# 二次开发随手记

# 我在哪里可以找到Piwik跟踪代码？

使用本页面描述的所有功能，您需要使用最新版本的跟踪代码。为你的网站上找到的跟踪代码，请按照下面的步骤：

登录到您的管理员或超级用户使用piwik帐户

点击设置访问管理区

点击网站列出的网站，你跟踪Piwik

你希望跟踪网站上点击查看跟踪代码

现在你可以复制和粘贴的JavaScript跟踪代码到你的网页，就在< / <body>标签。

使用piwik的跟踪代码如下：

*<!-- Piwik -->*

<script type="text/javascript">

**var** \_paq = \_paq || [];

(**function**(){ **var** u=(("https:" == document.location.protocol) ? "https://{$PIWIK\_URL}/" : "http://{$PIWIK\_URL}/");

\_paq.push(['setSiteId', {$IDSITE}]);

\_paq.push(['setTrackerUrl', u+'piwik.php']);

\_paq.push(['trackPageView']);

\_paq.push(['enableLinkTracking']);

**var** d=document, g=d.createElement('script'), s=d.getElementsByTagName('script')[0]; g.type='text/javascript'; g.defer=true; g.async=true; g.src=u+'piwik.js';

s.parentNode.insertBefore(g,s); })();

</script>

<!-- End Piwik Code -->

在你使用piwik的跟踪代码,{$piwik\_url }取代piwik url， { $ idsite }取代将在你跟踪Piwik网站idsite取代。

此代码可以看看那些你熟悉JavaScript有点奇怪，但那是因为它是由异步运行。换句话说，浏览器不会等待piwik.js下载文件来显示你的网页。

异步跟踪，配置和跟踪返回由全局数组

\_paq执行的，独立的piwik.js异步加载。格式：

\_paq**.push**([ 'API\_method\_name', parameter\_list ]);

你也可以执行推送函数。比如说呢：

**var** visitor\_id; \_paq.push([ **function**() { visitor\_id = **this**.getVisitorId(); }]);

或例如，拿一个自定义变量 variable (name,value) 使用异步代码：

\_paq.push(['setCustomVariable','1','VisitorType','Member']); \_paq.push([ **function**() { **var** customVariable = **this**.getCustomVariable(1); }]);

你可以推送\_paq数组甚至在piwik.js文件被加载并运行。

如果你使用piwik的跟踪代码看起来不喜欢这一个，你可以使用过时的版本。旧版本仍然正常工作，将跟踪您的访客，但我们强烈建议您更新您的网页，使用最新的跟踪代码。

# JavaScript的追踪功能,自定义显示在使用piwik网页名称

默认情况下，使用piwik使用当前页面的URL在piwik界面页面标题。如果你的URL是不简单的，或者如果你想自定义方式Piwik跟踪你的页面，你可以指定页面的标题中使用JavaScript代码。

一个常见的用途是设置HTML标题值作为文档的标题：

[...]

\_paq**.push**(['setDocumentTitle', document.title]);

\_paq**.push**(['trackPageView']);

[...]

如果你跟踪多个子域，在同一个网站，你可能想让你的网页标题作为前缀的子域名很容易让你看到交通数据为每个子域。你可以在做这么简单：

[...]

\_paq**.push**(['setDocumentTitle', document.domain + "/" + document.title]);

\_paq**.push**(['trackPageView']);

[...]

高级用户还可以动态生成页面的名称，例如，PHP：

[...]

\_paq.push(['setDocumentTitle', "**<?php echo $myPageTitle ?>**"]);

\_paq.push(['trackPageView']);

[...]

# 点击手动触发事件

默认情况下，页面视图，当JavaScript Piwik轨迹跟踪代码加载并执行在每个页面视图。然而，在现代的网站或Web应用程序的用户交互，并不涉及加载新页。例如，当用户点击一个链接JavaScript，或者当他们点击一个标签（这将触发一个js事件），或当他们与用户的交互界面元素，你仍然可以使用piwik追踪这些相互作用。

与piwik跟踪任何用户交互或点击，您可以手动调用JavaScript函数

trackEvent(category, action, [name], [value])

。例如，如果你想跟踪一个JavaScript菜单上单击，你可以写：

[...]

<a href="#" onclick="javascript:\_paq.push(['trackEvent', 'Menu', 'Freedom']);">Freedom page</a>

[...]

# 手动触发一个目标转换

默认情况下，目标在Piwik被定义为“匹配”的URL部分（开始，包含，或正则表达式匹配）。你还可以跟踪给定目标页面，下载，或网页链接点击。

在某些情况下，你可能想注册在其他类型的动作转换，例如：

当用户提交表单

当用户在超过一个给定的时间在网页上

当一个用户做一些互动的Flash应用程序

当用户提交了他的车并做了付款：你可以使用piwik的跟踪代码来支付网站将在你使用piwik数据库记录转换，在转换的制定的收益

触发一个目标使用Piwik JavaScript跟踪，你可以简单的做：

[...]

// logs a conversion **for** goal 1

\_paq.push(['trackGoal', 1]);

[...]

你也可以注册为这个目标转换一个自定义的收入。例如，您可以生成调用trackgoal动态确定交易收入：

[...]

*/\* logs a conversion for goal 1 with the custom revenue set \*/*

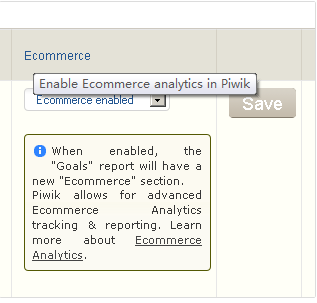
\_paq.push(['trackGoal', 1, **<?php** **echo** $cart->getCartValue(); **?>**]);

[...]

# 跟踪电子商务订单，车的更新和产品/类别页面视图

## 启用电子商务跟踪

电子商务报告，默认是禁用的。为了使电子商务分析报告，点击设置>网页>编辑”并选择启用电子商务电子商务中列的下拉菜单。

****

电子商务是一个网站一旦启用，目标菜单将被重新命名为电子商务与目标并将显示电子商务报告。

****

如果你没有看到目标菜单，这可能是由于目标的插件被禁用。

去tosettings和pluginsto使目标插件。

电子商务跟踪

一旦您启用跟踪电子商务网站，你需要实现你网站上的各种电子商务互动跟踪。有三种主要的电子商务互动，跟踪与piwik：

电子商务订单（和产品），

电子商务车升级（产品），

产品（或类）的网页浏览。

以下各节说明如何设置电子商务跟踪，利用JavaScript跟踪电子商务跟踪方法

## 跟踪电子商务订单和购买的物品（需要）

跟踪电子商务订单和产品的详细信息将使piwik处理报告：

产品规格，

品名。

顶级产品类别在你的网上商店的电子商务性能概述。

电子商务订单跟踪，使用两个JavaScript函数：

**addEcommerceItem(**[**productSKU**](http://piwik.org/?page_id=729#faq_124)**, productName, productCategory, price, quantity)** 这增加了一个产品的订单，并必须被调用的顺序为每个产品。productsku是一个必需的参数，它也建议，如果他们在你的电子商务商店可送你其他参数。

**trackEcommerceOrder(orderId, grandTotal, subTotal, tax, shipping, discount)** 这一跟踪电子商务订单和数据发送到您的Piwik服务器，这两个订单和产品以前添加的。只有OrderID和grandTotal（即收入）是必需的。

要跟踪包含两个产品–这个代码的电子商务订单应插在“订单”网页的一个例子，一旦客人退房并颁发的付款。如果你正确安装使用piwik，你应该已经有了一个跟踪代码页。在电子商务的代码必须添加基于订单生成：

[...]

// add the first product to the order

\_paq.push(['addEcommerceItem',

"9780786706211", // (required) SKU: Product unique identifier

"Endurance: Shackleton's Incredible Voyage", // (optional) Product name

"Adventure Books", // (optional) Product 类型. You can also specify an array of up to 5 categories eg. ["Books", "New releases", "Biography"]

8.8, // (recommended) Product price

1 // (optional, default to 1) Product 数量

]);

// Here you can add other products in the order

[...]

// Specifiy the order details to Piwik server &amp; sends the data to Piwik server

\_paq.push(['trackEcommerceOrder',

"A10000123", // (required) Unique Order ID

35, // (required) 订单收入总额（包括税收，航运，和减去折扣）

30, // (optional) 订单小计（不包括运费）

5.5, // (optional) 税收

4.5, // (optional) 运输费

false // (optional)折扣 (set to false for 未定义)

]);

// 我们建议离开的订单确认页面时候调用trackPageView（）

\_paq.push(['trackPageView']);

[...]

## 关于函数的参数的重要事项：

price (parameter of addEcommerceItem() function), grandTotal, subTotal, tax, shipping, discount (parameters of trackEcommerceOrder() function)

必须提供为整数或浮点数，而不是字符串。例如，以下所有的值是无效的：“14.4美元”，“eur14.4″，4″”，“14.4″。下面的值是有效的：5，3.3，5.44。如果你的电子商务软件提供的值的字符串，你可以确定后的字符串不包含例如货币符号或字符，调用JavaScript函数parsefloat()：parseFloat（“554.20″）。请注意，不支持JavaScriptparsefloat()逗号分隔的十进制值”25,3″所以你可能需要替换逗号用点再打parsefloat()。

**orderId**是一个独特的字符串的顺序标识符。如果包含trackecommerceorder()调用网页刷新由客户几个小时或几天的初始订单后，piwik将自动丢弃的顺序记录。您可以为不同的网站使用相同ID的订单号，但对于一个给定的网站，订单号必须是唯一的。

如果你打印出这些参数中的HTML代码，确保产品的型号，名称和类别带双引号，否则会创建JavaScript语法错误。

## 跟踪添加到购物车和项目添加到购物车（可选）

每次有客人添加（或更新，删除）从他/她的车的一个项目，你可以使用Piwik跟踪最新的车小计以及在车左产品。跟踪车更新将允许你看到多少收入留在你的访客的车每一天/周/等，以及报告哪些产品是最经常在车左，这类产品往往是遗弃在车。另外，你可以想像每个访问，他们被遗弃在访客记录购物车的内容。

对于放弃一些常见的原因是：

航运价格高，

为新用户复杂的形式（“匿名结帐”选项缺乏），

复杂冗长的结帐过程，

慢的网站，

游客只是浏览网站或想总成本与竞争对手的比较。

当你改变你的网站以较低的放弃率，并增加你的收入，你可以使用Piwik理解如果你的变化作出积极的影响。

与piwik跟踪购物车，你会调用JavaScript函数trackecommercecartupdate()每次用户添加，更新或从购物车中删除一个产品。

示例包含两个产品的电子商务订单跟踪：

该代码是添加的每一次“购物车”的页面被加载，或当“加入购物车”按钮点击：

[...]

// add the first product to the order

\_paq.push(['addEcommerceItem',

"9780786706211", // (required) SKU: Product unique identifier

"Endurance: Shackleton's Incredible Voyage", // (optional) Product name

["Adventure Books", "Best sellers"], // (optional) Product category, string or array of up to 5 categories

8.8, // (recommended) Product price

1 // (optional, default to 1) Product quantity

]);

// Here it is important to add all other products found in the cart, even the products not updated by the current "Add to cart" click

[...]

// Records the cart for this visit

\_paq.push(['trackEcommerceCartUpdate',

15.5]); // (recommended) Cart amount

\_paq.push(['trackPageView']);

[...]

## 跟踪产品的网页和类别页面视图（可选）

跟踪产品的网页浏览，和/或类别页面视图让Piwik过程的转化率为每一个产品和/或类别。

它是有用的知道哪些产品或者类别网页把访问者变成你的客户。例如，也许有些产品页面有低于平均水平的转化率，但产生高收益。你可能会提高这些产品的页面，并随着时间的推移，检查他们的转换率。

跟踪产品或类别页面视图，函数setecommerceview（productsku，ProductName，categoryName, price）时，就在调用piwiktracker。对产品/类别页面trackpageview()。

例如跟踪产品页面视图

在这个例子中，下面的代码添加到产品页面的书“Endurance ”category属于“Adventure Books”：

[...]

// all parameters are optional, but we recommend to set at minimum productSKU and productName

\_paq.push(['setEcommerceView',

"9780786706211", // (required) SKU: Product unique identifier

"Endurance: Shackleton's Incredible Voyage", // (optional) Product name

"Adventure Books", // (optional) Product category, or array of up to 5 categories

20.11 // (optional) Product Price as displayed on the page

]);

\_paq.push(['trackPageView']);

[...]

例如跟踪类别页面的视图

This code is added on the Category page “Adventure Books”

[...]

// on a category page, productSKU and productName are not applicable and are set to false

\_paq.push(['setEcommerceView',

productSku = false, // No product on Category page

productName = false, // No product on Category page

category = "Adventure Books" // Category Page, or array of up to 5 categories

]);

\_paq.push(['trackPageView']);

[...]

## 在使用piwik电子商务报告

在你使用piwik的界面，你可以找到电子商务报告通过点击主菜单的“电子商务与目标”。

假设你已经设置电子商务跟踪如上所述，piwik将提供以下电子商务报告

电子商务数据

电子商务订单

总收入

平均订单价值

电子商务的转化率

采购产品

弃车指标：废弃的车去，在车左收入

电子商务报告

电子商务概述：收入，订单，转化率，AOV（平均订单价值），等。

最好的productskus

最好的产品名称

最好的产品类别

电子商务的日志–显示所有包含一个电子商务订单的访问，所以你可以想象你的客户clickpath

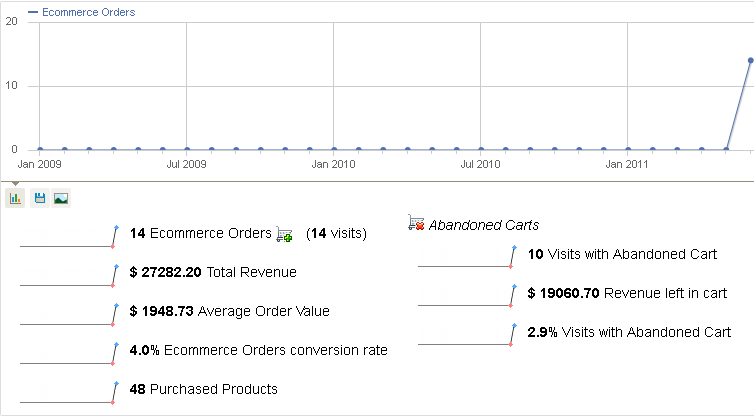
改进的标准报告

现有的报告将包括在“收入”栏的“电子商务收入”处理。

这些标准的报告也将有一个新的度量列：“电子商务的转化率”

电子商务活动概览

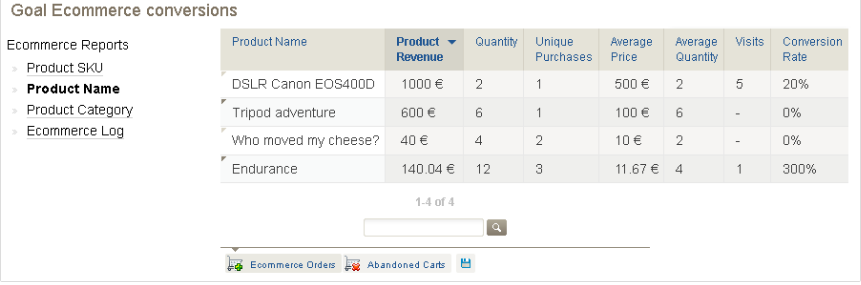
默认情况下，piwik将显示电子商务系统for the selected日期范围，as well as a图显示历史收益for the selected度量。



最好的产品、最好的类别

Piwik过程上销售的产品列表，任何的产品度量下令：收入，购买量，独特的购买，平均价格，平均数量，访问或转化率。

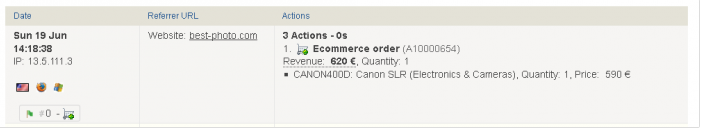
你可以通过产品SKU的顶级产品报告，按产品名称或类别。



请注意，您可以通过在每一列的列名称访问帮助定义。

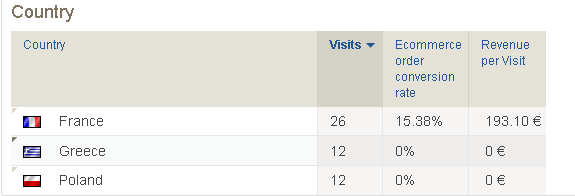
电子商务活动的日志

与piwik，您可以查看（实时！）详细的电子商务的日志，显示所有的访问使在你的网站上购买，以及在他们离开所有的游客都有产品废弃的购物车。



电子商务的度量标准的报告

现有的报告现在将显示在列列表的电子商务数据，使电子商务表现可以显示在任何现有的报告不仅在电子商务报告页。



先进的电子商务统计API

所有数据在用户界面的电子商务报告，也可以通过使用piwik分析API。

你可以很容易地在你的网站上购买的产品要求的列表，要么按产品型号，产品名称或产品类别。

每个条目返回以下指标：

Total revenue

Total quantity

Average price

Average quantity

Number of orders (or abandoned carts) containing this product

number of visits on the Product page

Conversion rate

总收入

总数量

平均价格

平均数量

订单数量（或废弃的车）包含该产品

在产品页面访问数

转化率

你可以要求这些报告的电子商务订单，也为废弃的车。

查看完整的文档和示例的链接：电子商务分析API参考

更多的信息

我们希望您喜欢使用Piwik跟踪你的电子商务商店（S），和电子商务分析可以帮助你从你的网站获得更多的收入，以及让你的客户满意。

对于列表中的功能要求，围绕电子商务跟踪和报告在使用piwik，看看票的电子商务分析功能要求。

如果您对本文档的任何意见或问题，请留言在论坛或在下面的反馈形式。

# 跟踪内部搜索的关键字，类别，和没有结果的搜索关键词

Piwik提供先进的网站搜索分析功能，让您跟踪你的访客如何使用你的网站内部的搜索引擎。默认情况下，使用piwik可以读取URL参数将包含搜索关键词。然而，你也可以记录网站搜索关键词手动使用

Javascript function

trackSiteSearch(...)

在你的网站上，在标准的网页，你通常会有一个调用来记录页面视图通过piwikTracker.trackPageView()

在你的搜索结果页面，你会叫相反

piwikTracker.trackSiteSearch(keyword, category, searchCount)

记录内部搜索请求。注：“关键词”参数是必需的，但类和searchcount是可选的

[...]

\_paq.push(['trackSiteSearch',

// Search keyword searched for

"Banana",

// Search category selected in your search engine. If you **do** **not** need this, **set** **to** **false**

"Organic Food",

// **Number** **of** results **on** the Search results page. Zero indicates a 'No Result Search Keyword'. **Set** **to** **false** **if** you don't know

0

]);

*// We recommend not to call trackPageView() on the Site Search Result page*

*// \_paq.push(['trackPageView']);*

[...]

我们也强烈建议设置searchcount参数，如piwik将特别报告“没有结果的关键字，如“关键字进行搜索，但没有返回任何结果。它通常是了解用户搜索很有趣但没找到（吗？）在你的网站。在用户DOC了解网站搜索分析。

# 自定义变量

自定义变量是一个强大的功能，使您能够跟踪每一个访问自定义值，和/或每个页面视图。请参阅跟踪自定义变量文档页面的一般信息。

您可以联系到5自定义变量（名称和值）为每个访问你的网站，和/或多达5个自定义变量为每个页面视图。如果你设置自定义变量访问，当他回来的时候，1小时或2天之后，这将是一个新的访问和他/她的自定义变量将是空的。

有两个“范围”，您可以设定您的自定义变量。“范围”是函数的第四个参数

setCustomVariable()

当scope = "visit" 范围=“参观”，自定义变量的名称和值将被存储在数据库的访问。你可以存储多达5个自定义变量的范围”，每一个访问”。

当 scope = "page" 范围=“页面”，自定义变量的名称和值将被存储为页面视图被跟踪。你可以存储多达5个自定义变量的范围”，为每个页面视图页面”。

自定义变量统计报告Piwik访问者>自定义变量下。自定义变量的范围“访问”和“页面”汇总报告。

设置一个访问自定义变量

**setCustomVariable (index, name, value, scope = "visit")**

这个函数是用来创建，或更新一个自定义变量的名称和值。例如，假设你要存储在每个访问用户的性别。你将存储自定义变量name = "gender", value = "male" or "female".

重要：一个给定的自定义变量名必须存储在相同的“指数”。例如，如果您选择存储变量 **name = "Gender"** in **index = 1**  and you record another custom variable in index = 1

then the "Gender" variable将被删除并替换为新的自定义变量存储在index 1。

[...]

\_paq.push(['setCustomVariable',

*// Index, the number from 1 to 5 where this custom variable name is stored*

1,

*// Name, the name of the variable, for example: Gender, VisitorType*

"Gender",

*// Value, for example: "Male", "Female" or "new", "engaged", "customer"*

"Male",

*// Scope of the custom variable, "visit" means the custom variable applies to the current visit*

"visit"

]);

\_paq.push(['trackPageView']);

[...]

你只需要用 scope "visit" 范围“参观”一旦设置一个变量值，并将记录整个访问。

设置一个页面视图自定义变量

**setCustomVariable (index, name, value, scope = "page")**

as well as跟踪自定义变量“参观”，它是有时useful to track自定义变量for each页面视图分别。For example，for a“新闻”网站或博客，categorized into one may be given条或几个类别。在这种情况下，你可以设置一个或几个custom variables with

name="category"

, one with

value="Sports"

 and another with

value="Europe"

 如果文章是在体育和欧洲类别分类。自定义变量的报告将有多少访问量和页面浏览量在每个网站的类别报告。这个信息很难获得与标准Piwik报告因为他们报告的“最佳网页URL”和“最佳网页标题”这可能不包含“类”的信息。

[...]

*// Track 2 custom variables with the same name, but in different slots.*

*// You will then access the statistics about your articles' categories in the 'Visitors > custom variables' report*

\_paq.push(['setCustomVariable', 1, 'Category', 'Sports', 'page']);

*// Track the same name but in a different Index*

\_paq.push(['setCustomVariable', 2, 'Category', 'Europe', 'page']);

*// Here you could track other custom variables with scope "page" in Index 3, 4 or 5*

*// The order is important: first setCustomVariable is called and then trackPageView records the request*

\_paq.push(['trackPageView']);

[...]

重要：它可以存储自定义变量的范围“访问”中的“指数”1，和存储一个不同的自定义变量范围的“页面”的“指数”1。因此，在技术上你可以跟踪你网站的每个网页最多10个自定义变量的名称和值（5个“网页”范围存储在实际的网页浏览，5个“参观”范围存储在访问）。

[...]

\_paq.push(['setCustomVariable',

*// Index, the number from 1 to 5 where this custom variable name is stored for the current page view*

1,

*// Name, the name of the variable, for example: Category, Sub-category, UserType*

"category",

*// Value, for example: "Sports", "News", "World", "Business", etc.*

"Sports",

*// Scope of the custom variable, "page" means the custom variable applies to the current page view*

"page"

]);

\_paq.push(['trackPageView']);

[...]

删除自定义变量

deleteCustomVariable (index, scope )

如果你创建了一个自定义变量，然后决定从一个访问或页面视图中删除该变量，你可以用deletecustomvariable。

坚持使用piwik服务器的更改，您必须在调用该函数  
trackPageView();

[...]\_paq.push(['deleteCustomVariable', 1, "visit"]); // **Delete** the variable **in** index 1 stored **for** the **current** visit

\_paq.push(['trackPageView']);

[...]

取名字和一个自定义变量的值

getCustomVariable (index, scope )

此功能是有用的如果范围=“访问”scope = "visit".。

在这种情况下，自定义变量记录的访问时间第一方Cookie（上一次的操作，30分钟后）。你可以检索自定义变量的名称和值使用piwiktracker.getcustomvariable。如果在请求的索引没有自定义变量，它将返回false。

[...]\_paq.push([ **function**() {

**var** customVariable = **this**.getCustomVariable( 1, "visit" );

*// Returns the custom variable: [ "gender", "male" ]*

*// do something with customVariable...*

}]);

\_paq.push(['trackPageView']);

[...]

# 对于域和子域Cookie设置

Piwik使用第一方Cookie来记录一些信息（访问次数，原来的引荐，和独特的访客ID）。第一[方Cookie保证较高的用户隐私（因为Cookie不发送给第三方服务器），并因此设置在大多数浏览器。

Piwik创建一组的每个域和子域的饼干。如果你想跟踪一些子域和分享准确统计同样的饼干，有必要定制Piwik跟踪代码。检查出的各种配置的例子。

# 如果你只跟踪一个域名或域中的一个单一的网站使用piwik

这是标准的使用案例。Piwik跟踪一个域名访问没有子域，在一个单一的Piwik网站。

[...]

*// Default Tracking code*

\_paq.push(['setSiteId', 1]);

\_paq.push(['setTrackerUrl', u+'piwik.php']);

\_paq.push(['trackPageView']);

[...]

# 如果你跟踪一个域名和若干子域在使用piwik一个网站

如果你想记录访问主域名以及其子，你想在所有领域分享饼干。您可以通过调用

setCookieDomain()

在所有子域跟踪代码。

[...]

\_paq.push(['setSiteId', 1]);

\_paq.push(['setTrackerUrl', u+'piwik.php']);

*// Same cookie as: example.com, www.example.com, subdomain.example.com, ...*

\_paq.push(['setCookieDomain', '\*.example.com']);

\_paq.push(['setDomains', '\*.example.com']); *// Download & Click tracking alias domains*

\_paq.push(['trackPageView']);

[...]

# 如果你跟踪域子目录，或在不同的网站页面Piwik

默认情况下，使用piwik只使用一个cookie的域名，和所有的网页和子目录。

有可能的情况下，你跟踪一个子目录作为一个单独的网站在使用piwik。如果访问者访问超过几个子目录，这将导致在您的报告不准确：一些网站上的时间，访问次数，转换引用，返回新的游客。在这个用例，你可以确保你的报告通过创建每个子路径，你在不同的网站跟踪不同的饼干留在Piwik准确。功能

setcookiepath()

用于设置cookie路径。

例如，如果你的网站的用户配置文件，你可以跟踪每个用户的个人资料页的分析作为一个独特的网站使用piwik。在主域的主页，你会使用默认跟踪代码。

[...]

// idSite = X **for** the Homepage

\_paq.push(['setSiteId', X]);

\_paq.push(['setTrackerUrl', u+'piwik.php']);

\_paq.push(['trackPageView']);

[...]

在/用户/ myusername页，你会写：

[...]

*// The idSite Y will be different from other user pages*

\_paq.push(['setSiteId', Y]);

\_paq.push(['setTrackerUrl', u+'piwik.php']);

\_paq.push(['setCookiePath', '/user/MyUsername']);

\_paq.push(['trackPageView']);

[...]

有关跟踪Piwik网站和子域的更多信息，参见常见问题：如何配置Piwik监测的几个网站，域和子域

# 忽略特定域或子域中的“网页链接点击跟踪

默认情况下所有链接到其他比目前的域启用点击跟踪领域，每一个点击的将被视为一个网页链接。如果您使用多个域和子域，您可能会看到在您的区域出现在页>点击类型报告。

如果你只需要点击到外部网站的出现在你的自制的报告，你可以使用这个功能

setDomains()指定的别名域或子域列表。通配符（\*域。例如。org）的支持，让你轻易忽略点击所有子域。

[...]

*// Don't track Outlinks on all clicks pointing to \*.hostname1.com or \*.hostname2.com*

*// Note: the currently tracked website is added to this array automatically*

\_paq(['setDomains', ["\*.hostname1.com", "hostname2.com"]]);

\_paq.push(['trackPageView']);

[...]

# 禁用下载网页链接跟踪

默认情况下，使用piwik的跟踪代码可以点击下载跟踪。禁用所有自动下载和网页链接跟踪，您必须删除通话的

enablelinktracking()

功能：

[...]

*// we comment out the function that enables link tracking*

*// \_paq.push(['enableLinkTracking']);*

\_paq.push(['trackPageView']);

[...]

# 禁用下载网页链接特定的CSS类跟踪

您可以禁用自动下载和网页链接与特定的CSS类链接跟踪：

[...]

*// you can also pass an array of strings*

\_paq.push(['setIgnoreClasses', "no-tracking"]);

\_paq.push(['trackPageView']);

[...]

This will result in clicks on a link

<a href='http://example.com' **class='no-tracking'>Test</a>**

 not being counted.

# 禁用下载或网页链接跟踪在一个特定的链接

如果你想一直忽略下载或网页链接跟踪在一个特定的链接，

you can add the 'piwik\_ignore' css class to it:

<a href='http://builds.piwik.org/latest.zip' **class='piwik\_ignore'>File I don't want to track as a download</a>**

# 强制一个链接被记录在使用piwik下载点击

如果你想使用piwik考虑给定的链接，下载，you can add the 'piwik\_download' css class to the link:

<a href='last.php' **class='piwik\_download'>Link I want to track as a download</a>**

注意：您可以自定义重命名使用强制点击被记录为一个下载CSS类：

[...]

*// now all clicks on links with the css class "download" will be counted as downloads*

*// you can also pass an array of strings*

\_paq.push(['setDownloadClasses', "download"]);

\_paq.push(['trackPageView']);

[...]

# 一个链接被记录为一个网页链接点击

如果你想使用piwik考虑给定的链接的网页链接（在当前域或一个别名域的链接），你可以添加“piwik\_link的CSS类的链接：

<a href='http://mysite.com/partner/' **class='piwik\_link'>Link I want to track as an outlink</a>**

注意：您可以自定义重命名使用强制点击被记录为一个网页链接的CSS类：

[...]

*// now all clicks on links with the css class "external" will be counted as outlinks*

*// you can also pass an array of strings*

\_paq.push(['setLinkClasses', "external"]);

\_paq.push(['trackPageView']);

[...]

或者，您可以使用JavaScript手动触发一个网页链接点击（它将页的意见或下载文件相同的工作）。在这个例子中，自定义网页链接被触发时，电子邮件地址，点击：

<a href="mailto:namexyz@mydomain.co.uk" target="\_blank" onClick="javascript:\_paq.push(['trackLink', 'http://mydomain.co.uk/mailto/Agent namexyz', 'link']);">namexyz@mydomain.co.uk </a>

# 改变暂停计时器

当用户点击下载一个文件，或单击一个出站链接，piwik记录。为了这样做，它增加了在用户被重定向到所请求的文件或链接，一个小的延迟。默认值是500ms，但是你可以设置它为一个短的时间。值得注意的是，然而，这样做的风险结果，这段时间不足够长的数据被记录在使用piwik。

[...]

\_paq.push(['setLinkTrackingTimer', 250]); *// 250 milliseconds*

\_paq.push(['trackPageView']);

[...]

# 更改文件扩展名列表来跟踪“Downloads”

默认情况下，任何文件结束的一个扩展，将被视为在piwik界面的'下载'：

7z|aac|arc|arj|apk|asf|asx|avi|bin|bz|bz2|csv|deb|dmg|doc|

exe|flv|gif|gz|gzip|hqx|jar|jpg|jpeg|js|mp2|mp3|mp4|mpg|

mpeg|mov|movie|msi|msp|odb|odf|odg|odp|ods|odt|ogg|ogv|

pdf|phps|png|ppt|qt|qtm|ra|ram|rar|rpm|sea|sit|tar|

tbz|tbz2|tgz|torrent|txt|wav|wma|wmv|wpd||xls|xml|z|zip

代替扩展你想跟踪文件的下载列表，您可以使用

setDownloadExtensions( string )

: [...]

*// we now only track clicks on images*

\_paq.push(['setDownloadExtensions', "jpg|png|gif"]);

\_paq.push(['trackPageView']);

[...]

如果你想跟踪一个新的文件，你可以将其添加到列表中使用

addDownloadExtensions( filetype )

[...]

*// clicks on URLs finishing by mp5 or mp6 will be counted as downloads*

\_paq.push(['addDownloadExtensions', "mp5|mp6"]);

\_paq.push(['trackPageView']);

[...]

# 多Piwik跟踪器

可以使用多个Piwik跟踪指向相同的或不同的Piwik服务器跟踪页。为了提高页面的加载时间，你可以加载piwik.js一旦。每次调用

Piwik.getTracker()

返回一个独特的Piwik跟踪对象（实例）可以配置。

script type="text/javascript">

window.piwikAsyncInit = **function** () {

**try** {

**var** piwikTracker = Piwik.getTracker("http://URL\_1/piwik.php", 1);

piwikTracker.trackPageView();

**var** piwik2 = Piwik.getTracker("http://URL\_2/piwik.php", 4);

piwik2.trackPageView();

} **catch**( err ) {}

};

</script>

piwikAsyncInit

方法引入Piwik 2.3将一旦Piwik跟踪加载和初始化执行。在早期版本中你必须加载Piwik同步。

请注意，您还可以设置网站ID和Piwik tracker url手动，而不是设置在gettracker呼叫：

// we replace Piwik.getTracker("http://example.com/piwik/", 12)

var piwikTracker = Piwik.getTracker();

piwikTracker.**set**SiteId( 12 );

piwikTracker.**set**TrackerUrl( "http://example.com/piwik/" );

piwikTracker.trackPageView();

# 所有的方法在跟踪API可用列表

## 从piwik类请求跟踪器实例

Piwik.getTracker( trackerUrl, siteId ) 获取跟踪一个新的实例

Piwik.getAsyncTracker()获取用于异步跟踪跟踪器的内部审

## 使用跟踪对象

enableLinkTracking( enable ) 安装连接跟踪所有适用的链接元素。设置启用参数来实现采用伪点击事件跟踪浏览器（如Firefox）不产生鼠标中键单击事件。默认是“真正的”鼠标单击事件的处理。

addListener( element ) 添加点击侦听特定链接元素。当点击，piwik将日志自动点击。

setRequestMethod( method ) 设置请求的方法是 "GET" or "POST"。（默认是"GET".。）使用POST请求的方法，使用piwik主机必须跟踪网站的主机相同（Piwik安装在同一域名作为您的跟踪网站）。

trackEvent(category, action, [name], [value]) 记录与事件类的事件（视频，音乐，游戏……），一个事件的行动（播放，暂停，持续时间，添加播放列表，点击下载，……），和一个可选的事件的名称和可选的数值。

trackPageView([customTitle]) 记录访问这个页面

trackSiteSearch(keyword, [category], [resultsCount]) 记录特定关键字的网站内部的搜索，在一个可选的类型，指定在搜索结果页的可选的计数。

trackGoal( idGoal, [customRevenue]) 手动记录为目标idgoal转换，通过自定义的收入customrevenue如果指定

trackLink( url, linkType ) 手动记录从您自己的代码点击。URL是完整的URL将被作为一个点击跟踪。链条热水可以“链接”的网页链接或“下载”，下载。

## 跟踪对象的配置

setDocumentTitle( string ) 重写document.title

setDomains( **array**) 将被视为本地主机或域阵列。通配符子域，您可以使用

setDomains('.example.com');or setDomains('\*.example.com');

setCustomUrl( string ) 覆盖页的报告的URL

setReferrerUrl( string ) 重写HTTP Referer检测

setSiteId( integer ) 指定网站的冗余getTracker()构造函数 可以指定ID

setApiUrl( string ) 指定HTTP URL Piwik API端点。指向Piwik根目录，例如http://piwik.example.org/或HTTPS：/ /组织/ Piwik /例。这个函数时是非常有用的“叠加”的报告是不工作。默认情况下，你不需要使用此功能。

setTrackerUrl( string ) 指定使用piwik服务器的URL。多余的：可以指定gettracker()构造函数。

setDownloadClasses( string | **array** ) 设置下载类（除了piwik\_download）

setDownloadExtensions( string ) 设置文件扩展名列表被公认为下载。例如：“DOC | PDF | txt”

addDownloadExtensions( string ) 指定附加的文件扩展名被公认为下载。例如：“DOC | PDF | txt”

setIgnoreClasses( string | **array** ) 如果要设置类中存在的链路忽视（除了piwik\_ignore）

setLinkTrackingTimer( integer ) 在毫秒跟踪链接设置延迟。

discardHashTag( bool ) 设置为true，没有记录的散列标签（锚）部分的URL

setGenerationTimeMs(generationTime) 默认情况下，使用浏览器的DOM API Piwik定时准确地确定需要生成并下载页面的时间。你可能会覆盖值指定一个毫秒的价值在这里。

appendToTrackingUrl(appendToUrl) 添加一个自定义字符串的HTTP请求结束piwik.php？

setDoNotTrack( bool ) 设置为true，不跟踪用户谁选择了使用Mozilla的跟踪（提出）不跟踪设置。

disableCookies()禁用所有第一方Cookie。现有的这个网站Piwik Cookie将在下页查看删除。

deleteCookies()删除“跟踪Cookie目前目前集（创造新的访问时，这是有用的）

killFrame()阻止跟踪frame/ iframe的。

redirectFile( url ) 强制浏览器加载生活网址如果跟踪的网页是从当地的一个加载的文件（例如，保存到某人的桌面）。

setHeartBeatTimer( minimumVisitLength, heartBeatDelay ) 记录网页多久被视若minimumvisitlength（秒）获得的heartbeatdelay；确定如何经常更新服务器

getVisitorId()返回给客人16个字符的ID

getVisitorInfo()以数组形式返回访问者的cookie内容

getAttributionInfo()返回的游客属性阵列（参考信息和/或活动名称及关键字）

属性信息是由Piwik信用正确的来源（第一或最后的referrer）任何目标转换。

您也可以使用下列函数来获得数据的特定属性：

* piwikTracker.getAttributionCampaignName()
* piwikTracker.getAttributionCampaignKeyword()
* piwikTracker.getAttributionReferrerTimestamp()
* piwikTracker.getAttributionReferrerUrl()

setCustomVariable (index, name, value, scope) 设置自定义变量。

deleteCustomVariable (index, scope ) 删除自定义变量。

getCustomVariable (index, scope ) 检索一个自定义变量

setCampaignNameKey(name) 设置广告名称参数（S）。（帮助：自定义广告参数名称）

setCampaignKeywordKey(keyword) 设置广告的关键字参数（S）。（帮助：定制广告的关键字参数名）

setConversionAttributionFirstReferrer( bool ) 设置为true属性转换为第一引荐。默认情况下，转换是由于最近的引荐。

## 配置和跟踪Cookies

从piwik 1.2，用于第一方Cookie。必须考虑到的保留时间和避免冲突与其他Cookie，跟踪，和应用程序。

setCookieNamePrefix( prefix ) 默认的前缀是“PK”。

setCookieDomain( domain ) 默认的是文件的域；如果你的网站可以在www.example.com和example.com参观，你可以使用：

tracker.setCookieDomain('.example.com'); or  tracker.setCookieDomain('\*.example.com');

setCookiePath( path ) 默认是“/”。

setVisitorCookieTimeout( seconds ) 默认是2年

setReferralCookieTimeout( seconds ) 默认是六个月

setSessionCookieTimeout( seconds ) 默认是30分钟

# 单元测试覆盖piwik.js

使用piwik的JavaScript跟踪API通过广泛的JavaScript单元测试套件包括保证代码质量尽可能高，而我们从来没有打破这个功能。试验用量子系统。运行测试，简单的检测使用piwik的Git仓库，去

/path/to/piwik/tests/javascript/

.。测试运行在浏览器里面。

使用piwik的JavaScript API已与众多的Web浏览器测试。覆盖率的最大化，我们使用像crossbrowsertesting.com和browsershots.org服务。

# 缩小piwik.js

piwik.js缩小的减少你的网站访客需要下载大小。YUI压缩机是用来减少JavaScript（更多信息）。你可以找到原始的非压缩版

[/js/piwik.js](https://github.com/piwik/piwik/blob/master/js/piwik.js#L1).

// 增加第一个产品到订单

\_paq.push(['addEcommerceItem',

"9780786706211", // (必填) 产品唯一ID

"Endurance: Shackleton's Incredible Voyage", // (可选) 产品名称

"Adventure Books", // (可选) 产品类型. 你可以用一个数组表示（最多5个） ["Books", "New releases", "Biography"]

8.8, // (可选，建议填写) 产品单价

1 // (可选, 默认为1) 产品数量

]);

//这里你还可以增加多个产品到订单里

[...]

// 制定和发送订单明显到采集服务器

\_paq.push(['trackEcommerceOrder',

"A10000123", // (必填) Unique Order ID

35, // (必填) 订单收入总额（包括税收，航运，和减去折扣）

30, // (可选) 订单小计（不包括运费）

5.5, // (可选) 税收

4.5, // (可选) 运输费

false // (可选)折扣 (false表示 未定义)

]);

// 我们建议离开的订单确认页面时候调用trackPageView（）

\_paq.push(['trackPageView']);

[...]

**购物车相关：**

该代码是添加的每一次“购物车”的页面或当点击“加入购物车”按钮被加载：

[...]

// 增加第一个商品到购物车

\_paq.push(['addEcommerceItem',

"9780786706211", // (required) SKU: Product unique identifier

"Endurance: Shackleton's Incredible Voyage", // (optional) Product name

["Adventure Books", "Best sellers"], // (optional) Product category, string or array of up to 5 categories

8.8, // (recommended) Product price

1 // (optional, default to 1) Product quantity

]);

// 这里要增加购物车内所有的商品，就算没增加或者减少商品都要编写

[...]

// 记录购物车

\_paq.push(['trackEcommerceCartUpdate',

15.5]); // (可选，但建议填写)购物车总额

\_paq.push(['trackPageView']);

[...]

**跟踪产品的网页和类别页面视图**

[...]

//所有参数可选，但我们建议填上产品ID或者名字

\_paq.push(['setEcommerceView',

"9780786706211", // (required) SKU: Product unique identifier

"Endurance: Shackleton's Incredible Voyage", // (optional) Product name

"Adventure Books", // (optional) 产品类别, or array of up to 5 categories

20.11 // (optional) 页面上产品的价格

]);

\_paq.push(['trackPageView']);

[...]

例如跟踪类别页面的视图

这段代码增加在类别为 “Adventure Books”的页面里

[...]

// 在类别页面，产品ID和名字都是不用的，所有设置为false

\_paq.push(['setEcommerceView',

productSku = false, // No product on Category page

productName = false, // No product on Category page

category = "Adventure Books" // Category Page, or array of up to 5 categories

]);

\_paq.push(['trackPageView']);

[...]

# 二次开发实例

1，在应用技术方面：分析技术主要有：HTTP协议头信息技术，第一方COOKIE，第三方COOKIE，P3P协议COOKIE，LSO技术以及Web Beacons和javascript技术。

2，系统核心工作流程：

    1，访问者通过浏览器向用户服务器发送请求。

    2，门户服务器响应请求，返回预先设置了**Piwik**的跟踪代码页面，

    3，客户端浏览器执行TAG标记，向**Piwik**请求数据获取的js代码。

    4，**Piwik**服务器响应请求，返回用于收集网页，浏览器和Cookie信息的js代码。

    5，客户端浏览器执行js代码，发送相关网页被访问的行为数据给**Piwik**服务器。

    6，**Piwik**服务器产生点击流日志，保存于后台数据库中，用来日后做分析。

**Piwik**收集的信息有：被系统屏蔽后的IP地址信息，访问来源链接，第一次访问网站时间，累计页面浏览数，和第几次来访问数据等。

    由于**Piwik**比较主用用户隐私，对IP信息进行了屏蔽，这也是**Piwik**的不足之处，为了精准定位访问人群，我们可以对**Piwik**进行二次开发。

**Piwik**收集IP信息的二次开发：

        （1）**Piwik**后台数据库改造

**Piwik**将用户的访问时间信息，访问事件和浏览器的环境等信息记录载mysql的**piwik**\_log\_visit表中，我们可以添加一个ipinfo 字段来存放用户的真实地址。

            alter table **piwik**\_log\_visit add column `ipinfo` char(20) not null default 0 after location\_provider

        （2）新建一个获取IP信息的类：**piwik**提供插件扩展功能的方法，所有的二次开发类必须遵守次方法。

            具体步骤：

                1，在**Piwik**的plugins方法中建立ipinfo.php扩展类，此类继承**Piwik**\_Plugin类，重写**Piwik**\_Plugin中getListHooksRegistered方法，获取已经注册的**Tracker**.Visit.setVisitorIp钩子。注意：钩子（hook)在wordpress 中也有提到。将此钩子绑定一个方法来获取ip信息，将IP信息存入**Piwik**\_log\_visit表的ipinfo字段中。

                ipinfo.php 部分代码：

                    class ipinfo extends **Piwik**\_Plugin {

                        public function getListHooksRegistered() {

                                //注册一个**Tracker**.Visit.setVisitorIp钩子类获取IP信息。

                                return array('**Tracker**.Visit.setVisitorIp'=>setVisitorIpAddress);

                        }

                        static public function applyIPMash($ip,$maskLength) {

                            //解开屏蔽ip信息

                             $i = **Piwik**\_Common:strlen($ip);

                            if($maskLength > $i) {

                                $maskLength = $i;

                            }

                            while($maskLegth -- >0) {

                                    $ip[--$i] = chr(0);

                            }

                            return $ip;

                        }

                        function setVisitorIpAddress($notifiaction) {

                            //奖励与**Tracker**.Visit.setVisitorIp钩子的绑定方法。

                            $ip = &$notification ->getNotificationObject();

                            //得到的参数引用相关联的钩子

                            $ip = self::applyIPMask($ip,**Piwik**\_**Tracker**\_Config::getInstance()->**Tracker**['ip\_address\_mask\_length']);

                        }

                    }

**Piwik**\_Plugin.php文件位置:安装目录/core/Plugin.php中。此类为抽象类，有兴趣的可以自行查阅。

# Piwik数据库调用API整理

By [strivescript](http://www.factj.com/archives/author/strivescript) | 2014 年 2 月 8 日

[0 Comment](http://www.factj.com/archives/271.html#comments)

**piwik**的数据库调用主要分成来源于两种情况：后台管理时候，跟踪用户的时候

1. 后台管理时候  
   在后台管理的时候，**piwik**的dao定义在**piwik**里面，主要使用的是  
   **Piwik**类里面的createDatabaseObject方法  
   Zend\_Registry::set(‘db’,$db)  
   我们使用postgreSQL的时候使用的是  
   Zend\_Registry::set(‘pg’,$pg)  
   调用的方法很简单，直接使用
2. if (isset($GLOBALS['**PIWIK**\_USE\_PG']) && $GLOBALS['**PIWIK**\_USE\_PG'] == true) {
3. $db = Zend\_Registry::get('pg');
4. }else{
5. $db = Zend\_Registry::get('db');

}

1. 跟踪用户的时候  
   初始化数据库的连接是在core/**Tracker**.php里面  
   类**Piwik**\_**tracker**\_Visit里面的connectDatabaseIfNotConnected()方法负责干这个活
2. public static function connectDatabaseIfNotConnected()
3. {
4. if (!is\_null(self::$db)) {
5. return;
6. }
7. try {
8. $db = null;
9. **Piwik**\_PostEvent('**Tracker**.createDatabase', $db);
10. if (is\_null($db)) {
11. $db = self::connectPiwikTrackerDb();
12. }
13. self::$db = $db;
14. } catch (Exception $e) {
15. throw new **Piwik**\_**Tracker**\_Db\_Exception($e->getMessage(), $e->getCode());
16. }
17. //初始化pg连接
18. if ($GLOBALS['**PIWIK**\_USE\_PG'] == true) {
19. try {
20. $pg = null;
21. **Piwik**\_PostEvent('**Tracker**.createDatabase', $pg);
22. if (is\_null($pg)) {
23. $pg = self::connectPiwikTrackerPg();
24. }
25. self::$pg = $pg;
26. } catch (Exception $e) {
27. throw new **Piwik**\_**Tracker**\_Db\_Exception($e->getMessage(), $e->getCode());
28. }
29. }
30. }
31. /\*\*
32. \* @return **Piwik**\_**Tracker**\_Db
33. \*/
34. public static function getDatabase()
35. {
36. return self::$db;
37. }
38. /\*\*
39. \* @return **Piwik**\_**Tracker**\_Db
40. \*/
41. public static function getPostgreSQL()
42. {
43. return self::$pg;

}

后面的两个方法，返回生成好的数据库连接对象

1. 公共使用方法  
   在plugins里面，可以使用core/PluginsFunctions/Sql.php中的方法来使用  
   **Piwik**\_Sql  
   这个类提供了很多很方便使用的方法，而且这个方式在后台和跟踪模式下面都可以很好的使用

if (isset($GLOBALS['**PIWIK**\_USE\_PG']) && $GLOBALS['**PIWIK**\_USE\_PG'] == true) {

if (!empty($GLOBALS['**PIWIK**\_**TRACKER**\_MODE'])) {

$db = **Piwik**\_**Tracker**::getPostgreSQL();

}

if ($db === null) {

$db = Zend\_Registry::get('pg');

}

}else{

if (!empty($GLOBALS['**PIWIK**\_**TRACKER**\_MODE'])) {

$db = **Piwik**\_**Tracker**::getDatabase();

}

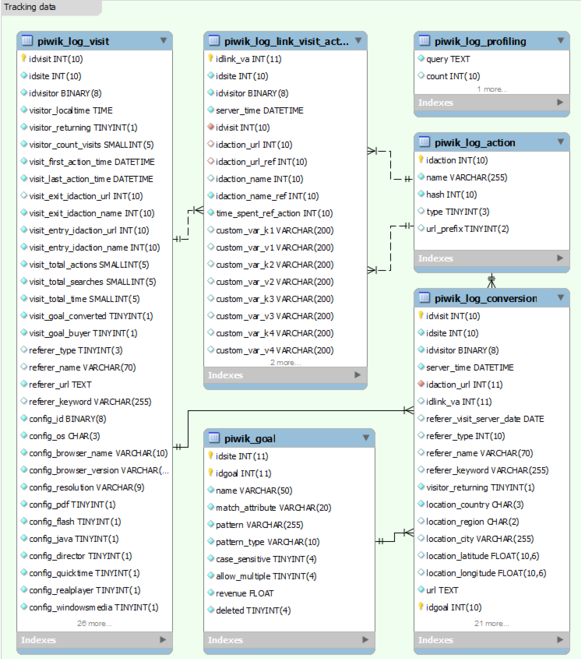
if ($db === null) {

$db = Zend\_Registry::get('db');

}

}

# Tracking data模型



# 新建插件介绍

**Piwik**提供了插件的结构，这种结构允许我们去建立插件，而无需修改**piwik**的核心代码

这篇文章将会介绍**Piwik**插件的结构，并展示给我们如何去创建我们自己的插件。

1，插件的作用

    1，收集目前**piwik**目前不能跟踪的数据

    2，定义新的小物件（widgets)能够用当前可用的或者新的数据形成可见的图像等。（类似形成报表？）

    3，定义新的或者覆盖已经存在的项目来定制我们自己的**Piwik**

    4,提供某种API，并且自动生成我们易于理解的数据格式。

2，统计跟踪是如何工作的。

    在我们开始写我们自己的插件之前，我们要知道**piwik**是如何跟踪并且加工数据的。最好的办法是我们安装一个新的**Piwik**程序，并观察他的数据库布局。

    数据库图示地址：http://cdn.geekmonkey.org/assets/files/000/000/025/screen/**piwik**-db-schema.png?1356779711

3，跟踪：

    每当有访问者来访问我们可用**piwik**的网站的时候，JavaScript都会提交一个访问过**piwik**.php脚本的关于访问者以及访问页面的最基本的数据设置信息，这个脚本会携带合法的数据，并将他存放到数据库中去。跟踪插件能够修改并添加数据

    例如：

        \_log\_visit  表：存储每个新的访问实体

        \_log\_link\_visit\_action:访问者随后访问的页面。

        \_log\_action: 目前为止，**piwik**追踪到的目前访问者访问过的页面作为实体存放到这类表中。这类表是被\_log\_link\_visit\_action引用的。

4，处理

    未加工的数据是指在我们追踪过程中不能被直接展示出来的数据。对于数据库来说，计算所有的请求负荷太高。因此，**piwik**处理这些数据为可视化，这个处理叫做存档。

    当你访问**piwik**接口的时候就会产生存档。在高流量网站上，存档应该由cronjob来完成，原数据被处理并放到存档表中。

5，存档

    每个月的由**piwik**监测的数据，在数据库中会创建两个表，这两个表分别为：\_archive\_YEAR\_MONTH，\_archive\_numeric\_YEAR\_MONTH，包含的内容为那个月的处理数据

比如2013-06那么对应的表为：**piwik**\_archive\_2013\_06,**piwik**\_archive\_numeric\_2013\_06

6，插件结构：

        文件结构：

            插件都放在plugins/文件中，每个插件都是它的子文件夹

plugins/

|-- VisitFrequency

| |-- API.php

| |-- Controller.php

| |-- templates

| | |-- index.tpl

| | `-- sparklines.tpl

| `-- VisitFrequency.php

        多数的插件都不少与3个php文件和一些视图模板。

        基本布局

            为了能够让**piwik**插件能够工作，插件系统要遵循一些约定

                <a>:插件要有一个和它所在插件目录名称相同的php文件。所以如果你想开发一个名为VisitorForeCast的插件，就需要创建一个名为 VisitorForeCast的目录，并且目录里包含一个名为VisitorForeCast.php文件。

                <b>:创建一个继承**Piwik**\_Plugin类的**Piwik**\_VisitorForeCast 类

<?php class **Piwik**\_VisitorForecast extends **Piwik**\_Plugin { }

*plugins/VisitorForecast/VisitorForecast.php*

**Piwik**\_Plugin有一个抽象方法getInformation()的抽象类，我们要继承，他的返回值为如下内容，当然也可以参考其他插件，下面内容为core/Plugin.php 内容，这个数组必须返回一个关联数组，这个数组包含有关于这个插件相关信息。

  /\*\*  
 21      \* Returns the plugin details  
 22      \* - 'description' => string        // 1-2 sentence description of the plugin  
 23      \* - 'author' => string             // plugin author  
 24      \* - 'author\_homepage' => string    // author homepage URL (or email "mailto:youremail@example.org")  
 25      \* - 'homepage' => string           // plugin homepage URL  
 26      \* - 'license' => string            // plugin license  
 27      \* - 'license\_homepage' => string   // license homepage URL  
 28      \* - 'version' => string            // plugin version number; examples and 3rd party plugins must not use **Piwik**\_Version::VERSION; 3rd pa    rty plugins must increment the version number with each plugin release  
 29      \* - 'translationAvailable' => bool // is there a translation file in plugins/your-plugin/lang/\* ?  
 30      \* - 'TrackerPlugin' => bool        // should we load this plugin during the stats logging **process**?  
 31      \*  
 32      \* [@return](http://my.oschina.net/u/556800) array  
 33      \*/

    VisitorForeCast.php 文件的完整内容为：

        <?php

            class **Piwik**\_VisitorCast extends **Piwik**\_plugin {

                public function getInformation() {

                        return array(

                                 'description' => 'Provide a forecast of visits for the day',  
                                  'author' => 'Your Name',  
                                  'author\_homepage' => 'http://yourwebsite.com',  
                                  'homepage' => 'http://example.com',  
                                  'license' => 'GPL v3 or later',  
                                  'license\_homepage' => 'http://www.gnu.org/licenses/gpl.html',  
                                  'version' => '0.1',  
                                  'translationAvailable' => false,

                        );

                }

             }

        ?>

7,翻译

**Piwik**可用超过45种语言，当我们想要创建我们习惯的插件的时候，我们可以利用**piwik**国际化的特性。如果想让我们的插件支持转译（translations)，我们需要 在getInformation方法中设置‘translationAvailable’=true,设置完之后，**Piwik**将会在我们的插件的子文件夹lang/中寻找这个支持翻译功能的字符串，无论我们想支持那种语言，我们都要准本一个用ISO（介http://en.wikipedia.org/wiki/List\_of\_ISO\_639-1\_codes）,639-1（http://en.wikipedia.org/wiki/List\_of\_ISO\_639-1\_codes）,alpha-2（http://en.wikipedia.org/wiki/List\_of\_ISO\_639-1\_codes）命名的php文件。

        每个语言文件(langle file)都要包含一个名字为：$translations的关联数组，这个数组的键（$key)映射到某个提供翻译（translation)的文件。而这些键($keys)是所有的提供翻译的文件之间共享的。当我们切换支持不同语言的时候，所依赖的正是这个语言文件。通常，这些键是用插件的名字作为前缀，例如，我们的VisitorForecast插件，如果想支持英文翻译那么就要命名为en.php ，完整路径（pugins/VisitorForecast/lang/en.php)，内容如下

<?php

    $translations = array('VisitorForecast\_PluginDescription'=>'')

?>

现在，我们给了翻译文件一个描述行的键（VisitorForecast\_PluginDescription），这样一来，翻译者工作起来就非常容易。**Piwik**提供了一个名为**Piwik**\_Translate($key)方法，它将会根据（依赖）用户选定语言返回用户想要的转化语言版本。

public function getInformation() { return array( 'description' => **Piwik**\_Translate('VisitorForecast\_PluginDescription'),

[..]

);

}

*Plugin Information with i18n*

    好了，翻译的内容基本介绍完了，有点晕。。。我们来整理下思路，如果看过wordpress的，可能会觉得有些地方有点像，都是为开发插件提供了最大的便利。便于部署，而不用动核心代码。

    现在我们已经有两个文件。

        1，**Piwik**\_VisitorForecast.php， 它和插件的名字基本相同，只是用**Piwik**做为前缀，并且，他要继承**Piwik**\_Plugin类。它有一个方法名字叫getInformation(),它返回一个关于我们开发插件的描述信息，作者，作者的主页，版本信息等。到现在此文件中的内容应该是如下内容

        class **Piwik**\_VisitorForecast extends **Piwik**\_Plugin{  
            public function getInformation() {  
                $info = array(  
                    //注意：VisitorForecast\_PluginDescription'是我们在en.php文件中数组的键  
                    'description' =>**Piwik**\_Translate('VisitorForecast\_PluginDescription'),                                    'author'=>'author\_name',

                    'author\_home'=>'home page',                                         'version'=>**Piwik**\_Version::VERSION,

                    );  
                    return $info;  
            }  
              
        }

        2，en.php文件：他主要是为**Piwik**\_Translage方法提供那个键，使得他获取到en.php文件中的内容，那例子中的en.php文件中的内容就应该是下面这样的。

            命名为en.php ? 为什么，上边说了他要符合ISO,639-1,alpha-2的命名，具体可以看上边三个字母后面的链接。就知道为什么是en.php 而不是eng.php了。  
            <?php  
                    $translations = array(  
                            'VisitorForecast\_PluginDescription'=>'Provide a forecast of visits for the day',  
                    );  
            ?>

8，Hooks(钩子，wordpress中也有，我们部门经理(大牛)很提倡用wordpress哦!)

    在**Piwik**内部，用了一个名字叫EventDispatcher(事件调度）的PHP包。我们的插件能将一个函数钩到某个预定义的事件上，当事件触发的时候会调用这个函数。  
     如果我们的插件需要注册任何钩子，我们就需要在我们的插件类中实现getListHooksRegistered 方法。还必须要返回一个数组，将钩子的名称和回调函数映射。

    比如说，如果你的插件想要注意到每个新的访问者，**Piwik**提供了一个叫**Tracker**.newVisitorInformation  
钩子，每当**Piwik**跟踪到一个新的访问者的时候，它就会被触发。

public function getListHooksRegistered(){

    return array(

       '**Tracker**.newVisitorInformation'=>'addVisitorInformation'

    );

}

public function addVisitorInformation($notification) {

    //我们通过$notification参数的引用和钩子关联起来。

    $visitorInfo = & $notification->getNotificationObject();

    //获取referrer\_url

    $url = $visitorInfo['referrer\_url'];

    //我们可以用这个url做一些事情

}

    当事件触发的时候，回调函数会取回一个对象通知，这个对象的内容取决于钩子，在这个例子中，这个对象是个数组，包含了访问者的信息

**Piwik**文档提供了钩子的文档，描述了当钩子被出发的时候将会返回什么样的对象，地址：  
[http://**piwik**.org/docs/plugins/hooks/](http://piwik.org/docs/plugins/hooks/)  
  
9，方法（函数）

    待序。。。

# Tracker介绍

优化tracker：<http://piwik.org/faq/how-to/faq_17033/>

（读取WEB服务器日志文件）

<http://112.96.28.249:8080/uba_test/piwik.php?action_name=wap.17wo.cn%2F%D2%BB%C6%F0%CE%D6&idsite=4&rec=1&r=538714&h=14&m=24&s=47&url=http%3A%2F%2Fwap.17wo.cn%2Fmusicale%2FProductIndex.action&_id=2012967a4b262a0d&_idts=1409725456&_idvc=1&_idn=0&_refts=0&_viewts=1409725456&cookie=1&res=320x480&gt_ms=425>

本节列出了不同的查询参数，通过跟踪API。一些这些领域的数据将不在你的应用程序/软件将是可用的，但是你应该提供尽可能多的信息

注：所有参数值为字符串（如“URL”，“action\_name’，等）必须URL编码。

猜想：

\_refts：请求提交的时间戳

\_idn：id的名字

## 必要参数：

idsite ：我们追踪访问/行为的网站的身份（设定的网站ID）。

rec：需要跟踪，必须设置一个，如& rec = 1。（没什么特别意义）

url：当前行为的完整URL。

## 推荐可选参数：

action\_name：被跟踪的行为的标题。它可以用“/”分隔设置行为的一个或者多个类别。例如，Help / Feedback 表示行为Feedback在类别Help里面。

\_id：唯一的访客ID，必须是16个字符的十六进制字符串。每一个访客必须分配一个不同的身份，而这个身份在设置后不能更改。如果不设置此值仍将追踪访问，但访客的度量可能是不准确的。

rand：意思就是保证生成一个随机值，以免跟踪请求被浏览器或者代理服务器缓存。

apiv：参数& apiv = 1，定义API使用的版本（目前都设置为1）

## 可选参数：

urlref：完整的HTTP referrer URL。此值用于确定访客如何到你的网站（IE，通过一个网站，搜索引擎或活动）。

\_cvar：访问自定义变量的范围。这是一个JSON编码的自定义变量数组的字符串。

\_idvc：该访问者当前的访问数。要正确设置此值，它将需要存储为你的应用程序中每个访问者的值（使用会话或数据库）。然后你可以手动增加每一个新的访问或“会话”，取决于你如何定义一个访问。这个值是用来填充报表Visitors > Engagement > Visits的访问数。

\_viewts：这位客人之前访问UNIX时间戳。这个参数是用来填充报告Visitors > Engagement > Visits自从上一个访问日期数。

\_idts：第一次访问这个访问者的UNIX时间戳。这可以被设置为日期在用户开始使用你的软件，或者当他/她创建了一个帐户。这个参数是用来填充Goals > Days转换报告。

\_rcn：活动的名称。用来填充Referrers > Campaigns报告。注：此参数仅用于访问的第一个页面。

\_rck：活动关键词。用来填充Referrers > Campaigns报告（点击活动运行展示所有关键词）。注：此参数仅用于访问的第一个页面。

res：该装置的访客使用的分辨率，如1280x1024。

h：当前小时（访客本地时间）。

m: 当前分钟（访客本地时间）。

s: 当前秒（访客本地时间）.

## 被访问者使用的插件，可以通过设置以下参数为1：

fla(Flash),

java(Java),

dir (Director),

qt(Quicktime),

realp(Real Player),

pdf(PDF),

wma(Windows Media),

gears(Gears),

ag(Silverlight).

ua：重写the User-Agent HTTP头字段的值，用来检测操作系统和浏览器。

lang：重写Accept-Language HTTP头字段的值，如果GeoIP未启用用来检测访问者的国家。

## 可选的action信息（衡量网页浏览，网页链接，下载，网站搜索）:

cvar:页面自定义变量的范围。这是一个JSON编码的自定义变量数组的字符串。

link：一个外部链接的用户已打开。用于跟踪网页链接点击。建议同时设置URL参数相同的值。

download：用户已下载的URL文件。用于跟踪下载。我们建议同时设置URL参数相同的值。

search:网站的搜索关键词。当指定时，请求将不会作为一个正常的浏览量跟踪，而应将作为一个站点的搜索请求的跟踪。

search\_cat:当搜索是指定的，可以选择此参数指定搜索类别。

search\_count:当搜索指定，建议设立search\_count在搜索结果显示在搜索结果页面数。当关键词和search\_count = 0他们将出现在“没有结果的搜索关键词“报告跟踪。

idgoal:如果被指定，将触发的网站被这个ID跟踪目标转换

revenue：一个货币值，产生目标转换的收入，在idgoal被指定的情况下使用。

gt\_ms ：从服务器生成这个动作的时间，以毫秒为单位。此值用于在网页的网址，网页标题报道页面速度报告平均时间过程，以及一个站点范围的平均运行的服务器的速度。注意：当使用JavaScript跟踪该值是服务器产生响应+客户端下载响应时间设置IME。

cs ：被跟踪页的字符集。默认UTF-8

## 可选事件跟踪信息：

e\_c：事件类型，不能为空(eg. Videos, Music, Games...)

e\_a：事件行为，不能为空(eg. Play, Pause, Duration, Add Playlist, Downloaded, Clicked...)

e\_n：事件名称(eg. a Movie name, or Song name, or File name...)

e\_v：事件的值。必须是一个浮点数或整数的值（数值），而不是一个字符串。

## 电子商务的信息：

前提：使用下面的值去记录购物车或者订单信息，在此请求中必须设置&idgoal=0。

ec\_id：电子商务订单标识符的唯一字符串

revenue：电子商务订单总计

ec\_st：订单小计，不包括运费

ec\_tx：税收

ec\_sh：运费

ec\_dt：折扣

ec\_items：在电子商务订单项目。这是一个JSON编码的项目。每一个项目都包括这些订单信息：项目产品，项目名称，项目类别，项目价格，项目数量。

## 特殊参数：

前提：该请求必须设置&token\_auth= 超级管理员的token\_auth。

token\_auth：32字符的授权密钥用来认证的API请求

cip：重写访客IP，支持IPV4\IPV6

cdt：覆盖的时间的请求（通常是用当前时间）。这可以被用来记录在过去的访问量和页面浏览量。预期的格式是：2011-04-05 00:11:42（记住，编码到URL值！）必须是在UTC时区的发送日期。注意：如果你的记录在过去的日期，你将需要Piwik强行重新进程处理过去的日期

cid：定义访客ID，你必须设置这个值是十六进制的16字符的字符串（含01234567890abcdefabcdef只读字符）。当指定的用户ID，将被“执行”。这意味着，如果有一个与在最近的一次新的访问者ID，一个将被创建。如果访问是发现在最后的30分钟，然后用您指定的用户ID，新的行动将被记录在这一次。

new\_visit：如果设置为1，将强制创建这个访客为新访客，这个功能JavaScript也可用。

country：重写国家的值，应该设置为2个字母的国家简写，如fr, de, us.

region：区域，应该设置为MaxMind's GeoIP数据库里面的2个字母区域的简写。在每一个国家（地区）是位于第二列码的区域，在左向右的名称和国家代码。

city：城市，访客的城市，如guangzhou

lat:访客纬度。

long：访客经度。

例子：

http://piwik-server/piwik.php?\_cvar={"1":["OS","iphone 5.0"],"2":["Piwik Mobile Version","1.6.2"],"3":["Locale","en::en"],"4":["Num Accounts","2"]}&action\_name=View settings&url=http://mobileapp.piwik.org/window/settings &idsite=8876&rand=351459&h=18&m=13&s=3 &rec=1&apiv=1&cookie= &urlref=http://iphone.mobileapp.piwik.org&\_id=af344a398df83874 &\_idvc=19&res=320脳480&

## 大量的跟踪

一些应用程序如Piwik日志进口商，必须跟踪的多次访问，有时数十，数百，数千甚至更多。在每次访问或行动的一个HTTP请求这些请求跟踪会产生巨大的延误是由于需要发送一个HTTP请求的时间量，使用大量的跟踪功能，然而，这些要求可以发送一次使应用程序更有效。

发送一个请求的HTTP堆跟踪，后必须用JSON对象使用谷歌追踪终点了。该对象必须包含以下属性：

requests：个人跟踪请求一个数组。每个跟踪请求应查询字符串，如果你要跟踪单个行动。

token\_auth：（可选）token\_auth是在API页找到。指定此如果你使用任何需要的参数，只需

token\_auth

例子：

{ "requests": [ "?idsite=1&url=http://example.org&action\_name=Test bulk log Pageview&rec=1", "?idsite=1&url=http://example.net/test.htm&action\_name=Another bul k page view&rec=1" ], "token\_auth": "33dc3f2536d3025974cccb4b4d2d98f4" }

curl -i -X POST -d '{"requests":["?idsite=1&url=http://example.org&action\_name=Test bulk log Pageview&rec=1","?idsite=1&url=http://example.net/test.htm&action\_name=Another bulk page view&rec=1"]}' http://piwik.example.com/piwik.php

上面将2个行为执行一个HTTP请求

## 启动跟踪日志：

验证您的数据被正确跟踪，您可以启用日志记录在使用谷歌跟踪文件调试，piwik.php。

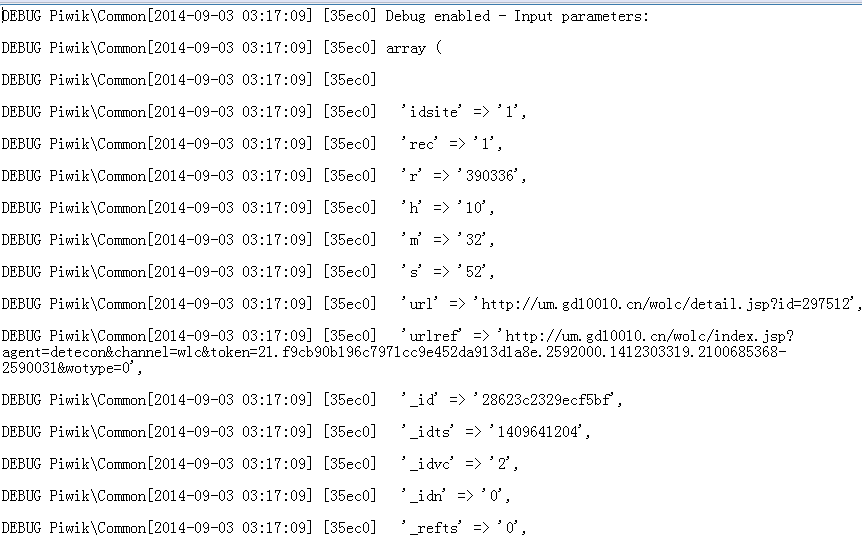
然后将输出跟踪要求跟踪日志消息而不是显示一个1×1的透明GIF信标。

/config/config.ini.php

[Tracker]

debug = 1

然后在浏览器观察请求。



# Piwik MVC结构：

## Piwik\FrontController

这个类可以单独调度请求到适当的插件控制器。

系统通过index.php使用此类处理所有请求，插件可以使用它来调用其他插件控制器行为。

例子：

**转发控制器请求：**

**public** **function** **myConfiguredRealtimeMap**() {

$\_GET['changeVisitAlpha'] = **false**;

$\_GET['removeOldVisits'] = **false**;

$\_GET['showFooterMessage'] = **false**;

**return** FrontController::getInstance()->dispatch('UserCountryMap', 'realtimeMap'); }

**使用其他控制器的方法：**

**public** **function** **myPopupWithRealtimeMap**() { $\_GET['changeVisitAlpha'] = **false**; $\_GET['removeOldVisits'] = **false**;

$\_GET['showFooterMessage'] = **false**;

$realtimeMap = FrontController::getInstance()->fetchDispatch('UserCountryMap', 'realtimeMap');

$view = **new** View('@MyPlugin/myPopupWithRealtimeMap.twig');

$view->realtimeMap = $realtimeMap;

**return** $realtimeMap->render(); }

**方法函数：**

此类定义了以下下方法：

getInstance() 返回一个实例，如果实例不存在，将会被创建；

Singleton 单例

Dispatch($module,$action,$parameters) 运行请求插件控制器的方法，返回控制器方法的结果

$module:控制器的插件 $action：控制器方法名 $parameters：方法参数

## 处理HTTP 请求

当HTTP请求被处理时候，MVC的第一段代码将被执行。每个前端请求都是发送到index.php文件，这个文件会创建一个FrontController实例去调度当前的请求。

FrontController会寻找module和action参数，再根据他去找出对应的插件和方法，如果action没定义，将会寻找相关插件的index方法。

For example, a URL with

module=MyPlugin&action=myAction

将会调用 **Piwik\Plugins\MyPlugin\Controller::myAction()** 方法. A URL with

module=MyPlugin&action=index

 将会调用 **Piwik\Plugins\MyPlugin\Controller::index()** 方法.

调用控制器方法必须要做一件事，就是输出点什么东西，作为一个插件开发者你可以在任何你喜欢的方式做到这一点，系统不会阻止你，但必须遵守一些约定，就是创建视图，通过查询APIs获得需要的数据然后渲染视图。

例如：

**class Controller extends \Piwik\Plugin\Controller {** **public** **function** **index**() {

$view = **new** View("@MyPlugin\index.twig");

$view->data = \Piwik\Plugins\MyPlugin\API::getInstance()->getData();

**return** $view->render(); } }

更多：<http://developer.piwik.org/guides/mvc-in-piwik>

# Report介绍

<http://developer.piwik.org/guides/all-about-analytics-data>

## 分析器相关

在分析系统中分析数据是搜索一组东西的模式，这些东西包括访问、网站行为和目标转换。

我们的搜索模式是把个性的东西分组创建成有意义的可识别的子集，系统是通过一个API来使用分组结果的数据、存储、显示和服务。

## 分析报告和元数据

系统存在2种类型的数据：报告和元数据，2者的区别在于报告是一个二维数组，元数据是单一的数值，报告一般包含着元数据，不过还可以包含其他人数据或者元数据的替代值。

## 核心元数据

所有的报告都由插件定义，元数据也可以由插件定义，但有几个元数据是由系统的核心定义和使用的，新的报告应该包含分析访客、行为类型或者转换这些元数据指标。

**下面是一个列表的核心指标，涉及一组访问：**

Visits:跟踪访问数（每个访问的间隔不超过30分钟），内部使用'nb\_visits'作指标名称；

**Unique Visitors**:唯一访问数，指标名'nb\_uniq\_visitors'；

**Actions**:跟踪行为数，指标名'nb\_actions'；

**Max Actions**:每个访问的最大行为数（活动数），指标名'max\_actions'；

**Sum Visit Length**: 每次访问的时间的总和，指标名'sum\_visit\_length'；

**Bounce Count**: 只有一个行为的访问数，指标名'bounce\_count'；

**Converted Visits**: 使用至少一次转换的访问数，包括一个网站每个目标的转换数，指标名'nb\_visits\_converted'；

**Conversions**:访问的转换数，包括一个网站每个目标的转换数，指标名'nb\_conversions'；

**Revenue**: 通过这些访问所产生的总收入。包括收入的网站和电子商务收入每一个目标，指标名'revenue'；

**下面是一个列表的核心指标，涉及到一个单一的行为类型：**

**Hits**: 访问次数，指标名'nb\_hits'；

**Sum Time Spent**: 用户访问一共花费的时间，指标名'sum\_time\_spent'；

**Sum Page Generation Time**: 服务器生成页面的总时间，指标名'sum\_time\_generation'；

**Hits With Generation Time**: 包括生成时间信息的访问次数，指标名'nb\_hits\_with\_time\_generation'；

**Min Page Generation Time**: 最小的页面生成时间，指标名'min\_time\_generation'；

**Max Page Generation Time**: 最大的页面生成时间，指标名'max\_time\_generation'；

**Unique Exit Visitors**: 页面唯一的退出访问数，指标名'exit\_nb\_uniq\_visitors'；

**Exit Visits**: 退出访问总数，指标名'exit\_nb\_visits'；

**Unique Entry Visitors**: 唯一开始访问数，指标名'entry\_nb\_uniq\_visitors'；

**Entry Visits**: 访问行为开始的数量，指标名'entry\_nb\_visits'；

**Entry Actions**:开始访问数，指标名 'entry\_nb\_actions'

**Entry Sum Visit Length**: 每个访问经过时间的总和，指标名'entry\_sum\_visit\_length'；

**Entry Bounce Count**:只包含同一个访问的数量，指标名 'entry\_bounce\_count'；

**Hits From Search**:使用搜索引擎访问的次数，指标名'nb\_hits\_following\_search'；

**下面是一个列表的核心指标，关系到电子商务的转换的集合（或所有订单或所有废弃的车）的一组访问记录：**

**Revenue Subtotal**: 订单的一部分或者放弃购物车的总成本，指标名'revenue\_subtotal'；

**Revenue Tax**: 总税额，指标名'revenue\_tax'；

**Revenue Shipping**: 总运费，指标名“revenue\_shipping”；

**Revenue Discount**: 折扣，指标名'revenue\_discount'；

**Ecommerce Item Count**: 订单数，指标名'items'；

**目标的具体指标：**

**下面是一个列表的核心指标，涉及一组访问网站的一个目标：**

**Goal Conversions**:目标或访问的转换数，指标名'goal\_%idGoal%\_nb\_conversions'；

**Goal Revenue**: 目标收益，指标名'goal\_%idGoal%\_revenue'；

**注意**：在上面显示的指标名称，“% idgoal %应该是一个目标ID替换。

目标的具体指标存储在点段‘goals’里面，这个字段包含一个目标ID的PHP数组和目标的指标名，这些指标值是通过过滤器AddColumnsProcessedMetricsGoal DataTable设置的。

## 处理元数据（指标）

考虑到效率问题（无论是存储过程的速度还是数据库的大小），许多指标是不存储在数据库中的。这些指标可以使用其他指标通过计算为报表服务。这些指标统称为**处理指标**。下面是处理指标，利用核心指标计算表。

各项的指标在需要的时候可以进行新增。

注意：一些处理指标可以能在下面出现多次，但他们在不同的报表中意义不同。

**下面是一个列表的处理指标，涉及一组访问：**

**Conversion Rate**: 转化率，指标名'conversion\_rate'；

**Actions Per Visit**: 平均访问数，指标名'nb\_actions\_per\_visit'；

**Average Time On Site**: 网站平均访问时间，指标名'avg\_time\_on\_site'；

**Bounce Rate**: 跳出率，访问了一个页面就离开，指标名'bounce\_rate'；

**下面是一个列表的处理指标，涉及到一个单一的行为类型：**

**Average Generation Time**: 平均生成时间，指标名'avg\_time\_generation'；

**Average Number of Search Result Pages Viewed**: 搜索的结果页面的平均数，指标名'nb\_pages\_per\_search'；

**Average Time On Page**: 平均页面发送时间，指标名'avg\_time\_on\_page'；

**Entry Bounce Rate**: 同一访问的跳出率，指标名'bounce\_rate'；

**Exit Rate**: 退出率， 整个访问完结再结束的比率，指标名'exit\_rate'；

**下面是一个列表的处理的指标，与设定的电子商务订单记录一组访问：**

**Average Order Revenue**: 每个订单的平均收入。指标名'avg\_order\_revenue'；

**下面是一个列表的处理指标，涉及到在一组命令或放弃购物车的电子商务项目：**

**Average Price**: 每个项目的平均价格，指标名'avg\_price'；

**Average Quantity**: 每个项目的平均数量，指标名'avg\_quantity'；

**Product Conversion Rate**: 产品转化率，指标名'conversion\_rate'；

**下面是一个列表的处理指标也是针对一个网站的一个目标：**

**Average Revenue per Visit**: 每个访问的平均收益，指标名：'goal\_%idGoal%\_revenue\_per\_visit'

注意：在上面显示的指标名称，% idgoal %应与目标问题的ID代替。

## 命名指标

插件需要计算和拥有自己的指标必须给他们以如下格式的名称：“pluginname\_metricname”

pluginname是插件名、metricname名称是指标的名称。

例如：“myplugin\_myfancymetric”；

这个命名是为了确定哪些插件定义的指标。不遵守本公约将结果存档过程中的错误。核心指标都有特别的名字，不遵守本公约。

### 报告和数据表

报告使用类DataTable存储在内存中，DataTable是一个行 的数组，而每行都是由所有字段的数据组成的数组。

这些行包括访问、行为、转换和一些其他的指标值，这些组合通过特殊的标签字段进行定义和描述，这些描述是怎样的取决于报告，例如

UserSettings.getBrowser 标签行 为firefox就表示所有访问都是使用这个浏览器的。

一些报告，像visitssummary.get将不会有一个标签栏。这些报告将只有一行，是指实体全套。（label字段的定义）

#### 行元数据（格式化字段值？）

除了指标，每行可以包含元数据。这通常会协助元数据标签栏在描述事物的行代表集。

一些有特殊意义的元数据。例如，元数据“logo”作为一个图像是用来描述行路径。这个图像一起显示的行的报告时，在界面上显示。该usersettings.getbrowser和usersettings.getos报告使用此元数据值显示为每个浏览器和操作系统上的一个图标。“URL”作为一个URL描述行。该行的标签为这个URL时，报告显示在用户界面。

#### 子报表

报告可以分层。在一份报告中每一行可以连接到另一个表中的数据。这些表中的任何行可以连接到多个表，如此永无止境。表上的行称为子报表。

子报表提供访问一行表示设置的进一步分析。例如，在actions.getpageurls报告包含的行描述一组页面视图的操作基于第一部分的页面的URL。如果这部分是目录，不是一个文件，该行可能有子表描述行的网页浏览行为是基于第二部分的页面的URL。

另一个例子：在referrers.getsearchengines报告包含每个搜索引擎，用于访问一行。每一行有一个子表，描述了用搜索引擎的关键词。子表列将包含特定关键字的访问，使用的指标值（在子表行确定）与一个特定的搜索引擎（由父行的确定）。

#### 命名报告

报告应以同样的方式被称为非核心指标。那就是，他们应该有如下格式：“pluginname\_reportname”

pluginname是插件名和reportname是报告名称。

例如：“myplugin\_myfancyreport”。

插件不遵守本公约将存档过程中导致错误。

#### 参数分析

报告和指标提供分析数据的一套东西。包括三部分：网站ID、时间段和分段元素（查询项？）。

网站ID：用于定义和区分网站，在HTTP请求中的参数名为idSite。

时间段：用于查询结果的时间段，在HTTP请求中的参数名为date或者period。

分段元素：选择基于布尔表达式使用访问性能考察。它是由段的所有HTTP请求查询参数指定可用于选择几乎任何可以想象的子集的访问。

分析参数通常存储在报告中作为元数据（即它们被存储为DataTable元数据）。

所有的报告和指标描述一组事物由这三个参数组成：the website, period and segment.

#### 报告和指标的保存（档案数据）

报告和指标被保存后被统称为档案数据，这意味着这些数据已经被缓存，无需重新计算。

保存报告和指标的索引为 网站ID，时间段和分段元素。日期和时间数据进行了计算和缓存也被连接到每个报告和指标。

#### 报告保存

报告是复杂的数据结构，在保存前我们有必要做一些额外的处理。

每个报告行的列表（an array of DataTable\Row instances）使用PHP的序列化函数进行处理然后使用gzcompress进行压缩。最后，在压缩数据上加上网站ID，时间段，分段元素和时间缓存，然后保存。

#### 记录

当报告被存档，我们将叫它为记录，我们给2者一个区别，因为多个报告有时候 可以通过一个记录生成。

例如，插件 UserSettings使用一个记录去保存浏览器信息，而这个记录包含了UserSettings.getBrowserVersion和UserSettings.getBrowser，第二个报告和第一个报告通过一种简单的处理方式生成一个新的报告 ，插件可以存档每个报告，但这大大的浪费了存储空间。

记录储存指南

必须注意尽量少保存记录。确保插入记录的归档数据之前，请遵循以下指导准则：

记录不应存储字符串列的名称。相反，他们应该用整数列ID取代（见列表现有指标）。

元数据可以添加使用现有的数据不应该被保存和报告。相反，他们应该在API方法转换成记录时增加到报告。

#### 归档处理过程

分析数据是计算和缓存的需求。当一个报告被请求时，系统会检查报告数据是否被缓存，如果没，就生成和缓存它。

归档逻辑（计算和缓存数据的逻辑）是被单独的插件定义的，当开始存档的时候，每个被插件定义的报告都一起归档而不是一个一个归档。

如果没有段是在数据查询提供的数据不能被找到，每一个插件的每一份报告将一次性生成并缓存。如果提供了一段，然后报告属于相同的插件作为请求的数据会产生和缓存。

插件格式

插件要存档的报告和指标定义一个类Archiver 继承Piwik\Plugin\Archiver.。这个类将被自动检测和实例化Piwik存档过程中。

#### 报告和指标的聚合

报告和指标是基于不同时间段进行计算。

按日期，visits/actions/conversions/etc. (称为日志数据)是它自己本身。

其他时间段，报告和指标标准的天的期间内都聚集在一起。例如，当生成一周期的报道，在本周内每天报告（IE，星期一，星期二，星期三，等）将查询和汇总在一起。这是不是聚集每个访问/行动/等，是整个星期跟踪更快，但产生相同的结果。

日志数据通过LogAggregator进行融合处理。

存档数据通过ArchiveProcessor::aggregateDataTableRecords和ArchiveProcessor::aggregateNumericMetrics进行融合处理，插件可以通过Piwik\Plugin\Archiver访问一个LogAggregator实例和一个ArchiveProcessor实例。

#### 报告和指标缓存

报告和指标使用ArchiveProcessor进行保存，指标使用ArchiveProcessor::insertNumericRecord插入，报告是先用DataTable::getSerialized进行序列化，在使用ArchiveProcessor::insertBlobRecord进行保存。

$archiveProcessor = *// ...*

*// insert a numeric value*

$myFancyMetric = *// ... calculate the metric value ...*

$archiveProcessor->insertNumericRecord('MyPlugin\_myFancyMetric', $myFancyMetric);

*// insert a record (with all of its subtables)*

$maxRowsInTable = Config::getInstance()->General['datatable\_archiving\_maximum\_rows\_standard'];j

$dataTable = *// ... build by aggregating visits ...*

$serializedData = $dataTable->getSerialized($maxRowsInTable, $maxRowsInSubtable = $maxRowsInTable,

$columnToSortBy = Metrics::INDEX\_NB\_VISITS);

$archiveProcessor->insertBlobRecords('MyPlugin\_myFancyReport', $serializedData);

#### 用定时任务提前缓存

虽然数据生成的需求，这将是非常低效的，如果我们依靠它为所有用户创建一个用户体验差。任何用户接收访问大量会之前能够看到他们的报道经验的大迟延。Piwik解决了这个问题，一个控制台命令启动存储过程。要执行的任务。

控制台命令可以为每一个网站，除了周期的每个时期档案数据范围。用于存储段的报告和指标也将被归档。

控制台命令会记得当它最后被执行，只会引发归档过程为一个网站如果以来已经被访问。

#### 禁用浏览器启动存档

对于有网站收到大量的访问用户，只允许按需存档通过浏览器会引起不良的延误。这些用户可以禁用浏览器启动存档。

#### 服务型报告

报告是通过单独的插件定义的API类服务。API方法访问持续记录转换成漂亮的报告和服务他们的报告可以通过Piwik API或HTTP响应代码（如控制器的方法）。

#### 把记录转换成报告

如上所述，记录和报告是不同的，记录的存储结构是不被任何人或其他软件所正常读取的，

因此，我们的API 不能简单地访问保存的数据，他们需要经过处理使得它可以查看。

DataTable过滤器

DataTable的实例，这是用来生成报告，通过遍历行和手动进行更改或通过DataTable的过滤器控制。DataTable过滤器以某种方式操纵DataTable的实例。有几个预定义的，让你不必写大量的代码做普通的事情。

做一个像样的报告涉及松开转变，使它存储更有效。列名可以从整数ID更改为字符串指标的名称通过replacecolumnnames DataTable过滤器：

$dataTable->filter('ReplaceColumnNames');

元数据和处理指标也应该添加在API方法。现有的过滤器（都在core/DataTable/Filter目录）可以用来执行这些任务。

#### API处理报告

<http://developer.piwik.org/guides/piwiks-reporting-api>

当报告返回一个API方法经历了一些额外的处理，根据查询参数设置的要求。要清楚地看到所发生的一个报告，阅读我们的报告API Piwik引导相关部分。

Piwik的报告API允许第三方应用程序通过HTTP请求访问数据和操作各种数据（任何其他报告或跟踪数据）。

##### API请求

每一个HTTP请求到api.index控制器的方法将被报告的API处理。要求必须有一个有效的查询参数的命名方法，引用调用的API的方法，例如：usersettings.getbrowser。

API请求以下列方式处理：

API.index 控制方法创建一个Piwik\API\Request实例并使用其调度API请求；

Piwik\API\Request 实例使用方法查询参数去确定是用哪个插件和API；

Piwik\API\Request 实例引用使用当前查询参数的插件的API方法；

询参数具有相同的名称作为方法的参数将被传递给方法，例如请求参数idSite对应方法参数$ idSite；

Piwik\API\Request实例使用ResponseBuilder实例处理API方法的结果和转换成需要输出的格式；

输出最后返回API.index并用echo输出。

##### API方法

报告API调用的方法，在每个插件的API类都能找到。这些类中的每个公共方法都作为一部分报告的API，除了那些文档包含@ignore（注释）。

查询参数作为方法参数传递。所以API方法必须承担的方法参数将字符串或者数组值。对象不能作为参数。

方法只被允许返回数值，字符串值，数组，DataTable实例或DataTable\Map实例。Piwik不知道如何格式化其他东西。

注意：当返回DataTable或DataTable\Map的情况下，过滤器将需要应用。确保队列，用于演示目的，直接适用于其他的过滤器。

如果一个方法抛出一个异常，其信息将会一并输出，可以通过改变responsebuilder调试期间显示堆栈跟踪，ResponseBuilder::DISPLAY\_BACKTRACE\_DEBUG为true.

API demo:

<http://demo.piwik.org/index.php?module=API&action=listAllAPI&idSite=7&period=day&date=yesterday>

##### 额外的报告处理

通过API方法返回的报告，无论是通过DataTable还是DataTable\Map，在输出前都通过报告相关的一些查询参数执行一套DataTable\BaseFilters。

以下是应用过滤器列表，他们做什么和什么查询参数控制是否运行或不。它们在列表中出现的顺序也在他们所订购的应用：

**Flattener**: 该过滤器把每层的DataTable合并在一起，并增加子表行，它只有设置了flat平台查询参数为1的时候被应用。

**Pattern:**删除不符合正则表达式的DataTable，当查询参数**filter**\_pattern设置为一个正则表达式时候被应用，\_column设置为正则表达式将应用于字段内（默认应用在label）。

PatternRecursive:删除DataTable一些不符合正则表达式的行或者子表行。**filter**\_pattern\_recursive被设置为正则表达式的时候被应用，**filter**\_column\_recursive置为正则表达式将应用于字段内（默认应用在label）。

**ExcludeLowPopulation:** 删除包含特定值的所有行的列低于最小阈值，参数**filter**\_excludelowpop被设置就会应用，它被设置去检查字段，参数**filter**\_excludelowpop\_value指定最小值（默认为0）。

**AddColumnsProcessedMetrics:**在报告的每行增加一些通用的处理指标，参数**filter**\_add\_columns\_when\_show\_all\_columns设置为1的时候应用。

**AddColumnsProcessedMetricsGoal:**增加处理指标到每个网站的每个目标的每行数据，filter\_update\_columns\_when\_show\_all\_goals设置为1时候启用，idGoal 决定了哪个目标，它可以是一个ID或者一个特殊的值：

AddColumnsProcessedMetricsGoal::GOALS\_OVERVIEW:增加目标概述而不是个人目标；

AddColumnsProcessedMetricsGoal::GOALS\_MINIMAL\_REPORT:只含一个目标指标revenue\_per\_visit；

AddColumnsProcessedMetricsGoal::GOALS\_FULL\_TABLE:展示所有目标指标，包括电子商务；

**idGoal**默认为**AddColumnsProcessedMetricsGoal::GOALS\_OVERVIEW**.

1.Sort:排序DataTable，设置filter\_sort\_column（排序字段）和filter\_sort\_order（排序方式asc、desc）；

2.Truncate:通过行索引删除所有行，filter\_truncate存在则启动，这个值应该是行序号。

3.Limit:删除不在索引范围内的行。当**filter\_limit**查询参数提供设置为一个整数，则启动。这是范围的大小。**Filter\_offset**查询参数决定范围的起始行索引。如果该参数设置为1，**Keep\_summary\_row**查询参数将确保摘要行保持在报告中。

4.SafeDecodeLabel: url解码然后清理**标签**列的值。始终应用此筛选器。

5. Queued Filters：所有的[数据表](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FDataTable)排队过滤器应用在这一点上。当**disable\_queued\_filters**查询参数存在且设置为1时，它们将不适用。

6. ColumnDelete：该操作是基于查询参与，以此来确定每行中要删除或要保留的列。如果定义了**hideColumns**查询参数或**showColumns**查询参数，将应用。两者应该是列名称的逗号分隔列表。

7.LabelFilter: 此筛选器将删除所有行除了那些由**标签**查询参数指定的。仅当**标签**查询参数设置后才适用。**标签**查询参数可以是单个值或创建子表中的行的路径。下降到子表，应包含>字符，例如urldir>urlsubdir>index。标签页可以是数组的值，例如，

**8.LabelFilter**： 此筛选器将删除所有行 （除一个 （或一些） 由**标签**查询参数指定。如果**标签**查询参数设置，只应用。**标签**查询参数可以是单个值或创建子表中的行的路径。下降到子表，应包含>字符，例如，label[]=arg1&label[]=arg2。

### 其他特殊的查询参数

这里有影响的报告的处理的方式的另外一些特殊的查询参数：

* **disable\_generic\_filters**： 如果设置为1，则编号 2 到9 的筛选器将不会应用。默认值为0则缺少此参数，该筛选器将启用。
* **format**： 确定返回值的[输出格式](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpiwiks-reporting-api%23output-formats)。这会影响所有返回的值，不论它是否是一份报告 （即存储在一个[数据表](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FDataTable)或[DataTable\Map](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FDataTable%2FMap)）。

## 输出格式

报告的 API 请求的输出是 API 方法返回值的序列化的字符串。此字符串的格式是由**格式**查询参数的值决定的。目前 Piwik 支持下列输出格式：

* [**xml**](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdemo.piwik.org%2Findex.php%3Fmodule%3DAPI%26method%3DUserCountry.getCity%26format%3Dxml%26idSite%3D7%26period%3Dday%26date%3Dyesterday%26expanded%3D1)
* [**json**](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdemo.piwik.org%2Findex.php%3Fmodule%3DAPI%26method%3DUserCountry.getCity%26format%3Djson%26idSite%3D7%26period%3Dday%26date%3Dyesterday%26expanded%3D1)
* [**csv**](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdemo.piwik.org%2Findex.php%3Fmodule%3DAPI%26method%3DUserCountry.getCity%26format%3Dcsv%26idSite%3D7%26period%3Dday%26date%3Dyesterday%26expanded%3D1)
* [**tsv**](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdemo.piwik.org%2Findex.php%3Fmodule%3DAPI%26method%3DUserCountry.getCity%26format%3Dtsv%26idSite%3D7%26period%3Dday%26date%3Dyesterday%26expanded%3D1)() Excel
* [**html**](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdemo.piwik.org%2Findex.php%3Fmodule%3DAPI%26method%3DUserCountry.getCity%26format%3Dhtml%26idSite%3D7%26period%3Dday%26date%3Dyesterday%26expanded%3D1)（一个简单的 HTML 表示，不使用[*报告可视化效果*](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fvisualizing-report-data)）。
* [**php**](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdemo.piwik.org%2Findex.php%3Fmodule%3DAPI%26method%3DUserCountry.getCity%26format%3Dphp%26idSite%3D7%26period%3Dday%26date%3Dyesterday%26expanded%3D1)（序列化的 PHP 数组）。

有是一个专用的输出格式值，请求来自 Piwik 使用[Piwik\API\Request](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FAPI%2FRequest)中的数据时可以使用。这种格式将迫使要作为未加工和反序列化的数据返回的结果。只应使用的是 PHP 中调用的 API 内 Piwik 时。

注： [Request::processRequest](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FAPI%2FRequest%23processrequest)方法自动使用原始的格式，因此在大多数情况下，您不需要指定format=original

.

## 特殊的 API 方法

一些 API 插件中的 API 方法具有特殊的意义和用法：

### 报表元数据

**API.getMetadata**、 **API.getReportMetadata**和**API.getProcessedReport** API 方法可用于获取有关一个或所有报告的信息。这些信息包括的报告，这些度量标准和[更多](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdemo.piwik.org%2Findex.php%3Fmodule%3DAPI%26method%3DAPI.getMetadata%26apiModule%3DUserCountry%26apiAction%3DgetCountry%26format%3Dxml%26idSite%3D7%26period%3Dday%26date%3Dyesterday%26expanded%3D1)的文件所载的指标.

这些方法可以采用第三方应用程序提供分析由 Piwik 存储接口。**API.getReportMetadata**，它返回的每个报表元数据，可以用于获得一个网站列表的所有可用的报告。**API.getMetadata**可用于获取有关单个报表的详细信息。**API.getProcessedReport**可以用来获取一份单一的报告以及报告中的数据的元数据。

报表元数据可以也可用于内 Piwik 动手术报告由用户指定的功能。例如， **ImageGraph**插件，输出使用报表数据图的图像，使用报表元数据值作为提示如何绘制输出关系图。**ScheduledReports**插件也以类似的方式使用报表元数据。

### 行演变

Piwik 的[行演化特征](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdemo.piwik.org%2Findex.php%3Fmodule%3DCoreHome%26action%3Dindex%26idSite%3D7%26period%3Dday%26date%3Dyesterday%23module%3DUserSettings%26action%3Dindex%26idSite%3D7%26period%3Dday%26date%3Dyesterday%26popover%3DRowAction%243ARowEvolution%243AUserSettings.getConfiguration%243A0%243AWindows%2425207%242520%24252F%242520Chrome%242520%24252F%2425201920x1080)，可通过用户界面也可通过报告 API 来体现。第三方应用程序可以使用**API.getRowEvolution**方法来获取[单个行进化数据](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdemo.piwik.org%2Findex.php%3Fmodule%3DAPI%26method%3DAPI.getRowEvolution%26idSite%3D7%26period%3Dday%26date%3D2013-11-01%2C2013-11-25%26apiModule%3DUserSettings%26apiAction%3DgetOS%26label%3DWindows%2B7)或[多行演变的数据](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdemo.piwik.org%2Findex.php%3Fmodule%3DAPI%26method%3DAPI.getRowEvolution%26idSite%3D7%26period%3Dday%26date%3D2013-11-01%2C2013-11-25%26apiModule%3DUserSettings%26apiAction%3DgetOS).

### 批量传输 API 请求 (API.getBulkRequest)

[像跟踪 API](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2Ftracking-api)，报告 API 支持批量传输请求。批量请求允许应用程序调用并检索到一个 HTTP 请求的多个 API 方法的结果。这可以用于该查询 Piwik HTTP API 在应用程序中节省时间。

若要发送批量请求，请发送一个 HTTP 请求到**API.getBulkRequest** API 方法。唯一所需的查询参数命名为**url**。它应该是一个数组的单个 API 请求 Url。举个例子：

http://demo.piwik.org/?module=API&method=API.getBulkRequest&format=xml&urls[]=module%3DAPI%26method%3DVisitorInterest.getNumberOfVisitsPerVisitDuration%26format%3DXML%26idSite%3D7%26period%3Dday%26date%3D2013-11-24%26expanded%3D1&urls[]=module%3DAPI%26method%3DUserSettings.getBrowser%26format%3DXML%26idSite%3D7%26period%3Dday%26date%3D2013-11-24%26expanded%3D1

此示例使用以下 API 发出的请求：

* module=API&method=UserSettings.getBrowser&format=XML&idSite=7&period=day&date=2013-11-24&expanded=1
* module=API&method=VisitorInterest.getNumberOfVisitsPerVisitDuration&format=XML&idSite=7&period=day&date=2013-11-24&expanded=1

注意： 单独的 API 方法是同步执行，所以对于长时间运行的 API 方法，使用批量请求可能是一个糟糕的主意。

### 其他方法

**API.get**： 我们将get定义为所有支持并合并这个结果的已加载的插件，方法所有输出的测量数据，该插件档案，所以**API.get**的结果是你 Piwik 安装支持 （为指定的网站及周期） 每个度量值的集合。

**API.getSegmentDimensionMetadata**： 返回对于每个受支持的[部分维度](http://www.microsofttranslator.com/bv.aspx?from=&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2Fsegmentation)的元数据。为每个细分维度返回以下信息：

* + 'type'：段维的类型。
  + 'category'：一个翻译的字符串，用来描述部分维度类别。
  + 'name'：翻译的字符串或描述部分维度本身的翻译令牌。
  + 'segment':段尺寸标识。这是什么获取部分表达式中使用。
  + 'acceptedValues':一个字符串，描述什么值应该用刀头尺寸。
  + 'sqlSegment':细分维度表列操作，例如，'log\_visit.idvisitor'.
  + 'sqlFilterValue':变换中段表达提供之前它在 SQL 表达式中使用的值的可选的 PHP 回调。
  + 'permission':是否当前用户可以使用此部分维度。
* **API.getSuggestedValuesForSegment**： 返回可以用一个指定的部分维度的值的列表。
* **API.getLogoUrl**： 返回的 URL 的 Piwik 标志。
* **API.getHeaderLogoUrl**： 返回的 URL Piwik 标志一个较小版本。

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## 什么被保存

Piwik 仍然存在两种主要类型的数据： 记录数据和存档数据。**日志数据**就是一切，Piwik 跟踪和**存档数据**缓存的分析数据。

Piwik 也存在其他更简单的形式的数据包括：

* 网站，
* 用户，
* 目标，
* 选项。

本指南介绍了正是此数据包含什么信息和 MySQL 后端的坚持到底它。

注： Piwik 使用 PHP 数组来保存数据将被持久化。当我们描述哪些信息是每个持久化实体中时，我们列出属性被用来在实体阵列中存储该属性的字符串名称。

## 日志数据持久性

有四种类型的日志数据、**访问**、**操作类型**、**转换**和**电子商务项目**。所有日志数据都保留在以类似的方式： 以高音量设置到不断添加新数据和更新非存在 （除了**访问**).

访问处于活动状态的同时，将更新**访问**数据。所以直到访问结束它很有可能 Piwik 将尝试更新它。

读取日志数据计算分析数据时，旧的数据有时将被删除 （通过[数据清洗功能](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fpiwik.org%2Fdocs%2Fmanaging-your-databases-size%2F)).

后端必须确保插入新的日志数据是最快的速度可能和聚合日志数据不是太慢了 （虽然很明显，更快地是更好的）。

### 访问

每次访问包含以下信息：

* 'idsite'： ID 的网站它被跟踪了
* 'idvisitor'： 访客 ID （8 个字节的二进制字符串）
* 'visitor\_localtime'： 访问日期时间在一天的访客的时间
* 'visitor\_returning': 这次访问是否对这位访客的首次访问
* 'visitor\_count\_visits'： 访客已向这一次数
* 'visitor\_days\_since\_last'： 的天数，因为这位访客的最后一次访问 （如果有）
* 'visitor\_days\_since\_order'： 自这位访客最后的命令 （如果有） 的天数
* 'visitor\_days\_since\_first'： 此访的第一次访问以来的天数
* 'visit\_first\_action\_time'： 此次访问的第一个行动日期时间
* 'visit\_last\_action\_time'： 此次访问的最后一步操作的日期时间
* 'visit\_exit\_idaction\_url'： 此次访问的最后一步操作的 URL 操作类型的 ID
* 'visit\_exit\_idaction\_name'： 此次访问的最后一步操作的页面标题行动类型的 ID
* 'visit\_entry\_idaction\_url'： 此次访问的第一个行动的 URL 操作类型的 ID
* 'visit\_entry\_idaction\_name': 这次访问的第一个行动的页标题行动类型的 ID
* 'visit\_total\_actions'： 在这次访问期间执行的操作的计数
* 'visit\_total\_searches'： 在这次访问期间执行的站点搜索的计数
* 'visit\_total\_events'： 在这次访问期间执行的自定义事件计数
* 'visit\_total\_time': 这次访问总的运行时间
* 'visit\_goal\_converted': 这次访问将不转换目标
* 'visit\_goal\_buyer'： 是否访客点在这次访问期间或不
* 'referer\_type'： 这位访客的引用的类型。可以是下列值之一：

**Common::REFERRER\_TYPE\_DIRECT\_ENTRY**： 如果将设置为此值，其他'referer\_...'字段没有任何意义。

**Common::REFERRER\_TYPE\_SEARCH\_ENGINE**： 如果将设置为此值，'referer\_url'它是搜索引擎的 url 和'referer\_keyword'（如果我们可以找到它） 所使用的关键字。

**Common::REFERRER\_TYPE\_WEBSITE**： 如果将设置为此值，'referer\_url'是该网站的 url。

**Common::REFERRER\_TYPE\_CAMPAIGN**： 如果将设置为此值，'referer\_name'是市场活动的名称。< — — TODO： 双重检查运动信息-->

* 'referer\_name'： 推荐人名称 ；它的意思取决于特定的引用类型
* 'referer\_url'： 引用的 URL ；它的意思取决于特定的引用类型
* 'referer\_keyword'： 如果一个搜索引擎是推荐使用的关键字
* 'config\_id': 这次访问的配置选项，包括操作系统、 浏览器名称、 浏览器版本、 浏览器语言、 IP 地址和所有的浏览器插件信息的哈希
* 'config\_os'： 短字符串识别作业系统用来进行这次访问。更多的信息，请参阅[UserAgentParser](https://github.com/piwik/piwik/blob/master/libs/UserAgentParser/UserAgentParser.php)
* 'config\_browser\_name'： 一个短的字符串，标识用于进行这次访问的浏览器。更多的信息，请参阅[UserAgentParser](https://github.com/piwik/piwik/blob/master/libs/UserAgentParser/UserAgentParser.php)
* 'config\_browser\_version'： 一个字符串，标识用于进行这次访问的浏览器的版本
* 'config\_resolution'： 一个字符串，标识用于进行这次访问 （例如，访问者的屏幕分辨率'1024x768')
* 'config\_pdf'： 是否访问者的浏览器可以查看 PDF 文件，或不
* 'config\_flash'： 是否访问者的浏览器可以查看 flash 文件，或不
* 'config\_java'： 访问者的浏览器是否可以运行 Java
* 'config\_director':
* 'config\_quicktime'： 访问者的浏览器是否使用 quicktime 要播放的媒体文件，或不
* 'config\_realplayer'： 是否访问者的浏览器可以播放 realplayer 媒体文件或不
* 'config\_windowsmedia'： 访问者的浏览器是否使用 windows 媒体播放器来播放媒体文件
* 'config\_gears':
* 'config\_silverlight'： 访问者的浏览器是否可以运行 silverlight 程序
* 'config\_cookie'： 访问者的浏览器是否启用与否的饼干
* 'location\_ip': 这次访问由该计算机的 IP 地址。可以[匿名](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fpiwik.org%2Fdocs%2Fprivacy%2F%23step-1-automatically-anonymize-visitor-ips)
* 'location\_browser\_lang'： 一个字符串，描述在访问者的浏览器中使用的语言
* 'location\_country'： 描述的该国第位访客浏览网站时位于两个字符字符串。由[UserCountry](https://github.com/piwik/piwik/tree/master/plugins/UserCountry)插件设置。
* 'location\_region'： 描述的该国来访者是在该区域的两个字符字符串。由[UserCountry](https://github.com/piwik/piwik/tree/master/plugins/UserCountry)插件设置。
* 'location\_city'： 一个字符串命名这座城市的访客是在同时访问该网站。由[UserCountry](https://github.com/piwik/piwik/tree/master/plugins/UserCountry)插件设置。
* 'location\_latitude'： 虽然他/她访问过的网站访客的纬度。由[UserCountry](https://github.com/piwik/piwik/tree/master/plugins/UserCountry)插件设置。
* 'location\_longitude'： 虽然他/她访问过的网站访客的经度。由[UserCountry](https://github.com/piwik/piwik/tree/master/plugins/UserCountry)插件设置。
* 'custom\_var\_k1': 这次访问中的第一个插槽的自定义变量名称访问自定义变量。
* 'custom\_var\_v1': 这次访问中的第一个插槽的自定义变量值访问自定义变量。
* 'custom\_var\_k2': 这次访问中的第二个插槽的自定义变量名称访问自定义变量。
* 'custom\_var\_v2': 这次访问中的第二个插槽的自定义变量值访问自定义变量。
* 'custom\_var\_k3': 这次访问的第三个插槽中的自定义变量的名称访问自定义变量。
* 'custom\_var\_v3': 这次访问的第三个插槽中的自定义变量的值访问自定义变量。
* 'custom\_var\_k4': 这次访问的第四个插槽中的自定义变量的名称访问自定义变量。
* 'custom\_var\_v4': 这次访问的第四个插槽中的自定义变量的值访问自定义变量。
* 'custom\_var\_k5': 这次访问的第五个插槽中的自定义变量的名称访问自定义变量。
* 'custom\_var\_v5': 这次访问的第五个插槽中的自定义变量的值访问自定义变量。

某些插件如[供应商](https://github.com/piwik/piwik/tree/master/plugins/Provider)插件，将添加新的信息访问。

#### 访问操作

访问还包含一个操作的列表，另一个用于每个操作的游客使访问期间。访问操作包含以下信息：

* 'server\_time'： 行动跟踪服务器的时区的日期时间
* 'idaction\_url'： 此操作 URL 操作类型的 ID
* 'idaction\_url\_ref'： URL 操作类型中访问的前一操作 ID
* 'idaction\_name'： 此操作的页面标题行动类型的 ID
* 'idaction\_name\_ref'： 页面标题行动类型中访问的前一操作 ID
* 'time\_spent\_ref\_action'： 做上一个操作所花费的时间量
* 'custom\_var\_k1'： 自定义的变量名称的第一个插槽页面自定义变量
* 'custom\_var\_v1'： 页面自定义变量的第一个插槽的自定义变量值
* 'custom\_var\_k2'： 页面自定义变量的第二个插槽的自定义变量名称
* 'custom\_var\_v2'： 自定义的变量值的第二个插槽页面自定义变量
* 'custom\_var\_k3'： 页面自定义变量的第三个插槽的自定义变量名称
* 'custom\_var\_v3'： 页面自定义变量的第三个插槽的自定义变量值
* 'custom\_var\_k4'： 自定义的变量名称的第四个插槽页面自定义变量
* 'custom\_var\_v4'： 第四槽页自定义变量的自定义的变量值
* 'custom\_var\_k5'： 自定义的变量名称的插槽页面自定义变量
* 'custom\_var\_v5'： 槽页自定义变量的自定义的变量值
* 'custom\_float': 未指定的浮点字段，通常用于保存服务器将此操作所花的时间

### 操作类型

行动类型，如特定的 URL 或页面标题，被分析访问。这种分析可以导致理解，例如，哪些页面是向游客比别人更有意义。

当 Piwik 遇到新的操作类型时，一种新的行动类型实体保持不变。

操作类型包含以下信息：

* 'name'： 一个字符串，描述操作类型。可以是一个 URL，页面标题、 活动名称或别的。含义确定的**类型**字段。
* 'hash'： 一个哈希值计算使用的名称。
* 'type'： 操作类型类别。可以是下列值之一：
  + **Piwik\Tracker\Action::TYPE\_PAGE\_URL**： 操作类型是在被跟踪的网站上的网页的 URL。
  + **Piwik\Tracker\Action::TYPE\_OUTLINK**： 操作类型是一个 URL 上被跟踪的网站的链接。访客点击它。
  + **Piwik\Tracker\Action::TYPE\_DOWNLOAD**： 操作类型是从被跟踪的网站下载的文件的 URL。
  + **Piwik\Tracker\Action::TYPE\_PAGE\_TITLE**： 操作类型是在被跟踪的网站上的网页的网页标题。
  + **Piwik\Tracker\Action::TYPE\_ECOMMERCE\_ITEM\_SKU**： 操作类型是网站上出售的是一个电子商务项目的 SKU。
  + **Piwik\Tracker\Action::TYPE\_ECOMMERCE\_ITEM\_NAME**： 操作类型是网站出售的电子商务项目的名称。
  + **Piwik\Tracker\Action::TYPE\_ECOMMERCE\_ITEM\_CATEGORY**： 操作类型是在网站使用的电子商务项目类别的名称。
  + **Piwik\Tracker\Action::TYPE\_SITE\_SEARCH**： 操作类型是网站搜索操作。
* 'url\_prefix'： 如果名称是一个 URL，这是指的 URL 前缀。前缀是从实际的 Url 中删除，因此在分析过程中忽略了一个 URL 的协议和**www.**部分。可以是下列值：
  + 0: 'http://'
  + 1: 'http://www.'
  + 2: 'https://'
  + 3: 'https://www.'

### 转换

当访问行动跟踪一个目标转换参数匹配时，转换创建并保存。转换是一个计数您的访客之一了所需的操作的统计。Piwik 将分析这些吻合结合导致他们为了帮助 Piwik 用户了解如何使他们的来访者更需采取的行动。

转换包括以下信息：

* 'idvisit': 这次访问使这种转换的 ID
* 'idsite'： 该网站的 ID 这种转换是
* 'idvisitor'： 造成这种转换的访客的 ID
* 'server\_time'： 在服务器的时区转换日期时间
* 'idaction\_url'： 导致这种转换的访问操作的 URL 操作类型的 ID
* 'idlink\_va'： 具体的 ID 访问导致在这个转换中的作用
* 'referer\_visit\_server\_date':
* 'url'： 造成这种转换要跟踪的 URL
* 'idgoal'： 此转换为目标的 ID
* 'idorder'： 如果这种转换是为电子商务订单或被遗弃的车，这将是订单的 ID
* 'items'： 如果这种转换是为电子商务订单或被遗弃的车，这将是订单/购物车中的项数
* 'revenue'： 如果这种转换是为电子商务订单或被遗弃的马车，这是该命令的收入总额
* 'revenue\_subtotal'： 如果这种转换是为电子商务订单或被遗弃的马车，这是订单/车中的商品的总成本
* 'revenue\_tax'： 如果这种转换是为电子商务订单或被遗弃的马车，这是适用于顺序/车中的商品的税金
* 'revenue\_shipping'： 如果这种转换是为电子商务订单或被遗弃的马车，这是航运的总成本
* 'revenue\_discount'： 如果这种转换是为电子商务订单或被遗弃的马车，这是适用于订单的总折扣

### 电子商务项目 （aka，转换项目）

电子商务项目是项目，是电子商务订单中销售或被遗弃在一个被遗弃的购物车。它们包括以下信息：

* 'server\_time':
* 'idorder'： 此电子商务项目的一部分的订单的 ID
* 'idaction\_sku'： 包含该项目的 SKU 的行动类型实体的 ID
* 'idaction\_name'： 包含电子商务项目名称的行动类型实体的 ID
* 'idaction\_category'： 包含这一电子商务项目的类别的行动类型实体的 ID
* 'idaction\_category2'： 包含这一电子商务项目的类别的行动类型实体的 ID
* 'idaction\_category3'： 包含这一电子商务项目的类别的行动类型实体的 ID
* 'idaction\_category4'： 包含这一电子商务项目的类别的行动类型实体的 ID
* 'idaction\_category5'： 包含这一电子商务项目的类别的行动类型实体的 ID
* 'price'： 这个个人的电子商务项目的价格
* 'quantity'： 这一项目，出现在相关的电子商务订单金额
* 'deleted'： 这一项目是否被调离该命令

## 存档数据持久性

存档数据包括**度量**和**报告**。度量是数字值，如存储。存储在[DataTable](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FDataTable)实例报告并将其保存为压缩的二进制字符串。

存档数据是与网站 ID、 期和段是随着数据的标识名称相关联。此信息查询所有的存档数据多次。目前，部分是进行哈希计算和附加到度量名称的末尾。

存档数据也保持不变与当前日期和时间所以有可能知道一些数据有多大。

所有存档数据将都包含以下信息：

* 'idarchive'： 与具有相同的网站 ID、 时期和段进行存档的存档数据的所有碎片共享 ID。
* 'name'： 报告或度量的名称。如果使用了一段，段的哈希值追加到名称。
* 'idsite'： 此存档数据是为该网站的 ID。
* 'date1'： 此存档数据为第一次约会时期。
* 'date2'： 此存档数据是期间中的最后一个日期。
* 'period'： 此存档数据是期间类型。可以是下列值之一：
  + 1：**一天**时间。
  + 2：**一周**时间。
  + 3：**月**期间。
  + 4： **年**期。
  + 5：**范围**时间。
* 'ts\_archived'： 存档数据被缓存日期时间值。
* 'value'： 一个数值 （米制） 或二进制字符串 （报告）。

## 其他数据持久性

### 网站 （aka 网站）

**网站**实体包含有关其访问跟踪网站的信息。不会有将近尽可能多的这些有互访及存档数据条目，但他们往往会受到质疑。

本报告所述的每个请求 （通过[报告的 API](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpiwiks-reporting-api)或通过 Piwik 的 UI） 查询将一个或多个站点实体。追踪器将只能查询网站数据，如果[跟踪缓存](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fall-about-tracking%23the-tracker-cache)需要更新。对于大多数的跟踪请求，网站数据，将不进行查询 （从而导致更高的性能，为追踪器）。

网站实体包含以下信息：

* 'idsite'： 该网站的唯一 ID。
* 'name'： 网站的名称。
* 'main\_url'： 主要的 URL 访问者应该使用访问本网站。
* 'ts\_created'： 网站实体保持时间和日期。
* 'ecommerce': 1如果该网站是一个电子商务网站， 0如果不是。
* 'sitesearch': 1如果站点包含内部搜索功能，0如果不是。
* 'sitesearch\_keyword\_parameters'： 查询参数该站点使用举行内部网站搜索关键字。这是一个以逗号分隔的列表。
* 'sitesearch\_category\_parameters'： 查询参数该站点使用举行内部网站搜索类别。这是一个以逗号分隔的列表。
* 'timezone'： 该网站的时区。
* 'currency'： 该网站使用的货币。唯一有效的如果该网站是一个电子商务网站。
* 'excluded\_ips'： 以逗号分隔的列表中的 IP 地址或 IP 地址范围。来自其中一个 IP 地址的访问将不会跟踪此网站。
* 'excluded\_parameters'： 查询参数的名称的逗号分隔列表。这些查询参数将从网页的 Url 删除之前访问和操作进行跟踪。
* 'excluded\_user\_agents'： 以逗号分隔的字符串列表。访问包含这些字符串中的一个用户代理将不会跟踪此网站。
* 'group':
* 'keep\_url\_fragment': 1如果 URL 片段 （之后的所有字符#） 应保持在 URL 中，当追踪行动， 0如果不是。

网站实体还包含额外可以用来访问该网站的 Url 列表。这些不存储在站点实体本身。

网站实体数据访问发生主要是通过[Piwik\Site](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FSite)类。任何不能通过该类查询可以查询通过[SitesManager](https://github.com/piwik/piwik/tree/master/plugins/SitesManager)核心插件。

#### 目标

每个站点有一个可选的目标列表。目标是网站访问者应采取所需的行动。

以下信息存储在目标实体中：

* 'idsite'： 这一目标属于该网站 ID。
* 'idgoal'： 这一目标 （只为这个网站的目标之一是唯一） ID。
* 'name'： 这一目标名称。
* 'match\_attribute'： 描述转换的目标时，应针对相匹配的请求的哪一部分的字符串。可以是下列值之一：
  + 'manually'： 目标通过[手动转换请求](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2Ftracking-javascript%23manually-trigger-a-conversion-for-a-goal)转换.
  + 'url'： 目标转换基于操作 URL 包含什么内容。
  + 'title'： 目标转换基于操作页面标题所包含的内容。
  + 'file'： 目标转换基于下载的文件的文件名包含什么内容。
  + 'external\_website'： 目标转换基于 outlink 的 URL 包含什么内容。
* 'pattern'： 要检查是否转换的目标时使用的模式。
* 'pattern\_type'： 模式匹配检查是否转换的目标时使用的类型。
  + 'contains'： 如果匹配属性包含的模式转换的目标。
  + 'exact'： 如果匹配属性等于该模式完全转换的目标。
  + 'regex'： 如果匹配属性匹配正则表达式模式与模式转换的目标。
* 'case\_sensitive': 1如果匹配应区分大小写， 0如果以其他方式。
* 'allow\_multiple': 1如果每次访问，允许使用多个转换0如果以其他方式。
* 'revenue'： 转换会生成 （如果有） 的收入的金额。
* 'deleted': 1如果这一目标已被 Piwik 用户删除0如果不是。

注： 电子商务和被遗弃的车目标是两个特殊目标具有特别的 Id。电子商务网站自动拥有了这些目标。

### 用户

用户实体描述每个 Piwik 用户除超级用户。以下信息存储在用户实体中：

* 'login'： 用户登录的句柄。
* 'password'： 用户的密码的哈希。
* 'alias'： 如果任何用户的别名。此值被显示而不是登录的句柄，当处理用户在 UI 中。
* 'email'： 用户的电子邮件地址。
* 'token\_auth'： 用户令牌验证
* 'date\_registered'： 用户数据持久化的日期。

在每次用户界面和[报告的 API](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpiwiks-reporting-api)的请求读取用户数据。< ~ — — TODO： 追踪器使用 token\_auth，这意味着它读取用户数据吗？或者该数据缓存吗？或者是只为超级用户，所以配置文件用吗？-->

还有一些用户相关的不直接在用户实体中存储的信息。他们是一般如下：

#### 用户的访问权限

用户可以允许和不允许对网站的访问。Piwik 坚持他们可以访问每个网站的每个用户的访问级别。如果他们不能访问到一个网站，然后将保存该用户 + 网站组合没有资料。

一种访问级别可以是下列值之一：

* 'view'： 用户具有查看访问权限，但不能添加目标或更改该站点的任何设置。
* 'admin'： 用户可以查看该网站的分析数据和添加目标或更改网站设置。

#### 用户语言选择

Piwik 还会坚持选择的每个用户的语言。用户登录是语言的与 （写在所选的语言，而不是语言的英语） 的名称相关联。

这种关联和持久性逻辑是由[LanguagesManager](https://github.com/piwik/piwik/tree/master/plugins/LanguagesManager)插件来实现的。

### [选项](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FOption)

选项是何地的关键是一个字符串值 （规模可能更大和可能的二进制） 的另一个字符串的键-值对。他们会询问每个 UI 和[报告的 API](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpiwiks-reporting-api)的请求。追踪器将[缓存](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fall-about-tracking%23the-tracker-cache)相关的选项值，所以只将查询选项，当缓存中需要更新的时候。

某些选项应该在每次非跟踪请求加载。这些选项有特殊**自动加载**属性设置为1.

## 数据库日志记录后端

Piwik 包括[日志实用程序](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FLog)可以用来帮助发展或疑难解答活 Piwik 安装。该实用程序可以输出到多个后端，包括数据库的日志消息。

每个日志条目包含以下信息 （所有的一切都保持不变）：

* 'tag'： 一个字符串，用于分类日志条目。这将是记录一条消息的插件的名称，或者，如果它不能被发现，类的名称。
* 'timestamp'： 当日志条目了。
* 'level'： （作为一个字符串） 日志级别。描述了该条目的严重性。级别的列表，请参阅[Piwik\Log](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FLog) 。
* 'message'： 消息该日志条目。

## 插件持久性

如果他们需要，插件可以提供为新数据的持久性。目前，因为 MySQL 是唯一支持的后台，这意味着直接添加和使用新的表。

若要将新表添加到 Piwik 的 MySQL 数据库，请执行

CREATE TABLE

插件描述符[安装](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FPlugin%23install)方法中的语句。举个例子：

use Piwik\Db;

use Piwik\Common;

use \Exception;

public class MyPlugin extends \Piwik\Plugin

{

// ...

public function install()

{

try {

$sql = "CREATE TABLE " . Common::prefixTable('mynewtable') . " (

mykey VARCHAR( 10 ) NOT NULL ,

mydata VARCHAR( 100 ) NOT NULL ,

PRIMARY KEY ( mykey )

) DEFAULT CHARSET=utf8 ";

Db::exec($sql);

} catch (Exception $e) {

// ignore error if table already exists (1050 code is for 'table already exists')

if (!Db::get()->isErrNo($e, '1050')) {

throw $e;

}

}

}

// ...

}

插件应该也去清理后自己的[卸载](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FPlugin%23uninstall)方法中删除的表：

use Piwik\Db;

use Piwik\Common;

use \Exception;

public class MyPlugin extends \Piwik\Plugin

{

// ...

public function uninstall()

{

Db::dropTables(Common::prefixTable('mynewtable'));

}

// ...

}

**注意： 新表应是适当**[**的前缀**](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FCommon%23prefixtable)**.**

### 扩大现有的表

插件也可以增强现有的表。如果，例如，插件想要跟踪额外的访问信息，该插件可以添加日志数据的表，并在跟踪过程中设置这些列的列。这也可以在[安装](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FPlugin%23install)方法：

use Piwik\Db;

public class MyPlugin extends \Piwik\Plugin

{

// ...

public function install()

{

try {

$q1 = "ALTER TABLE `" . Common::prefixTable("log\_visit") . "`

ADD `mynewdata` VARCHAR( 100 ) CHARACTER SET utf8 COLLATE utf8\_general\_ci NULL DEFAULT NULL AFTER `config\_os`,";

Db::exec($q1);

} catch (Exception $e) {

// ignore column already exists error

if (!Db::get()->isErrNo($e, '1060')) {

throw $e;

}

}

}

// ...

}

插件应该[卸载](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fapi-reference%2FPiwik%2FPlugin%23uninstall)方法，**除非做这么花很长一段时间**中删除该列。由于日志表可以有数以百万计和甚至数十亿的条目，从这些表中删除列，卸载插件时将一个糟糕的主意。

## MySQL 后端

本节列出了每个 MySQL 表用来存储上文所述的数据和细节我们给他们使 Piwik 尽可能快地运行所做的一切。

注： 在 MySQL 数据库中的所有表名都前缀中的值

*[database] tables\_prefix*

INI 配置。这是为了确保非 piwik 表不要得到覆盖或偶然使用。它还使它可能在一个数据库中存储的 Piwik 多个实例。

### 日志数据的表

#### log\_visit

此表存储[访问实体](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23log-data-persistence-visits).

CREATE TABLE

此表的 SQL 是：

CREATE TABLE log\_visit (

idvisit INTEGER(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

idsite INTEGER(10) UNSIGNED NOT NULL,

idvisitor BINARY(8) NOT NULL,

visitor\_localtime TIME NOT NULL,

visitor\_returning TINYINT(1) NOT NULL,

visitor\_count\_visits SMALLINT(5) UNSIGNED NOT NULL,

visitor\_days\_since\_last SMALLINT(5) UNSIGNED NOT NULL,

visitor\_days\_since\_order SMALLINT(5) UNSIGNED NOT NULL,

visitor\_days\_since\_first SMALLINT(5) UNSIGNED NOT NULL,

visit\_first\_action\_time DATETIME NOT NULL,

visit\_last\_action\_time DATETIME NOT NULL,

visit\_exit\_idaction\_url INTEGER(11) UNSIGNED NULL DEFAULT 0,

visit\_exit\_idaction\_name INTEGER(11) UNSIGNED NOT NULL,

visit\_entry\_idaction\_url INTEGER(11) UNSIGNED NOT NULL,

visit\_entry\_idaction\_name INTEGER(11) UNSIGNED NOT NULL,

visit\_total\_actions SMALLINT(5) UNSIGNED NOT NULL,

visit\_total\_searches SMALLINT(5) UNSIGNED NOT NULL,

visit\_total\_events SMALLINT(5) UNSIGNED NOT NULL,

visit\_total\_time SMALLINT(5) UNSIGNED NOT NULL,

visit\_goal\_converted TINYINT(1) NOT NULL,

visit\_goal\_buyer TINYINT(1) NOT NULL,

referer\_type TINYINT(1) UNSIGNED NULL,

referer\_name VARCHAR(70) NULL,

referer\_url TEXT NOT NULL,

referer\_keyword VARCHAR(255) NULL,

config\_id BINARY(8) NOT NULL,

config\_os CHAR(3) NOT NULL,

config\_browser\_name VARCHAR(10) NOT NULL,

config\_browser\_version VARCHAR(20) NOT NULL,

config\_resolution VARCHAR(9) NOT NULL,

config\_pdf TINYINT(1) NOT NULL,

config\_flash TINYINT(1) NOT NULL,

config\_java TINYINT(1) NOT NULL,

config\_director TINYINT(1) NOT NULL,

config\_quicktime TINYINT(1) NOT NULL,

config\_realplayer TINYINT(1) NOT NULL,

config\_windowsmedia TINYINT(1) NOT NULL,

config\_gears TINYINT(1) NOT NULL,

config\_silverlight TINYINT(1) NOT NULL,

config\_cookie TINYINT(1) NOT NULL,

location\_ip VARBINARY(16) NOT NULL,

location\_browser\_lang VARCHAR(20) NOT NULL,

location\_country CHAR(3) NOT NULL,

location\_region char(2) DEFAULT NULL,

location\_city varchar(255) DEFAULT NULL,

location\_latitude float(10, 6) DEFAULT NULL,

location\_longitude float(10, 6) DEFAULT NULL,

custom\_var\_k1 VARCHAR(200) DEFAULT NULL,

custom\_var\_v1 VARCHAR(200) DEFAULT NULL,

custom\_var\_k2 VARCHAR(200) DEFAULT NULL,

custom\_var\_v2 VARCHAR(200) DEFAULT NULL,

custom\_var\_k3 VARCHAR(200) DEFAULT NULL,

custom\_var\_v3 VARCHAR(200) DEFAULT NULL,

custom\_var\_k4 VARCHAR(200) DEFAULT NULL,

custom\_var\_v4 VARCHAR(200) DEFAULT NULL,

custom\_var\_k5 VARCHAR(200) DEFAULT NULL,

custom\_var\_v5 VARCHAR(200) DEFAULT NULL,

PRIMARY KEY(idvisit),

INDEX index\_idsite\_config\_datetime (idsite, config\_id, visit\_last\_action\_time),

INDEX index\_idsite\_datetime (idsite, visit\_last\_action\_time),

INDEX index\_idsite\_idvisitor (idsite, idvisitor)

) DEFAULT CHARSET=utf8;

**Index\_idsite\_config\_datetime**指数用于当试图识别回访访客。

聚合访问时，将使用**index\_idsite\_datetime**索引。由于日志聚合才会发生个别日时期，此索引有助于 Piwik 快速查找网站和期间访问。如果没有它，日志聚合将需要通过整个 log\_visit 表的表扫描。

#### log\_link\_visit\_action

此表存储[访问行动实体](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23log-data-persistence-visit-actions).

CREATE TABLE

此表的 SQL 是：

CREATE TABLE log\_link\_visit\_action (

idlink\_va INTEGER(11) UNSIGNED NOT NULL AUTO\_INCREMENT,

idsite int(10) UNSIGNED NOT NULL,

idvisitor BINARY(8) NOT NULL,

server\_time DATETIME NOT NULL,

idvisit INTEGER(10) UNSIGNED NOT NULL,

idaction\_url INTEGER(10) UNSIGNED DEFAULT NULL,

idaction\_url\_ref INTEGER(10) UNSIGNED NULL DEFAULT 0,

idaction\_name INTEGER(10) UNSIGNED,

idaction\_name\_ref INTEGER(10) UNSIGNED NOT NULL,

idaction\_event\_category INTEGER(10) UNSIGNED DEFAULT NULL,

idaction\_event\_action INTEGER(10) UNSIGNED DEFAULT NULL,

time\_spent\_ref\_action INTEGER(10) UNSIGNED NOT NULL,

custom\_var\_k1 VARCHAR(200) DEFAULT NULL,

custom\_var\_v1 VARCHAR(200) DEFAULT NULL,

custom\_var\_k2 VARCHAR(200) DEFAULT NULL,

custom\_var\_v2 VARCHAR(200) DEFAULT NULL,

custom\_var\_k3 VARCHAR(200) DEFAULT NULL,

custom\_var\_v3 VARCHAR(200) DEFAULT NULL,

custom\_var\_k4 VARCHAR(200) DEFAULT NULL,

custom\_var\_v4 VARCHAR(200) DEFAULT NULL,

custom\_var\_k5 VARCHAR(200) DEFAULT NULL,

custom\_var\_v5 VARCHAR(200) DEFAULT NULL,

custom\_float FLOAT NULL DEFAULT NULL,

PRIMARY KEY(idlink\_va),

INDEX index\_idvisit(idvisit),

INDEX index\_idsite\_servertime ( idsite, server\_time )

) DEFAULT CHARSET=utf8

idsite和idvisitor,为了避免 log\_visit 表联接在某些情况下从访问行动相关联的访问复制列。

**Index\_idvisit**索引允许 Piwik 快速访问查询的访问操作。

聚合访问操作时，将使用**index\_idsite\_servertime**指数。它特定期间允许快速访问被跟踪的访问操作用于一个特定的网站，让我们避免通过整个表的表扫描。

#### log\_action

此表存储[操作类型实体](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23log-data-persistence-action-types).

CREATE TABLE

此表的 SQL 是：

CREATE TABLE log\_action (

idaction INTEGER(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

name TEXT,

hash INTEGER(10) UNSIGNED NOT NULL,

type TINYINT UNSIGNED NULL,

url\_prefix TINYINT(2) NULL,

PRIMARY KEY(idaction),

INDEX index\_type\_hash (type, hash)

) DEFAULT CHARSET=utf8

**Index\_type\_hash**指数在跟踪期间用于查找现有的行动类型。

#### log\_conversion

此表存储[转换实体](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23log-data-persistence-conversions).

CREATE TABLE

此表的 SQL 是：

CREATE TABLE `log\_conversion` (

idvisit int(10) unsigned NOT NULL,

idsite int(10) unsigned NOT NULL,

idvisitor BINARY(8) NOT NULL,

server\_time datetime NOT NULL,

idaction\_url int(11) default NULL,

idlink\_va int(11) default NULL,

referer\_visit\_server\_date date default NULL,

referer\_type int(10) unsigned default NULL,

referer\_name varchar(70) default NULL,

referer\_keyword varchar(255) default NULL,

visitor\_returning tinyint(1) NOT NULL,

visitor\_count\_visits SMALLINT(5) UNSIGNED NOT NULL,

visitor\_days\_since\_first SMALLINT(5) UNSIGNED NOT NULL,

visitor\_days\_since\_order SMALLINT(5) UNSIGNED NOT NULL,

location\_country char(3) NOT NULL,

location\_region char(2) DEFAULT NULL,

location\_city varchar(255) DEFAULT NULL,

location\_latitude float(10, 6) DEFAULT NULL,

location\_longitude float(10, 6) DEFAULT NULL,

url text NOT NULL,

idgoal int(10) NOT NULL,

buster int unsigned NOT NULL,

idorder varchar(100) default NULL,

items SMALLINT UNSIGNED DEFAULT NULL,

revenue float default NULL,

revenue\_subtotal float default NULL,

revenue\_tax float default NULL,

revenue\_shipping float default NULL,

revenue\_discount float default NULL,

custom\_var\_k1 VARCHAR(200) DEFAULT NULL,

custom\_var\_v1 VARCHAR(200) DEFAULT NULL,

custom\_var\_k2 VARCHAR(200) DEFAULT NULL,

custom\_var\_v2 VARCHAR(200) DEFAULT NULL,

custom\_var\_k3 VARCHAR(200) DEFAULT NULL,

custom\_var\_v3 VARCHAR(200) DEFAULT NULL,

custom\_var\_k4 VARCHAR(200) DEFAULT NULL,

custom\_var\_v4 VARCHAR(200) DEFAULT NULL,

custom\_var\_k5 VARCHAR(200) DEFAULT NULL,

custom\_var\_v5 VARCHAR(200) DEFAULT NULL,

PRIMARY KEY (idvisit, idgoal, buster),

UNIQUE KEY unique\_idsite\_idorder (idsite, idorder),

INDEX index\_idsite\_datetime ( idsite, server\_time )

) DEFAULT CHARSET=utf8

这并不会转换实体上面列出的表中存储的所有额外信息访问实体这种转换是从被复制。这可以使我们避免在某些情况下加入 log\_visit 表。

聚合转换时，将使用**index\_idsite\_datetime**索引。它特定期间允许快速访问到被跟踪的转换为一个特定的网站，并让我们避免通过整个表的表扫描。

#### log\_conversion\_item

此表存储[电子商务项目实体](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23log-data-persistence-ecommerce-items).

CREATE TABLE

此表的 SQL 是：

CREATE TABLE `log\_conversion\_item` (

idsite int(10) UNSIGNED NOT NULL,

idvisitor BINARY(8) NOT NULL,

server\_time DATETIME NOT NULL,

idvisit INTEGER(10) UNSIGNED NOT NULL,

idorder varchar(100) NOT NULL,

idaction\_sku INTEGER(10) UNSIGNED NOT NULL,

idaction\_name INTEGER(10) UNSIGNED NOT NULL,

idaction\_category INTEGER(10) UNSIGNED NOT NULL,

idaction\_category2 INTEGER(10) UNSIGNED NOT NULL,

idaction\_category3 INTEGER(10) UNSIGNED NOT NULL,

idaction\_category4 INTEGER(10) UNSIGNED NOT NULL,

idaction\_category5 INTEGER(10) UNSIGNED NOT NULL,

price FLOAT NOT NULL,

quantity INTEGER(10) UNSIGNED NOT NULL,

deleted TINYINT(1) UNSIGNED NOT NULL,

PRIMARY KEY(idvisit, idorder, idaction\_sku),

INDEX index\_idsite\_servertime ( idsite, server\_time )

) DEFAULT CHARSET=utf8

idsite, idvisitor, server\_time和idvisit是从这一电子商务项目属于转换实体复制列。他们被复制，所以我们可以不必联接其他表聚合电子商务项目。

聚合电子商务项目时，将使用**index\_idsite\_servertime**索引。它允许快速访问被跟踪的项目为一个特定的网站和特定期间，让我们避免通过整个表的表扫描。

### 存档表

在 MySQL 中后端存档数据按月存档数据为分区。所以报告聚合访问从 2012 年 1 月，将在不同表中报告聚合访问从 2012 年 2 月举行。

Piwik 创建两种类型的存档表，另一个用于每种类型的存档数据。**Archive\_numeric**表存储度量数据和**archive\_blob**表中存储的报告数据。

存档表是动态创建的。当 Piwik 需要查询或插入的某一个月的存档数据，它不能找到包含此数据的表时，创建表。

年份和月份存档表的名称后缀作为附加。所以 2012 年 1 月的**archive\_numeric**表将有名称**archive\_numeric\_2012\_01**.

#### archive\_numeric

CREATE TABLE

此表的 SQL 是：

CREATE TABLE archive\_numeric\_YYYY\_MM (

idarchive INTEGER UNSIGNED NOT NULL,

name VARCHAR(255) NOT NULL,

idsite INTEGER UNSIGNED NULL,

date1 DATE NULL,

date2 DATE NULL,

period TINYINT UNSIGNED NULL,

ts\_archived DATETIME NULL,

value DOUBLE NULL,

PRIMARY KEY(idarchive, name),

INDEX index\_idsite\_dates\_period(idsite, date1, date2, period, ts\_archived),

INDEX index\_period\_archived(period, ts\_archived)

) DEFAULT CHARSET=utf8

在查询存档数据时，将使用**index\_idsite\_dates\_period**指数。它可以让 Piwik 快速查询任何网站及期间，和过去的某一日期-时间归档的数据的存档数据。

用**index\_period\_archived**的索引时[清除存档数据](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fpiwik.org%2Fdocs%2Fmanaging-your-databases-size%2F)。它允许 Piwik 是足够老，被清除的特定时间内快速查找存档数据。

#### archive\_blob

CREATE TABLE

此表的 SQL 是：

CREATE TABLE archive\_blob (

idarchive INTEGER UNSIGNED NOT NULL,

name VARCHAR(255) NOT NULL,

idsite INTEGER UNSIGNED NULL,

date1 DATE NULL,

date2 DATE NULL,

period TINYINT UNSIGNED NULL,

ts\_archived DATETIME NULL,

value MEDIUMBLOB NULL,

PRIMARY KEY(idarchive, name),

INDEX index\_period\_archived(period, ts\_archived)

) DEFAULT CHARSET=utf8

在一个**archive\_numeric**表中相同的方式使用**index\_period\_archived**索引。

**archive\_blob**表没有索引，使查询的网站、 句点和存档的日期行快速。这是因为他们不应该质疑这种方式。相反，应该质疑**archive\_numeric**表和

idarchive

保存的值。到**archive\_blob**表中查询数据，可以使用这些值。

### 其他表

#### 网站

此表存储[网站](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23other-data-site)实体。

CREATE TABLE

此表的 SQL 是：

CREATE TABLE site (

idsite INTEGER(10) UNSIGNED NOT NULL AUTO\_INCREMENT,

name VARCHAR(90) NOT NULL,

main\_url VARCHAR(255) NOT NULL,

ts\_created TIMESTAMP NULL,

ecommerce TINYINT DEFAULT 0,

sitesearch TINYINT DEFAULT 1,

sitesearch\_keyword\_parameters TEXT NOT NULL,

sitesearch\_category\_parameters TEXT NOT NULL,

timezone VARCHAR( 50 ) NOT NULL,

currency CHAR( 3 ) NOT NULL,

excluded\_ips TEXT NOT NULL,

excluded\_parameters TEXT NOT NULL,

excluded\_user\_agents TEXT NOT NULL,

`group` VARCHAR(250) NOT NULL,

keep\_url\_fragment TINYINT NOT NULL DEFAULT 0,

PRIMARY KEY(idsite)

) DEFAULT CHARSET=utf8

##### site\_url

此表存储额外的 Url 供[网站](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23other-data-site)实体。

CREATE TABLE

此表的 SQL 是：

CREATE TABLE site\_url (

idsite INTEGER(10) UNSIGNED NOT NULL,

url VARCHAR(255) NOT NULL,

PRIMARY KEY(idsite, url)

) DEFAULT CHARSET=utf8

#### 目标

此表存储[目标](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23other-data-goal)实体。

CREATE TABLE

此表的 SQL 是：

CREATE TABLE `goal` (

`idsite` int(11) NOT NULL,

`idgoal` int(11) NOT NULL,

`name` varchar(50) NOT NULL,

`match\_attribute` varchar(20) NOT NULL,

`pattern` varchar(255) NOT NULL,

`pattern\_type` varchar(10) NOT NULL,

`case\_sensitive` tinyint(4) NOT NULL,

`allow\_multiple` tinyint(4) NOT NULL,

`revenue` float NOT NULL,

`deleted` tinyint(4) NOT NULL default '0',

PRIMARY KEY (`idsite`,`idgoal`)

) DEFAULT CHARSET=utf8

#### 用户

此表存储[用户](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23other-data-user)实体。

CREATE TABLE

此表的 SQL 是：

CREATE TABLE user (

login VARCHAR(100) NOT NULL,

password CHAR(32) NOT NULL,

alias VARCHAR(45) NOT NULL,

email VARCHAR(100) NOT NULL,

token\_auth CHAR(32) NOT NULL,

date\_registered TIMESTAMP NULL,

PRIMARY KEY(login),

UNIQUE KEY uniq\_keytoken(token\_auth)

) DEFAULT CHARSET=utf8

##### 访问

此表存储[用户访问信息](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23other-data-user-access).

CREATE TABLE

此表的 SQL 是：

CREATE TABLE access (

login VARCHAR(100) NOT NULL,

idsite INTEGER UNSIGNED NOT NULL,

access VARCHAR(10) NULL,

PRIMARY KEY(login, idsite)

) DEFAULT CHARSET=utf8

##### user\_language

此表存储[用户语言选择信息](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23other-data-user-language-choice).

CREATE TABLE

此表的 SQL 是：

CREATE TABLE user\_language (

login VARCHAR( 100 ) NOT NULL ,

language VARCHAR( 10 ) NOT NULL ,

PRIMARY KEY ( login )

) DEFAULT CHARSET=utf8

这张桌子是由[LanguagesManager](https://github.com/piwik/piwik/tree/master/plugins/LanguagesManager)插件创建的。

#### 选项

此表存储[选项](http://www.microsofttranslator.com/bv.aspx?from=en&to=zh-CHS&a=http%3A%2F%2Fdeveloper.piwik.org%2Fguides%2Fpersistence-and-the-mysql-backend%23other-data-options)数据。

CREATE TABLE

此表的 SQL 是：

CREATE TABLE `option` (

option\_name VARCHAR( 255 ) NOT NULL,

option\_value LONGTEXT NOT NULL,

autoload TINYINT NOT NULL DEFAULT '1',

PRIMARY KEY ( option\_name ),

INDEX autoload( autoload )

) DEFAULT CHARSET=utf8

#### logger\_message

由数据库日志记录后端使用此表。

CREATE TABLE

此表的 SQL 是：

CREATE TABLE logger\_message (

idlogger\_message INTEGER UNSIGNED NOT NULL AUTO\_INCREMENT,

tag VARCHAR(50) NULL,

timestamp TIMESTAMP NULL,

level VARCHAR(16) NULL,

message TEXT NULL,

PRIMARY KEY(idlogger\_message)

) DEFAULT CHARSET=utf8

#### 会议

这张桌子不存储实体数据。Piwik 由用于存储会话数据 （如使用基于文件的会话的替代方法）。

CREATE TABLE

此表的 SQL 是：

CREATE TABLE session (

id CHAR(32) NOT NULL,

modified INTEGER,

lifetime INTEGER,

data TEXT,

PRIMARY KEY ( id )

) DEFAULT CHARSET=utf8

## 其他后端

目前 MySQL 后端是唯一可用的后端。使用 MySQL 被遍布 Piwik，一刻也不太可能创建其他后端。

说，使用其他执著的解决方案是在我们的 TODO 列表。Piwik 最终将工作与不同的关系型数据库和不同 NoSQL 解决方案。