Tracking users in OpenACS/dotLRN
David Arroyo Menéndez
Félix Hernández del Olmno
Raul Morales Hidalgo

In this paper we are covering the available possibilities of tracking user activities in OpenACS/dotLRN. Currently, we have user-tracking and views which are going to be described among a new project: tracker.

1. Introduction

Many users would like to have the possibility to know things like the number of visits to a forum posts, who has done it, the user interaction within a course, which is the real usage of a course, etc. To do so we need a general way to store user interaction within OpenACS/dotLRN.

2. Description of possibilities

User interaction is measured by HTTP requests by the user which are stored in the server’s access log, so a possibility is to extract information from the access log, which is what user tracking does.

The problems of such approximation are:
• Retrieving information from the logs is slow and not quite declarative.
• Retrieving information from the logs is not integrated within OpenACS context
• Text files can’t guarantee ACID

Another approach is to store every webserver request in tables like this:
create table tracker_visits ( 
    visit_id integer primary key, --integer references acs_objects(object_id) primary key, 
    visit_date timestamp, 
    url varchar, 
    page_type char(1) default ('d') check (page_type in ('d', 's')), 
    request integer, 
    session_id integer, --references tracker_sessions(session_id) on delete cascade, 
    user_id integer references users(user_id) on delete cascade, 
    object_id integer references acs_objects(object_id) on delete cascade 
); 

create table tracker_sessions ( 
    session_id integer, 
    user_id integer references users(user_id) on delete cascade, 
    ip varchar, 
    ip_reverse varchar, 
    user_agent varchar, 
    primary key (session_id, user_id) 
); 

create table tracker_parameters ( 
    visit_id integer primary key references tracker_visits(visit_id) on delete cascade, 
    name varchar, 
    value varchar 
); 

Because this information is now stored in a DB, we have ACID, a more standard approach to OpenACS integration and a more declarative way to get information: SQL. This is the tracker package approach.

There is a third available approach, which is using views package. This approximation gives us the visits to an object in a very simple way, using the views table.

create table views ( 
    object_id integer, 
    constraint views_object_id_fk references acs_objects(object_id) on delete cascade 
    not null, 
    viewer_id integer, 
    constraint views_owner_id_fk references parties(party_id) on delete cascade 
    not null, 
    views integer default 1, 
    last_viewed timestamptz default now() 
);
The pro of this approach is that it gives a bigger semantic power, on the other hand it doesn’t store information in the best way. When recovering information from access logs or from tracker approach you’ve true information, but when you recover information from views you’ve interpreted information.

The new rows is inserted in views table calling to views::record_view, for instance, we want know if an user have visited a forum post we do calls like:

```
set views [views::record_view -object_id $message(message_id) -viewer_id $viewer_id]
```

Someone can do questions like if the user is visiting a thread, then is he visiting all messages or only the new messages? what happen with the messages read by a notification?

When the developer is using views, the developer take decissions about of a click’s meaning that he won’t can undo in the future

### 3. Conclusions

We can read a summary with pros and cons of every approach:

#### Table 1. Pros and Cons

<table>
<thead>
<tr>
<th>Server Log</th>
<th>Tracker</th>
<th>Views</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text files can’t guarantee ACID</td>
<td>Can guarantee ACID</td>
<td>Can guarantee ACID</td>
</tr>
<tr>
<td>Can’t use SQL to retrieve information</td>
<td>Can use SQL to retrieve information</td>
<td>Can use SQL to retrieve information</td>
</tr>
<tr>
<td>Can’t answer to the question who has seen what object in an easy way.</td>
<td>Can’t answer to the question who has seen what object in an easy way.</td>
<td>Can answer to the question who has seen what object in an easy way.</td>
</tr>
<tr>
<td>Many information is storaged in flat files</td>
<td>All dialog between the browser and the server is storaged in a database</td>
<td>Only the semantic information is storaged in a database.</td>
</tr>
</tbody>
</table>
Perhaps to can take a decision about what approach is better would be interesting know what is the cost to add a semantic from the dialog between server and browser. It would be a big work, but we can an intuition about how to do it viewing some queries from tracker

```sql
select person__name(user_id) as name, count(visit_id) as visits
from tracker_visits
where object_id=23036 group by user_id order by visits desc
```

```sql
select count(*) as num_visits
from tracker_visits tv, tracker_parameters tp
where tv.visit_id=tp.visit_id and tv.url like '%forums/message-view' and tp.name='message_id'
```

The first query returns the number of times where any user visit a specified package. The second query returns the number of clicks done in a thread. Generally, if the semantic is simple, the query will be simple, too.

My personal point of view is that views a good tool to retrieve information about who has visited objects in a fast and comfortable way and tracker is a good tool to store all dialog between without semantic interpretations.