OpenACS Getting Started Guide

Roberto Mello (rmello@fslc.usu.edu)

This is a guide to help you get started with OpenACS. This is not a complete guide to OpenACS and its modules... it would be too long. For detailed documentation on a module, refer to the www/doc directory in the OpenACS distribution. This guide assumes you have a working installation of OpenACS.

This is the February 2001 revision of the documentation.

Table of Contents

Next
Thinking about your web service

1. Thinking about your web service

OpenACS is a toolkit for online communities. This is a pretty powerful concept. If you think that most of the successful websites are community-oriented, you'll begin to see the power that OpenACS brings to you.

Most people get too focused on how their website is going to look, rather than thinking what it's going to provide for the users. Looks are important, but if you don't have content and services well organized, you can have as much looks as you want. Yahoo.com is pretty much all text-based, but it's fast and provides lots of content to users, or magnet content as Philip Greenspun calls.

OpenACS has over 40 modules to facilitate collaboration among users. You need to think which of these modules you can use in your web service and what's the best way to do it. A complete list of OpenACS modules is at http://www.arsdigita.com/pages/toolkit. After you have that planned and defined, you can start worrying about look and feel. You should also read the Webmasters Guide, which is at /doc/webmasters.html.

Refer to Philip and Alex's Guide to Web Publishing for more info on how to build great web services.

2. OpenACS Basics

2.1. Admin pages are your friends

For most of the OpenACS modules, you can configure them through the admin pages, accessible through */admin* from any browser. You must be registered as a site-wide administrator to access those pages (refer to the installation guide to know how to do that).

By visiting the /admin pages, you'll have most of the modules in front of you, and you'll have the opportunity to set them up. This is a great way to explore the possibilities that OpenACS opens to you.

If you see a module that you don't understand how to setup, just visit the /doc directory to learn how that module works and what it's intended to do.

For the ecommerce module, type in */admin/ecommerce* to be taken to its admin pages.

2.2. Content Sections

URL: /admin/content-sections

The content sections module allows you to add a section of the website, static or dynamic, to the users' Workspace and tell if that's publicly available or only to registered users, plus provide an introduction and help to the users.

2.3. User Groups

URL: /admin/ug

User groups are a great feature of OpenACS and gives you lots of flexibility. Unfortunately, as of this release, not all modules are user groups-ready, but that should change with OpenACS 4 (coming this summer).

There are *User Group Types* and *User Groups*. One user group type can have several user groups, which in turn can have several *subgroups*. Each user group can have modules associated with it, that members of that group will be able to use and that will be specific to that group.

For example, in the USU Free Software and GNU/Linux Club we had several projects that were being carried by club members and we wanted to provide a way for them to collaborate. Something besides mailing lists. So we created a group type called projects and several groups inside it, one for each project. Each group had an administrator assigned to it, who had control over that group an*d that could create subgroups if he/she thought necessary. OpenACS provides group pages, all ready, in* /groups.

There's much power in user groups and I highly recommed you to read the documentation about it on /doc/ug.

Prev Home Next
Thinking about your web service Look and feel

3. Look and feel

There's quite a bit of information about this on *Establishing Style and Supporting Multi-Lingualism* at /doc/style.html.

Basically you have four ways of modifying the look and feel of your OpenACS website:

1. Editing ad_header and ad_footer
2. Customized ADP tags
3. Templates
4. ArsDigita Templating System [1]

3.1. Editing ad_header and ad_footer

Almost all of the tcl pages shipped with OpenACS make calls o ad_header and ad_footer, two procedures defined in *tcl/ad-defs.tcl.preload*

ad_header returns the initial HTML tags and title to the page, and optionally returns extra stuff for the <head>. This is its usage:

```
ad_header page_title { extra_stuff_for_document_head "" }
```

ad_footer stuff an <hr> and a e-mail signature (defaults to system owner) and then it closes the body and html tags. Its usage is:

```
ad_footer { signatory "" } { suppress_curriculum_bar_p "0" }
```

For example, you'd call them from an adp page like this:

```
<%= [ad_header  My First OpenACS Page ] %>
<%= [ad_footer  me@mydomain.com ] %>
```

The <%= means evaluate this and then return as a ns_puts . The above would return a page with the title My First OpenACS page and an e-mail signature in the bottom saying me@mydomain.com .

Ths disadvantages of this approach is that it's very limited and requires restarting AOLserver for changes in the ad-defs file to take effect.

3.2. Customized ADP tags

OpenACS has some utility procedures to help you. One of them is *ad_register_styletag* (defined in tcl/ad-style.tcl).

With ad_register_styletag you can register a tag can will be available for use under ADP and TCL pages. It will also register documentation for that tag through the proc_doc OpenACS procedure.

Usually what I do is rename the file *tcl/photonet-styles.tcl* to *tcl/myservice-styles.tcl* and then edit it. In that file I define some tags like pagetop , pagebottom , pageside . Then I call these tags from ADP pages just like regular HTML tags (e.g. <pagetop></pagetop>). From .tcl pages, you'd call these tags through ad_style_pagetop (or [ad_style_pagetop] if you are calling it from inside a ns_write).

ad_register_styletag used the AOLserver API call to ns_register_adptag, which will give you more flexibility on defining your tags (e.g. you can define tags that take arguments). Read the documentation for ns_register_adptag for more info.

Although this approach is more flexible, it also requires you to restart AOLserver to make changes.

3.3. Templates

Templates are very flexible and do not require an AOLserver restart. You can find full documentation on this at /doc/style.html.

Basically, if you have a .tcl page that has the Tcl and SQL code in /test/mypage.tcl, you have this page call *ad_return_template* at its bottom. ad_return_template will search for an .adp template at */templates/test/mypage.*.adp* and return it.

You can also use *ad_return_template template_file*, where *template_file* is a .adp file in your template subdirectory that should be returned to the user.

You can have several templates with different interfaces or even languages, such as mypage.plain.adp (for users than want text-only) or mypage.fancy.adp (for users that want graphical site) or mypage.plain.pt.adp (for users that speak Portuguese). ad_return_template will server the page according to the user's preferences.

This means that programmers will edit the .tcl pages and define some variables that HTML designers will then use in the the .adp templates they will handle. If I am not mistaken, there are mods for Dreamweaver to handle adp pages.

Unfortunately, because the templating module is fairly new, only a few modules are template-enabled, ecommerce being one of them.

3.4. ArsDigita Templating System (ATS)

ArsDigita has written a very powerful publishing system that was included in ACS/Oracle 3.2.3. Older incarnations of this system could be used with OpenACS with very few modifications. Newer versions -designed for ACS 4.x- however, are not so easy. We are looking into a port of the templating system done by Vlad Seryakov, and the templating system will probably be included in the next release of OpenACS. Stay tuned!

The ATS uses XML and defines some special tags that completely separates programming from presentation. People that have been using it told me that it's really nice.

You can find more information about the ATS at http://developer.arsdigita.com/doc/acs-templating.

Notes

[1]

---

4. Backups tips from Don Baccus

4.1. The Strategy

The need for making backups should be self-explanatory. There are several strategies you can use. My own strategy for minimizing the odds that I'll lose all my data is quite simple:

- The database is stored on a mirrored (RAID 1) disk.
- The machine has battery backup.
- Backups are made nightly onto a third disk on another controller
- ftp is used to copy the resulting backup to two separate remote servers in two locations

Rather than making remote copies, you might choose to dump to tape or writeable CD media. Whatever strategy you use, it is important to routinely check dumps to make sure they can be reloaded. The strategy outlined above means that in the case of catastrophic failure, I'll lose at most one day's data.

By mirroring disks and using a battery backup, preferably one that can trigger an automatic and controlled shutdown of the system when the battery runs low, you greatly lower the odds of ever having to use your nightly backup. Despite this, it is important to take backups seriously if the data stored at your site is valuable to you or your users.

It is also important that you use the Postgres dump utility, *pg_dump*, rather than simply copy the files in the database directory unless you stop the Postmaster while making your copy. If you copy the files while the Postmaster is running and updates to the database are being made, you'll end up with inconsistent tables and a potentially unusable database. *pg_dump* makes a consistent dump even when the database is in use.

I find it convenient to use AOLserver's schedule proc routine to run a Tcl script which generates my nightly backups, rather than use cron. The major benefit is that you can very easily make a web page that calls this script, and link to it from an admin page. If there are problems making your backups (connectivity problems seem to crop up a few times a year, in my experience), you can just click on the link to force a backup with no need to remember where you placed the script, what you called it, any arguments that might be needed, etc.

4.2. The *vacuum analyze* command

Currently, Postgres doesn't automatically reclaim space in a database table when an existing row is deleted or updated.

The "vacuum" command must be run periodically to reclaim space. The "vacuum analyze" form additionally collects statistics on the disbursion of columns in the database, which the optimizer uses when it calculates just how to execute queries. The availability of this data can make a tremendous difference in the execution speed of queries. I run this command as part of my nightly backup procedure - if "vacuum" is going to screw up my database, I'd prefer it to happen immediately after (not before!) I've made a backup! The "vacuum" command is very reliable, and has never caused me a problem, but conservatism is the key to good system management.

4.3. Sample Tcl script

Here's a sample script based on the one used to back up the database backing my most important personal site, "birdnotes.net". If you're wondering why this procedure doesn't backup the scripts for the site as well, it is because they're developed on a local machine, which is backed up separately.

```
# Back up the database, scheduled to be run nightly. As written, it     # keeps a
month's worth of daily backups, cycling over the same files   # which are suffixed
with the day of the month on which the backup is   # created.      # This version:
ftp only.       # NOTE: The indenting gets screwed up during conversion to HTML
proc backup {} {
  # Set these to the appropriate values for your installation.
  set b "/usr/local/pgsql/bin"
  set bak "/home/birdnotes_backup/"
  set db [ns_db gethandle]
  set sql "select date_part('day','today'::date) as day"
  set selection [ns_db 1row $db $sql]
  set_variables_after_query
  set data "birdnotes_$day.dmp"
  ns_log Notice "Backup of [ad_system_name] starting."
  ns_log Notice "pg_dump beginning..."
  if [catch {append msg [exec "$b/pg_dump" "birdnotes" ">$bak/$data"]} errmsg] {
    ns_log Error "pg_dump failed: $errmsg"
   ns_sendmail [ad_system_owner] [ad_system_owner] "[ad_system_name] : pg_dump
failed..." "$errmsg"
     ns_db releasehandle $db
     return
}
  append msg "\n"
  ns_log Notice "gzip of data beginning..."
  if [catch {append msg [exec "gzip" "-f" "$bak/$data"]} errmsg] {
     ns_log Error "gzip of data failed: $errmsg"
  ns_sendmail [ad_system_owner] [ad_system_owner] "[ad_system_name] : gzip of data
failed..." "$errmsg"
     ns_db releasehandle $db
     return
  }
  append msg "\n"
  ns_log Notice "ftp data beginning..."
  set fd [open "$bak/ftp_data.tmp" w]
  # Replace "your_username", "your_password", and "your_remotedir" with the values
  # appropriate for the remote system on which you're keeping backup copies.
 puts $fd "user your_username your_password\nbinary\nput $bak/$data.gz
your_remotedir/$data.gz\nquit\n"
    close $fd
# "your_remoteserver" should be set to the IP of the remote system which stores your
# backups.
  if [catch {append msg [exec "ftp" "-n" "your_remoteserver" "<$bak/ftp_data.tmp"]}
errmsg] { \
     ns_log Error "ftp data failed: $errmsg"
    ns_sendmail [ad_system_owner] [ad_system_owner] "[ad_system_name] : ftp data
failed..." "$errmsg"
    ns_db releasehandle $db
    return
  }
  append msg "\n"
  # Replicate the above code to make remote copies to other systems
  ns_log Notice "vacuum beginning..."
  if [catch {append msg [exec "$b/psql" "-q" "-c" "vacuum analyze"]} errmsg] {
     ns_log Error "vacuum failed: $errmsg"
    ns_sendmail [ad_system_owner] [ad_system_owner] "[ad_system_name] : vacuum
```

```
failed..." "$errmsg"
    ns_db releasehandle $db return
  }
  ns_db releasehandle $db
  ns_log Notice "Backup succeeded."
  append msg "Backups succeeded"
  ns_sendmail [ad_system_owner] [ad_system_owner] "[ad_system_name] : backup
succeeded" "$msg
}
ns_share -init {set schedule_backup 0} schedule_backup
if {!$schedule_backup} {
  ns_schedule_daily 0 backup
  ns_log Notice "Backup has been scheduled."
}
```

5. Intranet Getting Started Guide

This section was written by Jade Rubick with slight modifications by Roberto Mello. The ACS Intranet module is being used by large corporations such as Siemens to manage their intranets.

Congratulations, you've just installed ACS. You've managed to get through the install guide (or the OpenACS install guide), you've gotten Oracle or Postgres working, you're now a pretty cool person because you got it working.

Well, if you're using the Intranet module, you're going to need some more setup, and currently, I don't know of any good documentation on the module except for what is at ArsDigita. Read that first.

The Intranet is a pretty complicated module because it relies conceptually on so many other modules. Unless you already understand the ACS, it may take a while to get a handle on it.

First of all, if you're using Postgres 7.02, I recommend applying a patch that fixes the http://www.postgresql.org/bugs/bugs.php?4~2. Preferably, you would install this before installing OpenACS, because otherwise you have to export and import your data to get this working. But the directions are pretty good, so just follow them carefully. I'm not sure if this works completely though -- I started from a fresh copy. (*NOTE: This is most likely fixed in PostgreSQL 7.0.3 and later versions.*)

Steps to getting your intranet running:

- Set up your admin account and so on like the directions say.
- Go through your */web/servicename/parameters/servicename.tcl* or .ini file and set the preferences for Intranet and New-ticket modules. Make sure you enable the Intranet module.
- In your web browser, go to */admin/users* and add in the users you want. I assume these are Employees -- they won't be able to use the Intranet unless you then go to /admin/ug and click on Intranet. Add them to both the Authorized Users and Employees category.
- While you're in the Intranet/Employees section of */admin/ug*, click on "*add module*" under "*Modules associated with groups in Employees*". Set up the news module. They will then be able to make company-wide postings for the main page, and be able to see other postings. Otherwise, they will get an error when they click on the post an item link on the main intranet page.
- Go to the /intranet page. Click on "*Offices*", and set up an office.
- Set up your partner types. This is a pain, unless I'm overlooking an easier way to do it. You have to open up a shell, start up postgres, and change the categories in the database. Then, you have to restart AOLserver, because I believe the variables are cached in memory.

```
[postgres@intranet pgsql]$ psql intranet (or your servicename)
Welcome to psql, the PostgreSQL interactive terminal.

Type: copyright for distribution terms
      h for help with SQL commands
      ? for help on internal slash commands
      g or terminate with semicolon to execute query
      q to quit

intranet=# select * from im_partner_types ;

partner_type_id |    partner_type    | display_order
----------------+--------------------+---------------
              1 | Usability          | 1
              2 | Graphics           | 2
              3 | Strategy           | 3
              4 | Supplier           | 4
              5 | Sys-admin          | 5
              6 | Hosting            | 6
              7 | Systems Integrator | 7
(7 rows)

intranet=# update im_partner_types set partner_type = 'Service Provider' intranet
-# where partner_type = 'Usability';
UPDATE 1

intranet=# select * from im_partner_types ;
partner_type_id |     partner_type       | display_order
----------------+------------------------+---------------
              2 | Graphics               | 2
              3 | Strategy               | 3
              4 | Supplier               | 4
              5 | Sys-admin              | 5
              6 | Hosting                | 6
              7 | Systems Integrator     | 7
              1 | Service Provider       | 1
(7 rows)
```

- Don't forget to restart Aolserver
- If you don't have site-wide-search, you might want to disable the site-wide-search box on the main /intranet page.
- Set up the calendar categories in /calendar/admin. Look at the documentation that won't be relevant to this at all :) Think of categories like: Social, Project Meeting, etc...
- Get pictures for everyone in your office and put them in. Or let everyone do it themselves.
- Add in any Discussion Groups you might want.
- Add in teams for the *new-ticket* module. Go to */team* to add in teams. Make sure everyone who will be using the ticket system is part of a team. If you don't do this, tickets won't work.
- Change the project types if the project types don't fit what you need. I haven't done this yet, so no help here. Sorry!

Backups tips from Don Baccus

Acknowledgements

6. Acknowledgements

Several people have contributed to this document in either content or error reporting and I would like to thank them. Please let me know if I forgot to include your name.

Contributors (in no specific order):

Don Baccus, Michael Cleverly, Gregory McMullan, Ben Adida, Jade Rubick.