**Google Analytics Module**

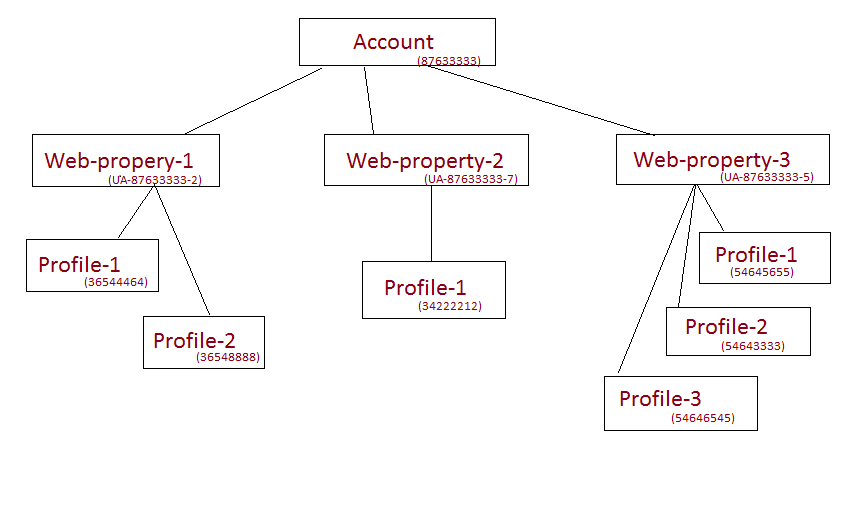
**Introduction**:

Google Analytics is a service provided by Google which keeps record of traffic on a site. This module provide the same facility (tracking the traffic on the site) to user in a user-friendly way. We are using Google API to make use of this facility. Google API is a tool which acts as a mediator between the application and Google services.

**Brief structure of accounts in Google Analytics:**

There are three layers in any Google Analytics account- Account, Web-property, and Profile.

1. Account: It is the top-most layer in this structure. To maintain uniqueness, a unique ID (numeric value) is assigned to it means no two accounts can have same ID.
2. Web-property: It is the second layer in the structure. We can call it as the child of ‘Account’. Every web-property has a unique ID (varchar value). There is a similarity between ID of Account and web-property. Every child web-property is inheriting their parent account’s ID as numeric value in their ID. E.g., Account ID = 65767898 then its web-property must be UA-65767898-xx. All the web-properties of an account will be having this numeric value in between of their own ID. A web-property is also a domain or website for which one can activate Google Analytics service. Activating Google Analytic service means allowing Google to track traffic on their site.
3. Profile: It is the third layer in the structure. We can call it as child of ‘Web-property’ and grand-child of ‘Account’. No two profiles share the same ID (numeric value).



In this way, there can be n-number of accounts and for those ‘n’ accounts we can have their sub-levels.

**Google API** uses oAuth protocol to authenticate the user against his Google account. **oAuth protocol** is a secure protocol that enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on its own behalf.

**Flow of module:**

1. User log-in to Domainz.se
2. On domain-list page, user can add domains by clicking on the ‘Add Domain’ domain button. In the same button row, user will find a button called as ‘Authenticate their domains’. This button shows that user can authenticate for their Google Analytical domains.
3. When user clicks on this button, o-Auth process starts.
4. O-Auth process includes authentication of user against his Google or Gmail account and ask user to allow to Domainz.se to access their private information, in our case we will ask user to allow us to access their Google Analytics information.
5. If user denies giving access then we redirect user to domain-list page by displaying message stated as ‘You have denied giving access to Domainz.se’.
6. When user allows then we fetch all the accounts, web-properties and profile information from his analytics account.
7. Now profile-selection module begins where user selects and maps his domains registered on Domainz.se and domains registered Google account. We have provided check-box to select the domains for mapping. When user selects domains and clicks on Import button then we suggest user the perfect match for the domain(s).
8. This process includes various cases like if an already authenticated user has added new domains or deleted previous ones or added new profiles or deleted previously used profiles then we alert user to synchronize these profiles on Google and Domainz.se in order to get correct information.
9. After profile selection, we fetch all the analytic information related to those profiles from Google and save the data in database.
10. The representation of analytical data is in one of the tab called ‘Analytical Data’ in the accordion for every domain on domain-list page. The tab contains analytical information of previous day. It also consists of a button called ‘Get More Info!’ which triggers to open a pop-up.
11. The pop-up contains all the tabs present in the accordion. ‘Google Analytics’ tab contains analytic information of that domain, by default we are showing previous day’s information. A date picker is also provided to get the data for a date-range. Date-picker is visible only on the Analytics tab.
12. We have used High-Charts API to display statistical data through charts and graphs.
13. For fetching analytics information every day, a cron-job runs that fetches information for every domain of an authenticated user.
14. A simultaneous cron-script function runs to check for any updates or modification if user has made in his Google Analytic account regarding the adding and deleting profiles or accounts.

**Initial set-up for application in API console:**

**Steps:**

1. Go through the link <https://code.google.com/apis/console/> . For using API, first we need to register our web application in Google API console.
2. During registration we can select the service(s) we need to activate for the application.
3. After successful registration, console generates a Client-id, secret key, API key, redirect URI (URI where API will redirect user after oAuth process, in our case we are redirecting user to profile-selection page).

**API integration and set-up:**

**Steps:**

1. Download Google API for PHP application from the link <http://code.google.com/p/google-api-php-client/downloads/list> .
2. Upload the unzipped folder on the root of web application.
3. Edit the configuration file (path: src/config.php) of API by entering information like Client ID, Developers key (or API key), Redirect URI, and Client secret ID.
4. Now your API is completely ready to work for a web application.

**Files used in module:**

1. **GoogleAuth.php:** used to get the authentication link of API on domain-list page.
2. **AuthPage.php**: used as a landing page after redirecting from API on our application. The page contains all the profiles of GA account.
3. **GaData.php:** an intermediate file called via AJAX request. We are using API and prepare business logic and rules to get data from Google and store into our database.
4. **GoogleDataForDomainsHandler.php:** used to interact with database for fetching, storing and modifying analytical data.
5. **bp-domainlist-core.php**: all the functions defined in database interaction file (GoogleDataForDomainsHandler.php) are called here and the functions of this file are called in our other files.
6. **suggestion**.**php**: used to render the suggested list to user during profile selection process.
7. **view**-**report.php:** used for showing contain in the pop-up.
8. **my**-**domains.php:** used to show analytical data in accordion(upper).
9. **ajax-response.php:** used toshow data in accordion(upper) since it is called via AJAX request when user clicks on ‘See All’ button.
10. **custom\_domain\_core.js:** it renders accordion (lower accordion).
11. **custom\_domain.js:** it renders accordion when user drags and drops the domain in group.
12. **google-common.js:** contains all the function and events called for this module.
13. **view-report.js:** contains all the functions called for pop-up.
14. **google-charts.js:** contains functions and chart options definition for High-Charts API.
15. **google-common.css:** styles used for profile-selection page and authentication link.
16. **view-report.css:** styles used for pop-up window.

**Process-wise description of module:**

1. **Authentication process**: We can call this process as oAuth process because we are using oAuth 2.0 protocols to authenticate user for his Google account. On domain-list page, when user clicks on button called as ‘Authenticate your domains’ then API begins the interaction between our application and Google. User is asked to enter their login credentials for their Google (or Gmail) account. When credentials are valid then API asks user to allow Domainz.se to access his GA data. If user permits, then redirecting user to the redirect URI (which we have defined in the Google console while registering the application) with a valid and unique key in the URL and then API sets an access token. If user denies giving access then also API will redirect user to the same redirect URI by providing an error message.

**Access token** is a token/secret key that is being used by our application to access the data of the service for which we have asked to (in our case the Google service is Analytic information). User authenticates for one time only.

**Scripts:** GoogleAuth.php, AuthPage.php

1. **Profile-selection process:** This is the process where we bring user a number of times. Initially after successful user authentication, this is the next important step of this module for bringing the analytical data from Google. In this process, we bring all the profiles and web-properties from user’s Google account where we are providing two options to user.

a)      User can map Google web-properties to the domains he has registered on our site. When user selects the properties to be mapped then we give a suggestion list to user. If user approves then we save the web-propertyID and profileID into our database, then fetch the analytical data from Google and save in database.

b)     User can add all the web-properties to our site. In this, we add all the domains to our site and save the web-propertyID and profileID into the database and save the respective the analytical data.

**Scripts:**Auth-Page.php, bp-domainlist-core.php, GoogleDataHandler.php, suggestion .php, google-common.js

1. **Fetch Analytical data from Google:** According to the profiles selected by user, we fetch data for those profiles and save the data into our database. As we update our database with the new data from Google we are running a cronjob. We have scheduled cron on the first minute of every day. So that we fetch all the data of a day passed before today (means yesterday’s data). There are some special metrics and dimensions defined by Google code developers that will be used to get analytical data.

**Scripts:** GaData.php

1. **Different cronjobs scheduled for this module:** We have scheduled two cronjobs in this module. One is for fetching analytical data from user’s Google account and save it into our database. Second one is scheduled to check whether user has made any modifications into their Google account. Modification means if user has added a new web-property (or domain) **OR** added a new profile in an existing web-property **OR** deleted any web-property **OR** deleted any profile. **IF** any updates found then we notify user by showing a notice on domain-list page.

**Script:** analyticscron.php, AnalyticsCron.php

1. **Showing analytical data in accordion:** We have handling three cases while showing data in accordion. One case can be that a user has yet not authenticated him against his Google account. Second case can be user has been authenticated but yet not selected or matched any profile for the respective domain. Third one can be user is authenticated and data is visible to user for that particular domain. By default we are showing yesterday’s data as per client’s requirement. For more details we are giving an action button in the bottom of the details exactly inside the accordion. When user clicks on the action button we are opening a pop-up window.

**Scripts:** my-domains.php, ajax-reposnce.php, custom-domainlist-core.js, custom-domain.js

1. **Content in pop-up window:** The significance of pop-up window is to take all details in that window itself. There is a date-picker provided specifically for analytics information bar as per client’s requirement so that user can see such details for a particular date range. We are showing various charts in pop-up window e.g. visits from various countries, various browser user with their versions and various operating systems with their versions.

**Scripts:** view-report.php, view-report.js, google-chart.js, analytical –ajax.php

1. **Update profiles on site:** As we are running a cronjob for finding any updates regarding profiles in user’s Google Analytic account. If any modification found then put a notice on domain-list page and provide a link to user to get onto that page and made appropriate changes to the account he has in our site. User can make modifications either by making new or deleting previous profile(s) or web-property(s).
2. If user has added a new web-property then we list all the web properties in tabulated form. We have provided a check-box before every property so that user can check and add the selected ones to [domainz.se](http://domainz.se/). In this procedure, we add the property to user account and consequently fetch its respective data from Google.
3. If user has added a new profile to any of his property then we updates the profiles for that domain in our database and fetches new data for that domain from Google.
4. If user has deleted any property from Google and that property is added onto our site then we recommend user to delete it from our site also. When user clicks on the ‘Delete’ button we delete the record. We are deleting all the analytical records we have in the data i.e. from tables ‘UsersAnalyitcalData’ and ‘UsersTrafficData’.
5. If user has deleted any of a profile for a domain means if a domain has two profiles and user has deleted either of them then we recommend user to delete update their profile information onto our site also. When user clicks on ‘Delete’ button we update the row and delete useless profile and keep the one which is alive in user’s GA account. We are deleting all the analytical records we have in the data i.e. from tables ‘UsersAnalyitcalData’ and ‘UsersTrafficData’.

**Scripts:** AuthPage.php, bp-domainlist-core.php, GoogleDataForDomainsHandler.php

1. **Multiple-profile synchronization:** As we know Google Analytics serves the facility of creating various accounts or web-properties for any user’s account. It also provides to create multiple profiles in a single domain or account. So, we are fetching analytic details for both the profiles of an account. In the pop-up window, we are displaying profile records one below the other.

**Functions defined in database interaction file:**

1. **update\_refresh\_token**
2. **get\_refresh\_token**
3. **update\_matched\_domains**
4. **add\_profiles\_for\_domains**
5. **get\_google\_profiles**
6. **insert\_analytical\_data**
7. **insert\_analytical\_traffic\_data**
8. **get\_yesterday\_visits**
9. **get\_analytical\_data**
10. **get\_analytical\_data\_for\_domain**
11. **get\_analytical\_data\_for\_domain\_rowwise**
12. **get\_traffic\_data**
13. **get\_traffic\_data\_for\_domain**
14. **getCommentsOnDomain**
15. **get\_keyrank\_for\_domain**
16. **getUserAlerts**
17. **delete\_google\_profiles**
18. **arrangeBrowsers**
19. **arrangeCountries**
20. **arrangeProviders**
21. **arrangeOS**

**Flow chart:**

User can create list of his choice manually

Save the mapped profiles into our DB and fetch the analytical data from GA account of user using API.

**Flow Diagram for update Google profile:**

**1.**

**2.**