == CURLM\_CALL\_MULTI\_PERFORM);

}

}

// 关闭各个句柄

curl\_multi\_remove\_handle($mh, $ch1);

curl\_multi\_remove\_handle($mh, $ch2);

curl\_multi\_close($mh);

这里要做的就是打开多个cURL句柄并指派给一个批处理句柄。然后你就只需在一个while循环里等它执行完毕。

这个示例中有两个主要循环。第一个 do-while 循环重复调用 curl\_multi\_exec() 。这个函数是无隔断（non-blocking）的，但会尽可能少地执行。它返回一个状态值，只要这个值等于常量 CURLM\_CALL\_MULTI\_PERFORM ，就代表还有一些刻不容缓的工作要做（例如，把对应URL的http头信息发送出去）。也就是说，我们需要不断调用该函数，直到返回值发生改变。

而接下来的 while 循环，只在 $active 变量为 true 时继续。这一变量之前作为第二个参数传给了 curl\_multi\_exec() ，代表只要批处理句柄中是否还有活动连接。接着，我们调用 curl\_multi\_select() ，在活动连接（例如接受服务器响应）出现之前，它都是被“屏蔽”的。这个函数成功执行后，我们又会进入另一个 do-while 循环，继续下一条URL。