

Manual for CalendarX: A Calendar for Plone CMS portals

Lupa Zurven, <http://calendarx.org>
v. 0413-01, 3rd public draft, February 12, 2005 for CalendarX-0.4.13(stable)

Table of Contents:

Part I:	CalendarX Overview	3
Part II:	A guide for Users: How to browse and add/edit events.....	8
Part III:	A guide for Administrators: How to configure the calendar	10
Part IV:	A guide for Developers: Modifying functionality in CalendarX.....	17
Part V:	Use Case: Creating a Resource Scheduler in CalendarX.....	18
Appendix A:	Installing CalendarX.....	24
Appendix B:	Guide to property sheets in CalendarX.....	28
Appendix C:	Guide to python scripts in CalendarX.....	49
Appendix D:	Guide to page template views and macros in CalendarX.....	54
Appendix E:	Guide to CSS and JavaScript code in CalendarX.....	59
Appendix F:	About Python, Zope, Plone, and CalendarX 0.5 branch	60

Note: This early draft of the *Manual for CalendarX* contains all material from the /docs folder, but is recompiled and beautified, and expanded a bit, and gradually some screenshots are appearing. But it's still a draft.

The HISTORY.txt and all other important docs will still be available in the /docs folder, just as they always have been, and they are kept updated as well.

Status of the *Manual for CalendarX*:

Part I:	(second draft) CalendarX Overview	3
Part II:	(just a note) A guide for Users	8
Part III:	(basic howto notes) A guide for Administrators:	10
Part IV:	(just a note) A guide for Developers:	17
Part V:	(first draft) Use Case: Resource Scheduler	18
Appendix A:	(full notes) Installing CalendarX.....	24
Appendix B:	(full notes) Guide to property sheets in CalendarX.....	28
Appendix C:	(first draft) Guide to python scripts in CalendarX.....	49
Appendix D:	(full draft, pics) Guide to page template views and macros in CalendarX....	54
Appendix E:	(just a note) Guide to CSS and JavaScript code in CalendarX.....	59
Appendix F:	(brief notes) About Python, Zope, Plone, and CalendarX 0.5 branch.....	60

Title: Manual for CalendarX: A Calendar for Plone CMS portals

Author: Lupa Zurven, <http://calendarx.org>

Version: v. 0413-01, 3rd public draft, February 12, 2005 for CalendarX-0.4.13(stable)

Contents copyright Lupa Zurven, 2004-2005. Some rights reserved.

This manual is made available in electronic form, and may be freely downloaded either with the CalendarX software, or it may be downloaded from the CalendarX.org website, or it may be made available at the SourceForge.net website. In a significant departure from the first public draft, this manual is also released under a much less restrictive license. For details of this license, see the CreativeCommons website, and specifically the 2.0 version of the "Attribution-NonCommercial-ShareAlike 2.0" license:

<http://creativecommons.org/licenses/by-nc-sa/2.0/>

In short, you may redistribute this work or derivative works for non-commercial purposes, as long as you give the original author credit.

In full, you are free to copy, distribute, display, and perform the work and to make derivative works under the following conditions:

- * Attribution: You must give the original author credit.
- * Noncommercial: You may not use this work for commercial purposes.
- * Share Alike: If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.
- * Reuse: For any reuse or distribution, you must make clear to others the license terms of this work.
- * Any of these conditions can be waived if you get permission from the copyright holder.
- * Your fair use and other rights are in no way affected by the above.

I make my poor living as an open source programmer, and it takes time and effort to write this manual and this software. I give my software away as an open source project, and I give this manual away to anyone who downloads CalendarX or obtains it from my websites. You can use this manual as a starting point for your own portal users, or deliver copies of it to your development team, whatever.

Just don't sell the manual, thank you. If you need to sell a manual to your clients for your version of CalendarX, please write your own, or contact me for licensing this one. I am happy to license the manual or portions of it to organizations for distribution, either on a per piece or site license basis, and I am happy to be hired to modify and customize the manual (and CalendarX itself) to make both of them perfectly suit your users, or you can do the customization of the manual yourself, but a license is absolutely required for such commercial distribution.

I hope you find this *Manual for CalendarX* (and CalendarX itself) useful.

+lupa+

Part I: CalendarX Overview

CalendarX is a web-based calendar application that is easy for users, allows a great deal of flexibility in configuration options to calendar administrators, and provides a modifiable, open source code base for developers to build on. The standard calendar includes four calendar views (month, week by days, week by hours, and daily) and provides metacalendar tools: each event has a subject, and the calendar can be filtered to show only those events belonging to one or more selected subjects. This filtering can be controlled by the calendar user, or a calendar can be configured by an administrator to show only desired subjects to all users. Users may learn more about each event on the calendar by simply rolling the mouse over the event's title (which shows a small popup box containing more information) or by clicking on the event and going directly to a page with the event's details.

CalendarX is built for the Plone content management system (CMS), which allows for a great variety of customization options for developers. In order to make basic customization easier for calendar administrators, CalendarX utilizes handy property sheets to allow access to many configuration options. These property sheets are available via the ZMI (Zope Management Interface), in the `portal_skins/CalendarX` folder.

There are many options that can be configured by the CalendarX administrator through the property sheets. These include controlling the types of events shown on the calendar, the subjects shown, the amount of information shown in the event rollover popup text, etc. There are also many controls on the CSS display of the calendar. First, the many colors of the calendar itself can be adjusted to fit the look-and-feel of the rest of the Plone site. Second, CSS styles can be applied to each event's text on the basis of what its subject is. Third, each event subject can also have a unique icon associated with it, providing rapid visual clues on a busy calendar.

Finally, for developers who wish to modify it, the source code for CalendarX is accessible to allow overrides for each instance of CalendarX in a site. It is even possible to create subcalendars, as is illustrated here in the creation of a basic Resource Scheduling calendar. Custom views can also be created for CalendarX. CalendarX is released as open source software under the General Public License (GPL).

About this *Manual for CalendarX*:

For the purposes of this manual, I'm considering the needs of three types of people:

Users: visitors and/or members of a Plone portal who may need calendar instruction for browsing and/or for entering Events for the calendar.

Administrators: someone setting up one or more CalendarX instances for a portal and configuring CalendarX to suit the needs of the portal.

Developers: programmers needing a solid calendar application for customization, or for advanced Administrators looking for additional functionality.

Part II is a **User's Guide**, a description (with pictures) on how to use CalendarX. **Part III** of the manual is an **Administrator's Guide**, offering instruction on configuring CalendarX to meet the needs of a portal's visitors. **Part IV** is a **Developer's Guide**, providing a closer look at the inner workings of CalendarX, a brief guided tour to the source code and where to look for the various functions of CalendarX. **Part V** of the manual is a case study in advanced configuration: how to use CalendarX with its subcalendar abilities to create a *Resource Scheduling* calendar suitable for an organization with equipment to loan out to its members. It would be equally suitable for a *Room Scheduling* calendar, I believe, although I personally haven't used it for such yet.

CalendarX Significant Features:

CalendarX has many features that make it a usable web-based calendar right out of the box, but its real strength is in its configurability and customizability. We will first list some of the features that make it useful with no modifications to the configuration property sheets at all.

Major Out-Of-The-Box Features:

1. Comfortable look-and-feel, like a regular calendar with month, week-by-day, week-by-hour, and day views.
2. Each event can be assigned one or more Subjects, and the calendar can be sorted by the user to show only events belonging to one or more of those Subjects.
3. Navigation is sensible; users can go to the previous or next month, week or day via a simple text link, or they can jump to a specific date in the past or future.

CalendarX has many additional features that can be configured easily by the calendar administrator without changing any of the source code. This is accomplished through the use of property sheets, which contain attributes that can be adjusted through the use of checkboxes and text boxes through the ZMI.

Major Configuration Features:

1. Colors, fonts, font sizes, and calendar cell sizing are all adjustable, as well as the type of information shown to users when they roll the mouse cursor over an event on the calendar.
2. Subjects for the events can be modified. Have as many as you like, and similar subjects may be grouped and given a group name, or nicknames can be applied to subjects.
3. Calendars can be restricted to show a limited type of event in several ways. For example, a Music calendar could show only those events that have Music as their subject, or perhaps that have Concert, Recital, Rehearsal or Jam Session as their subject.
4. Users can be shown a link ("Add New Event") that allows them to add new events to the calendar if they have the proper permission to do so (based on their

- authentication and role in the Plone CMS). You can also control the type of event and where this event will be stored through this link method.
5. You can manage the workflow of the event publishing method -- should user contributed events go straight up onto the calendar? Or should the events go through a review/publish cycle first? You decide with CalendarX and Plone.
 6. Subcalendars can be created with different properties from the master calendar. This is useful for creation of specialty calendars with many events, such as a Resource Scheduling calendar, for people to reserve rooms or equipment. Read Part V of this manual for an example of using CalendarX as a Resource Scheduling calendar.

CalendarX is further customizable by changing the source code itself. CalendarX is open source software released under the GPL, which gives you a great deal of freedom. You can change anything in the source code of CalendarX that you want, and all of the source code is available to you for this purpose. Here are a few ideas on what can be accomplished through relatively modest changes in the source code itself.

Possible new functionality available by rewriting CalendarX code:

1. Creation of new views, or changing the behavior of existing views.
2. Creation of a customized version of CalendarX to distribute.
3. Creation of custom Event types, perhaps that include space for an uploaded image, or a sound file, or a digital video clip.
4. Creating new queries to access outside databases of events.

CalendarX Licensing:

If you do create something really neat with CalendarX, we'd greatly appreciate it if you'd share your work with the CalendarX community. We have a community website at <http://CalendarX.org> where you can add Tutorials or HowTo documents, or enter bug reports or feature requests. If you create interesting modifications to the source code underlying CalendarX, we'd certainly like to hear about those too, so that we can improve CalendarX for everyone. You don't necessarily have to share your modifications, but you should be aware that under certain circumstances you may have to share your code, or else you'll not be in keeping with the license agreement that accompanies CalendarX.

CalendarX is licensed under the General Public License (GPL) to give you a great deal of freedom with the code, and to give the developer (and copyright holder) a great deal of freedom as well. The GPL allows you to customize CalendarX with NO restrictions, and to distribute your customizations with very few restrictions. In brief, this software is free and you can't license it in any way that takes away its freedom. You can give away your changes to others, even a deep rewriting of the code, but you can't do so under different licensing conditions if they are not compatible with the GPL. So for instance, you are certainly allowed to make changes that would make CalendarX more appropriate for a corporate intranet calendar and install such a program for as many corporations as you can convince to use it. You may even charge the

corporations for your services customizing, installing and configuring that program, or for training and maintenance of the program, or for any other ancillary services. But you must license it to those corporations under the GPL or a compatible license, which preserves all these same freedoms for them. You must provide the source code to these clients. They will have the right (under the GPL) to further modify and give away the changes you made for them or to their copy of the code (that's what the GPL does). They may choose not to give them away, but they must be granted the right to do so. So to put it another way, you cannot use CalendarX source code as a basis for a proprietary (non-free) computer program. Use CalendarX for making free software.

Why this license for CalendarX? The GPL is designed for software that we want to keep free for growth and exploration, free to be customized and crafted by anyone who desires to do so. In order to establish this freedom, the GPL says you can change things in the software however you want, but in order to let others use your changes, you must release your code to them with the same or freer conditions. If you just want to change the code for yourself and not release your changes, that is fine! Go ahead -- you are free to keep any changes to yourself! But in order to share your code that utilizes CalendarX code, you must ensure our CalendarX's freedom by licensing your modifications in a compatible way.

Of course, this GPL is not such an onerous license. IBM identified a tremendous opportunity in selling consulting services based on the Linux operating system; Linux is released under the GPL so IBM doesn't really sell Linux, but it does sell consulting, installation, configuration, training, and of course server hardware all designed to help organizations run computers with Linux. If IBM feels comfortable making money with GPL software, I think the rest of us can find ways to make it work too.

Therefore, you are granted the freedom to make money installing, configuring and customizing CalendarX for others, but you must release to them your code changes under comparably licensed conditions (for simplicity... just stick to the GPL). These are the same terms that apply to Plone, which is also (as of version 2.0.5) released under the GPL. These are comfortable terms for most people, and work fine for most open source consultants like me. If you aren't comfortable with those licensing terms, you should work with a different calendar application, one that allows you to change the source code and release it as your own and sell it as your own. We chose the GPL for CalendarX because we want to build a community around the calendar that feels that their contributions will be kept free for the good of all. Many of the best features of CalendarX come either from code contributions or the freely offered suggestions of our users. If you have any questions about CalendarX and its licensing, feel free to contact me and ask.

Licensing FAQs:

Q: If I want to use CalendarX as a starting point for my proprietary calendar app, will you consider licensing it to me under a non-GPL license?

A: No. We don't want to, and we really can't. CalendarX uses contributed code that has been given under the conditions that CalendarX be a GPL'd product. Also, CalendarX is fairly tightly

integrated with Plone, importing key modules and such, and since Plone is GPL'd, CalendarX is best kept a GPL product. Besides, we like it this way, not having to worry about licensing considerations and knowing it will always be around to use as long as it has use for folks.

Q: Has anyone actually asked that question about CalendarX?

A: No. We can always hope though.

```
<dtml-comment>
```

```
"We"? Why do I keep using "we" when it seems to be just "me"?  
Lupa has many friends and helpers and has been working on and off  
on CalendarX in Zope for three years before actually releasing it  
for Plone in 2004. Ongoing conversation with everyone helps  
tremendously. When it seems appropriate, as in the case of  
licensing, I'll use the royal "we" to indicate that it's more  
than just me that thinks this way. Other times, when I'm writing  
for myself, I'll use first person singular. And when I'm feeling  
testy, I might write in the third person impersonal, or even  
first person abusive. So just watch it.
```

```
</dtml-comment>
```

Part II: A guide for Users: How to browse the calendar and add/edit events

CalendarX is easy and intuitive to use. If you simply want to browse through the calendar, there are only a few things you need to know:

- How to change views (Month, Week and Day views).
- How to read details about an Event.
- How to change dates (Next period, Previous period, Jump to a date)
- How to look at just one Subject of Events at a time, or a selection of Subjects.

If your calendar has been set up to allow you to add and/or edit events for the calendar, then there are some extra steps you should learn:

- How to Add an Event (three ways).
- How to enter the Edit mode for an Event.
- What the various parts of an Event entry mean.
- Getting your event "Published".

This User guide will cover the instructions for browsing the calendar first, and then cover the details of adding and editing events. All instructions here will be for use with a standard installation of CalendarX... modifications to the calendar by your website Administrator may cause your calendar to look or act somewhat differently, but it should be similar enough that you don't have much problem following these instructions.

Welcome to CalendarX!

Step One. Let's get to know your calendar. Figure 1 is a

Top

Caption - calendar overview 01

[to be continued]

[notes from the CalendarX Help Tab]

Use the tabs across the top of the calendar screen to select different ways of viewing the organization's calendar.

Use the "previous" and "next" links to scroll through the dates to reach the date you are interested in, or alternatively you can select a date from the drop-down menus and click on "jump" to go there directly.

In any of the week/month views you can click on an individual date to view the details of events happening on that particular day.

The default view of the calendar is to show all the important dates in the organization. However, if you are only interested in (for example) dates relevant to one category of events, you can use the tick boxes and "Refresh" button near the top of the page to make the the calendar show only those dates. You can have as many selection criteria as you like, but note that you will need to make sure "view all" is NOT ticked if you want the filtering to work.

On all screens, you can hover the mouse over a particular event to find out more information about it. There may also be supplementary information for the event which you can find by clicking on it.

TO ADD AN EVENT:

Click on your *my folder* link, and select "Event" from the dropdown list. Enter information about your event, and make certain to fill out all the relevant parts like End date and Start date. Make your entries informative so that viewers of your event can find out everything they need to know.

Part III: A guide for Administrators: How to configure the calendar

How to get to the property sheets.

How to change some common properties.

How to migrate from an older version of CalendarX.

[notes from the /docs folder]

From the CUSTOM.txt file:

I. Overview

A. Go to portal_skins/CalendarX. Find one of the seven property sheets (named CX_props_yadayada). Click on one of them to view it, and then click the "Customize" button to move a copy of the property sheet into the /custom folder. Now you can modify the properties (use the Properties tab!) and start changing the calendar behavior. In brief:

CX_props_calendar: controls most basic calendar functionality, including what types of events are shown, how the Subject bar is displayed, etc.

CX_props_css: provides many opportunities for changing the colors and fonts displayed in the calendar.

CX_props_custom: does nothing. If you add new functionality to your calendar and need properties, put them in here.

CX_props_popup: checkboxes to select which text is displayed in the rollover text box associated with each event displayed in the calendar.

CX_props_addeventlink: If you wish to have an "Add New Event" link displayed in the subject bar, configure the link here.

CX_props_subcalendar: If you have subcalendars beneath your main calendar, you will need to configure them here.

CX_props_eventdisplays: allows you to use different icons and CSS classes based on the Subject of each event.

B. You can customize anything else in the /CalendarX folder. Page templates, macros, CSS or Python scripts, or Javascript... anything you find there. Usually this means clicking the "Customize" button which puts a copy of the object in the /custom folder for you, where you can change things and then refresh your calendar to see the results.

C. To create more than one instance of CalendarX is easy... just add another one from the dropdown list. However, all your calendars will use the same properties from the /portal_skins/custom or /portal_skins/CalendarX folders.

To make each calendar different:

IN ZMI: CUT all the customized objects that you need from the /portal_skins/custom folder, and PASTE them into your CalendarX folder.

Now your calendar can be customized locally and independently from any

other calendars. I like to use this approach anyway, even if I'm only using ONE CalendarX instance in my Plone site, because it cleans up the /custom folder for other uses.

D. If you want to use Subcalendars, move your customized objects out of the /portal_skins/custom folder and put them directly into your /calendar folder (using the ZMI). You will need a copy of CX_props_subcalendar in each of your MAIN and SUBcalendar folders... and you will very very likely need CX_props_calendar in each of them as well. There's a detailed section later in the manual that provides an example of using Subcalendars (Part V).

II. Tips.

1. Slots/Portlets. For most users, we recommend disabling the slots on your calendar folder, or more specifically, making the slots be empty.

1A. I WANT PORTLETS.

If you want portlets appearing in the slots, next to your calendar, use the ZMI and navigate to your CalendarX folder instance. Click on the Properties tab of your calendar folder in the ZMI. What you need here are two properties called "left_slots" and "right_slots", and they should each be of the "lines" type of properties. In these properties, type in the macro calls that will add portlets in these locations... you can see the examples of how these should look by going to the Properties tab in the root of your Plone site to see how the default ones are set up for your whole Plone portal.

1B. I DON'T WANT ANY PORTLETS TO SHOW.

If you want a full width CalendarX, then you want EMPTY left_slots and right_slots. Follow the instructions above, and make sure that your slots properties are empty. This tells Plone to collapse the left and right columns, allowing CalendarX to expand fully left and right... or nearly fully. Still seems to be a little bit of space reserved on the right side that I haven't explored how to collapse completely. But pretty good.

2. Categories for the Calendar

The calendar uses categories based on the default Event Subjects. Currently these are Appointment, Convention, Meeting, Work, Social Event. You can change these in the portal_metadata/Elements/Subject. If you do this, CalendarX needs to be told about it... this will be fixed in a future version to be automatic. Actually, it is now semi-automatic. If you want CalendarX to choose Subjects based on all the Subjects available, see the instructions in CX_props_calendar_text.txt

So, find the CX_props_calendar sheet, and list the Subjects line by line in the "listOfSubjects" attribute, just like you made them in portal_metadata. And then you're done.

Actually, there's more. You may want/need to change the CSS classes that go along with each category (subject). For this you'll need to customize the calendar.css file. See more about this in Tips #10 below.

Actually, now there's even more. If you have used very LONG subject names, like "French Connection United Kingdom", there is an option to create shorter "nicknames" for each of your subjects, and show them at the top of the calendar instead of the LONG names. For example, you could use the much shorter "FCUK" instead of the long, full name of the Subject used in the earlier example.

And now there's even more! You can make groupings of subjects, using the nicknames approach. First, in the 'listOfSubjects' attribute, list your subject groupings with commas like this:

```
Mrs Wilson 3rd Grade,Mr Smith 3rd Grade  
Mrs Farber 4th Grade,Mrs Jasper 4th Grade  
Mr Spinky 5th Grade,Mr Zurven 5th Grade  
Field Hockey,Jump Rope,Basketball,Soccer,Chess Club
```

Then in the 'listOfSubjectTitles' attribute, put the nicknames in that will be shown in the Subject Bar, like so:

```
3rd Grade  
4th Grade  
5th Grade  
Sports
```

Now when a user clicks 'Sports', events from all the sports listed in the corresponding listOfSubjects line will be returned in a calendar. This is usually much preferred to listing all 11 Subjects in the Subject Bar of CalendarX.

Also, this functionality makes subcalendar usage very powerful, as you will see if you keep reading this drivel.

3. Basic: Access to your calendar for your visitors

First, publish your /calendar folder from within Plone so that it becomes available on the portlet_navigation. Or not, if you don't want it to show up there.

Second, another nice hook is to create a portal_tab to go to your new calendar.

- A. In the ZMI, go to portal actions.
- B. Add a new action as follows:
 - Name: Calendar
 - Id: calendar
 - Action: string:\${portal_url}/calendar

Condition: (empty or whatever)
Permission: View
Category: portal_tabs
Visible? (checked)

This will display your defaultView (whichever view you have selected in the defaultView property in CX_props_calendar). Default is 'month'.

4. Multiple Calendars.

Now you can have multiple calendars on your site. Place another CalendarX instance wherever you want the calendar to appear. Chances are that if you have multiple calendars, each one will be configured differently. In that case, simply customizing a property sheet won't be enough, because portal_skins/custom can't have two different property sheets both named the same thing. SO... go into the ZMI and cut any CalendarX property sheets, scripts, icons or templates out of portal_skins/custom, and paste them directly into your CalendarX folder instance. Now these customized sheets will ONLY be found by the desired calendar instance, and you can have as many independent, unique calendars as you wish.

5. Restricting events on your calendars.

A common request is to restrict the types of events shown on a calendar. There are now several ways to do this without customizing the calendar views or macros, by using attributes in CX_props_calendar properties.

5a. Subject restrictions.

If you want to display ONLY events that contain certain subjects, you can use the "restrictToThisListOfSubjects" attribute along with the "listOfSubjects" attribute. Read the directions, and simply list the Subjects that you wish to show. For example, this might be a way to have a Music Events calendar on your site, with multiple types of Music events, as well as an Arts Events calendar, with its own set of Arts event categories.

5b. Event Type restrictions.

If you want to display ONLY events of a certain "portal_type", use the "restrictToThisListOfTypes" attribute along with the "listOfTypes" attribute. A "portal_type" is the type of Plone object that you are using for your events. For example, you can go to portal_types in the root of your Plone site, copy and paste the existing Event type, and modify it so that its type is a "Staff Event". Then you can create a calendar that will ONLY show Staff Events, just for Staff usage. A good idea is to add a getNotAddableTypes.py script in your site that will restrict usage of the "Staff Event" type to just members of your "Staff". See howtos on the Plone.org website for more instructions on how to use getNotAddableTypes.

5c. Path restrictions.

If you want to display ONLY events that are found within certain folders of your Plone site, you can use the "restrictToThisListOfPaths" attribute along with the "listOfPaths" attribute. Read the directions, and simply list the full paths (as found in the Path index in the portal_catalog) for the folder objects you wish to use. Then ONLY those events found within those folders will display on your calendar.

6. Look at all the calendar properties, and read about them in the /docs folder. There are lots of things you can adjust, and they're all documented. Try them and see what you like.
7. There's an attribute in calendar properties that lets 'visible' as well as 'published' events show up on the calendar.
8. Nearly all the most important CSS attributes are now adjustable from the CSS properties sheet. Font sizes are all expressed in percentages so that they can vary the way the rest of Plone can, with simple stylesheet changes.
9. "Add New Event" link: A new addition to 0.4 branch is the ability to put a link in the Subject Bar that allows users to click on it and go to an appropriate folder where they can add a new Event. There is a property sheet that controls this behavior (i.e., where the link goes and who gets to see the link at all). In brief, the macro controlling this link first checks to see that the user is Authenticated on the site (few sites allow un-Authenticated users to add Events, and those who do can easily disable this in the macro code). The link then can be set to the following possibilities:
 - A. Link to the Member's personal folder.
 - B. Link to a specified folder.
 - C. Link for certain users to go to different folders (one specified folder per specified user).
 - D. Link for users with specific Roles to go to different folders (one specified folder per specified Role).
 - E. Link to a specified subfolder within the Member's folder.If choice C or D is selected, and the current users is not listed among the possible choices (for listed users or Roles), then the link will roll back to choice B or A, if either of these has been checked. If the user is not found in C or D and neither B nor A has been checked, then the "Add New Event" link will NOT be displayed for this user.
10. New Icons and CSS classes, selected by Event Subject. Now your calendar can really show its colors by letting you highlight each event on your calendar in color and with icons depending on the Subject. Simply

check the appropriate box and follow the examples, listing a single line for each of your Subjects that tells it what icon to use. Add those icons to your /custom folder, or /calendar instance folder, and your calendar will take on a whole new look. Read the details in the CX_props_eventdisplays.txt file in the /docs folder. Or just check the box and try it with the icons included in the default settings.

You'll also need to add special CSS classes for your subjects. The default ones are in calendar.css... look at them to see how to do it for yourself. There are currently two classes for each subject: one class for the link shown within the calendar proper, and one class for the Subject bar up at the top of the calendar page.

Also: Icons and CSS by Event Type. Same as above, but works for the specified Event types listed. Should come in handy for tricky subcalendars.

11. **SPEED TIPS.** CalendarX is a busy product, doing lots of queries and lookups in order to display a full-featured calendar. Here are two speed tips to make sure that CalendarX is running as fast as reasonable. The first one is imperative, the second is definitely optional. Neither have been rigorously benchmarked, but informal benchmarks give the first one a potential speed up of as much as five-fold. **SO DO IT**, if it's not already done for you. It's painless.

Speed Tip #1: The skins of Plone create an unbearable performance hit. This tip speeds up the calendar by putting it near the top of the skins list of layers. Moving it up from the bottom of the skin layers list can increase speed severalfold. Go to portal_skins in your Plone site, and click on the Properties tab. Then find CalendarX in the listing for your plone skin (usually Plone Default or Plone Tableless) and move CalendarX up to the second spot, right underneath "custom". That's it. Now your calendar skin will be found very quickly.

Note: CalendarX tries to install up at the top below "custom" at installation. So this is probably already done for you. But if you install other products **AFTER** CalendarX, they may install ahead of CalendarX in your skins. So you may need to go back and move CalendarX up in the layers at some future time.

Speed Tip #2: Customize the query methods from /portal_skins/CalendarX folder and move them locally to the location of the calendar instance itself... no more skin-inefficiency, because they are found locally. This should help, but I have no real benchmarks to prove it. In order to really work, you should move virtually all the scripts and property sheets to your CalendarX instance. This should definitely be faster than having your scripts all in the skin layers hierarchy, but probably not much of a

boost over Speed Tip #1 and it is more work, so don't bother.

I'll keep looking for other tips and speed ups. I made some code changes in 0.4.3 to speed up the month query, and need to consider some similar changes in the other templates soon. There are other speed optimizations in the new 0.5 branch, but it will not likely be faster than the 0.4 branch because it also has more complex queries.

Part IV: A guide for Developers: Modifying functionality in CalendarX

How CalendarX works -- a brief guided tour.

Contents: (future)

The main folder, and what it does.

The python scripts.

The page templates.

The property sheets.

The macros.

The CSS and Javascript files.

A. The main folder, and what it does. -- For now, read Part III for Administrators over again.

B. The python scripts. -- For now, read Appendix C.

C. The page templates. -- For now, read Appendix D.

D. The property sheets. -- For now, read Appendix B.

E. The macros. -- also covered (briefly) in Appendix B.

F. The CSS and Javascript files. -- For now, the miniscule Appendix E must suffice.

Part V: Use Case: Creating a Resource Scheduler with CalendarX

A common application for calendar applications is the scheduling and reservations of available equipment, rooms, and facilities for an organization. CalendarX can handle the basic functionality of a Resource Scheduling application through the use of subcalendars. Here we will go through the steps you need to create a Resource Scheduling calendar application for an environmental group that has a number of Beamers (LCD projectors), Bicycles, Cameras, Laptops, and Rooms at their headquarters that are made available to their members on a first-come-first-served basis.

The calendar will be organized into one Main calendar with several nested Subcalendars. We'll name the Main calendar "Resources", and we'll name each of the Subcalendars for its primary resource: Beamers, Bicycles, Cameras, Laptops, and Rooms.

Imagine that you are in this organization and you have an important meeting to attend in Antwerp next week. It's a last minute invitation to speak to the executive board of another environmental group, and you'll need to take along with you both a beamer and a laptop. You should be able to quickly examine the calendar to see only the laptop and beamer schedules, so that you know whether you can borrow ones from the group, or if you might have to make other arrangements.

That's an easy one. The Resources calendar will display ALL the resources that are being used on each day, but it will have the ability to allow users to filter the Subjects so that only select resource categories are visible. This is just like the regular Subject filtering done by CalendarX elsewhere, except here our Subjects of interest will be Laptops and Beamers. So you would just check the boxes by Beamers and Laptops (and uncheck the one that says View All), and hit the Refresh button. Now you can see which beamers and laptops are checked out on the days you need them for Antwerp.

Overall, this one view should work fine for most uses. Use the Month view to start, then zoom in to the week or day views. Even with just one Subject category chosen (e.g., Laptops), the calendar would show all the scheduled Laptop reservations, and a quick rollover with the mouse could display which laptops are reserved, etc. But sometimes a more detailed view is desired, or perhaps you ONLY want to see the reserved times for the really nice laptops, the ones with 1 Gb of RAM and the 17" wideview monitors. To accomplish this, you would click on the Subject link for Laptops and you can enter the Laptops subcalendar. Here only laptop reservations are shown, and the Subject bar now lists all the available laptops. To reduce the clutter on the screen, select just the few laptops that you are interested in, and hit the Refresh button to show their availability.

So let's build this resource scheduling calendar. Right now. It should take about 30-40 minutes or so if you have a clean Plone portal to work on. The first step will be to create some custom Event types for the various resources we'll be scheduling. The second step will be to build the Main calendar and configure it, and finally we'll add the subcalendars and configure them. We use custom Event types because it makes it much easier to restrict our Resources

calendar to only see the events that we're interested in, and they won't interfere with other calendars in this Plone portal.

Step 1. Create New Event Types.

First you need to create content types for each resource event. We do this because restricting CalendarX by event type is a handy way to create a nicely organized resource scheduling calendar. An easy way to get new content types is to "repurpose" the existing Plone Event type:

- A. In the ZMI for your portal, click on *portal_types*.
- B. Select the checkbox beside the *Event* type. Then hit the COPY button at the bottom of the page, and then hit the PASTE button. This will create a duplicate of the *Event* type called *copy_of_Event*, and we can rename this for our first Resource type.
- C. Select the checkbox next to *copy_of_Event* and then click the RENAME button at the bottom of the page. In the form, give it the new name *Reserve a Beamer*.
- D. Click on the new *Reserve a Beamer* type, which will open up the form for editing it. Now type "Reserve a Bicycle" in the form for the *Portal Meta Type* property, which will nicely identify this special Event type in the *portal_catalog* (this will show up in the catalog metadata as both "portal_type" and as "Type"). Optionally, you can type in a new description here too ("Use this Event type to Reserve a Beamer on our Resource Scheduling Calendar"). Hit SAVE at the bottom of the form.
- E. Repeat this process: start with the *Reserve a Beamer* type, COPY, PASTE, RENAME several times to create event types for Bicycles, Cameras, Laptops and Rooms. Be sure and edit each one to change its *Portal Meta Type* property. Now you have all the Event types you need.

Before we go on, we should also set up the proper subject categories for these new event types. For resource scheduling, the Subject will be the specific resources that will be available to schedule. In this case, we'll add **Beamer1**, **Beamer2**, **Beamer3** as our three possible Subjects for the *Reserve a Beamer* event type (we only have three beamers available):

- F. In the ZMI for your portal, click on *portal_metadata*.
- G. Click on the *Elements* tab at the top, to take you to the Update Element Metadata Policies tab. Then click on the *Subjects* element type in the menu list of metadata elements available.
- H. Now you can see a list of Content Types (such as *Event*) and a number of options associated with their Subject metadata. Subjects are created specifically for each content type, using this form. Since your content types are new, they aren't yet listed here. Drop down toward the bottom of this form to the section for adding a new type to the roster... it's an empty form for a <new type>. Use the dropdown list for *Content type* to select *Reserve a Beamer*, and in the Vocabulary textbox, list your three Subjects, one per line:
Beamer1
Beamer2
Beamer3

Also check the box that says *Enforce vocabulary* (yes). This means that users must choose one of the Beamers when they fill out this *Reserve a Beamer* form. Then click ADD to add this to the metadata.

- I. Repeat step H for each of the other *Reserve a Resource* types you've created. Give them each a vocabulary of resources available for checkout. For the *Rooms*, list the room numbers or room and building numbers, whatever is appropriate for your users to readily identify each resource. If Beamer intensities or other properties are important, name your resources accordingly (*Zoom 2400L*, *NoZoom 1500L*, etc.)

Now your new Resource content types are ready. Let's make the calendar.

Step 2. Create the Main Resources calendar.

- A. In the ZMI, add a CalendarX Content, and select CalendarX (the only option) and name it what you want (the id of the calendar). This will be your MAIN calendar. We'll create your Subcalendars in a bit.
- B. Go to /portal_skins/CalendarX and hit customize on each of these three property sheets: CX_props_calendar, CX_props_popups, and CX_props_subcalendar. Then in the /portal_skins/custom folder, select all three of them and click the CUT button. Navigate to your new MAIN calendar instance and then click the PASTE button, so that all three of these fresh property sheets are now inside your MAIN calendar folder.
- C. Set the following configuration settings in this MAIN calendar (we'll set up the subcalendars afterward)...

MAIN calendar:

config: CX_props_calendar:

1. useMultiSubjects: checked. The cool subcalendar menuing options ONLY will work with this option selected.
2. listOfSubjects: names of the subcalendars I'm using: Beamers, Bicycles, Cameras, Laptops, Rooms
3. eventTypes: I'm using five new Event types created by simply repurposing the CMF Event type, with new names and subjects.
4. restrictToThisListOfTypes: checked, because I want this calendar to ONLY show these special reservation events.
5. useCalendarHelp: checked. I think calendar help will be needed because subcalendars may present conceptual problems for some users.

MAIN calendar:

config: CX_props_subcalendar:

1. useSubCalendarSubjectMenu: checked, because you need that to properly navigate to your subcalendars.
2. listOfSubCalendars: fill this with the list of IDs for the subcalendars. Here this is: beamers, bicycles, cameras, laptops, and rooms.

3. isSubCalendar - nope, not one.
4. nameOfSubCalendar - nope.

MAIN calendar:

config: CX_props_popup:

1. expanded popup coverage by checking several options, including: showPOPTitle, Type, Subject, Start, End, Creator, Description.

Step 4. In the ZMI, inside the MAIN calendar, add a new CalendarX Content, and select CalendarX (the only option) and name this one what you want for the id of one of your Subcalendars.

Step 5. Go to /portal_skins/CalendarX and hit customize on each of these two property sheets: CX_props_calendar and CX_props_subcalendar. Then in the /portal_skins/custom folder, select both of them and click the CUT button. Navigate to your new Subcalendar instance and then click the PASTE button, so that these two fresh property sheets are now inside your Subcalendar folder.

Step 6. Set the following configuration settings in this Subcalendar (we'll set up the other subcalendars you want afterward)...

SUB calendar:

config: CX_props_calendar:

1. useMultiSubjects: checked. The cool subcalendar menuing options ONLY will work with this option selected.
2. listOfSubjects: names of the subjects desired for this subcalendar. For example, in the Bicycles subcalendar I'm using: Bicycle1, Bicycle2, etc. If this was a school calendar, these might be names of teachers, or sports, or... whatever you want.
3. eventTypes: I list the one type of Event that I want to show in this subcalendar because I'm restricting my events this way: for my Bicycle subcalendar, this uses: Reserve a Bicycle.
4. restrictToThisListOfTypes: checked, because I want this calendar to ONLY show these special reservations for bicycle events.
5. useCalendarHelp: checked. I think calendar help will be needed because subcalendars may present conceptual problems for some users.

SUB calendar:

config: CX_props_subcalendar:

1. useSubCalendarSubjectMenu: UNchecked. ONLY for main calendars.
2. listOfSubCalendars: Unused. ONLY for main calendars.
3. isSubCalendar - Yes, because this is a subcalendar.
4. nameOfSubCalendar - A name (Bicycle).

Step 6b. Note that we did NOT include a `CX_props_popup` in the subcalendar.

That's because we want the Subcalendar behavior to inherit the values of the MAIN calendar for these popups. ONLY add property sheets into a subcalendar if you don't want to inherit the values of the properties of the MAIN calendar.

Step 7. In the ZMI, in the MAIN calendar, select the checkbox for your Subcalendar, and click the COPY button. Then hit the PASTE button four times, to create four subcalendars. Each one will already have the property sheets inside them, although they each need some configuration, as follows (leave other choices as they were set for the first subcalendar):

SUB calendar:

config: `CX_props_calendar`:

2. `listOfSubjects`: names of the subjects desired for each of these additional subcalendars.

For example, in the Cameras subcalendar I'm using: `Camera1`, `Camera2`, etc. If this was a school calendar, these might be names of teachers, or sports, or... whatever you want.

3. `eventTypes`: I list the one type of Event that I want to show in this subcalendar because I'm restricting my events this way: for my Camera subcalendar, this uses: `Reserve a Camera`.

SUB calendar:

config: `CX_props_subcalendar`:

4. `nameOfSubCalendar` - A name (ex. `Camera`).

Step 7b. You make those changes for EACH of the subcalendars.

Step 8. Go add some of your new resource scheduling events. See them on your calendar. You'll need to publish them, of course. That's fairly comparable in workflow terms to asking permission to schedule a given resource, so I think that's a reasonable default setup. Alternatively, you can check the `includeReviewStateVisible` property in `CX_props_calendar`, and then all entries will be shown as soon as they are entered in the Visible state.

Optionally, you can use some of the other property sheets to tweak your calendar further. For example, you probably will want to skin the calendar's CSS properties so that the colors and fonts match the rest of your website. So when you do Step 2B above (or later, after you've thought about it), repeat Step 2B for the `CX_props_css` and `CX_props_addeventlink` property sheets into the Main calendar folder, and configure them appropriately. You won't need to copy

them into the subcalendar folders unless you want different CSS options for the subcalendars... I can see when this might provide a nice touch, but not on most calendars.

A good reason to use the *CX_props_addeventlink* property sheet here is if you want your Users to put all their *Reserve a Resource* events into one folder. That might make more sense for some organizations, and will certainly make cleanup of old Resource events easier for a calendar Administrator. In that case, configure the *Add New Event* link to direct all users into a single Plone folder where events will be stored. Make sure that the folder has permissions set properly so that all Users can add events there.

That's all. Now go and add lots of events and see how it all works. See how the navigation works, and allows you to filter according to the types of resources you may want, or lets you drill down and only look at one resource at a time in the subcalendars.

Go crazy and dream up new uses for subcalendars. I think it should work great for school calendars, and I'll probably be setting one up that way soon. Let me know what great calendars with subcalendars you create. Go to CalendarX.org and add a *CalendarX Sighting* document with a link to your CalendarX calendar so the community can see what you've done.

Appendix A: Installing CalendarX for an existing Plone portal

I am not going to tell you how to install Plone (or Python or Zope) here. I have opinions on that but I'm not going to tell you them here. So I am assuming that you have a reasonably current (version 2.0.x) Plone installation, and that's why you want CalendarX. If you have not used Plone before and you just want to try CalendarX out, then I'd recommend getting one of the easy Plone installers (the binary Plone installer for Windows machines works alright for me) and follow the instructions first, and then come back here.

CalendarX 0.4 branch is a Plone Product. It installs as a Product in your Products folder. The instructions below explain this.

I. Software requirements

II. Instructions for Installation for the first time.

III. Re-installing, or upgrading CalendarX

I. Software requirements

Software Requirements for CalendarX (0.4 branch):

1. Plone 1.0+ along with Python/Zope. Tested on Plone 2.0.5, Zope 2.7.3.
Not tested yet on Plone 1.0+ but I'm pretty sure it will still work.

Software Options, Suggestions (NOT REQUIRED):

1. Python 2.2+ (required if AdvancedQuery is used.)
2. Zope 2.6.2+ (may work on earlier Zopes, but untested. Python 2.2+ is not standard until Zope 2.7+)
3. Plone 2.0+
4. AdvancedQuery (tested on 0.3+ at this time)

this is available at Dieter's site:

<http://www.dieter.handshake.de/pyprojects/zope/>

AdvancedQuery is one of two query techniques used. By default, the traditional ZCatalog query method is used, but using AQ is simply a switch option in the skins property sheet.

II. Instructions for Installation for the first time.

1. Make sure you have the required software and additional Zope Products in place.
2. Acquire the CalendarX-0.4.x.tar.gz file from sourceforge. This unzips as a tar file, which untars as a single folder, called CalendarX-0.4.x.
3. In your Zope INSTANCE, place this folder in your INSTANCE_HOME/Products/ folder and rename the folder to /Products/CalendarX instead of /Products/CalendarX-0.4.x.

4. Restart Zope.
5. Go to your Plone portal, and log in as a manager. Go to plone_setup and Add New Products. Find CalendarX in the list and check it, and click the Install button. This is equivalent to navigating in the ZMI to your Plone instance, and using the portal_quickinstaller to install CalendarX.
6. IN PLONE: Navigate to where you want a CalendarX instance, and add a CalendarX in the normal fashion. Name it "calendar" or whatever you want. It creates an "empty" folder, but when you now navigate to it in Plone, it will display the Calendar, set to the default view (Month view).
NOTE: CalendarX is **not** an ordinary Plone folder. The folder_listing and folder_contents have been disabled. Do attempt to store Plone content inside your CalendarX instance. Certain advanced features of CalendarX require you to use this folderish behavior to store CalendarX scripts and other files within it (such as having multiple calendars on one site, and the use of subcalendars). See step #9 below for more information.
7. That's it. Navigate in Plone to your calendar folder and the default (month) view should appear. You may then publish (or not) your calendar as you wish.

III. Re-installing or upgrading CalendarX.

- A. Upgrading from 0.4 branch versions of CalendarX.
 - B. Upgrading from 0.2 branch versions of CalendarX.
 - C. Upgrading from 0.1 branch versions of CalendarX.
-
- A. Upgrading to from 0.4 branch versions of CalendarX.
 1. Replace the old version of CalendarX in your INSTANCE_HOME/Products folder with the new version of CalendarX (0.4 or higher).
 2. Restart Zope, or in the ZMI you can go to Control Panel, Product Management, CalendarX, and in the Refresh tab, click Refresh. This step is necessary to get some parts of CalendarX recompiled.
 3. In ZMI, navigate to your Plone root, and go to portal_quickinstaller, and reinstall CalendarX. This step is needed to refresh the skins so they know what's in the new CalendarX skins folder. Alternatively, in Plone (2.0+), you can go to Plone Setup, Add/Remove Products, and under "Installed

Products", look for CalendarX and click the link that says "This product has been upgraded, click here to reinstall." That's just another way (through Plone) of accessing the portal_quickinstaller.

4. Any parts of CalendarX that you have customized should be compared with fresh versions (and the HISTORY.txt file) to see if there are changes that you should be aware of in your customizations. If all you have changed are the CX_props_XXX property sheets, then there is little chance of any problems.
5. There is not currently, nor immediate plans to create, an automated upgrade feature that will migrate your configurations to the new versions of the property sheets, etc. This is because CalendarX is designed to allow easy TTW access to all the scripts and templates for customization, and any one CalendarX installation will be expected to have many customization choices that an automated upgrade mechanism will not properly handle. Document the changes you make so that you can more easily upgrade to newer versions of CalendarX.
6. As you make your own customizations to scripts, page templates or macros in CalendarX, you'll see my examples of how you can make comments for any changes you make to the code. If you make changes, BE SURE and add comments so that you remember in the future what you have done, and you can easily find your changes when upgrading time comes. And if you make really good changes, TELL US ABOUT THEM!!

B. Upgrading from 0.2 branch versions of CalendarX.

User Question: I looked through all the docs and I don't see any instructions on how to migrate from version 0.2.13 to 0.4.3. Since you can't use the quickinstaller to remove 0.2.13 from Plone, will 0.4.3 simply overwrite 0.2.13 or do you have to manually remove files from the file system?

Answer: Ah, migration.

It really isn't much problem, although it is not perfectly transparent.

1. Leave your current (old) calendar in place for now. The two can run concurrently if you wish, until the new one is all set up. Or forever (I have both running as demos at calendarx.org).
2. All the files for the 0.2.13 branch are stored in the instance folder, so they will not interfere with the new one at all. When you're done with 0.2.13, just delete that folder.
3. Install 0.4.x as per instructions (untar, unzip in /Products, restart Zope, use QuickInstaller to add CalendarX to your Plone site. Then (in

Plone), use the dropdown to add a CalendarX instance to your Plone site. Name it whatever you want. You can change the name later in the usual Plone way.

4. Customize your CalendarX in approximately the same way that you did before (with 0.2.13) except that now it's more Plone-ish... you go to `portal_skins/CalendarX` and find the property sheet that you want to alter, hit Customize and then change the Properties in the Properties tab. You can likewise customize any of the property sheets, the macros, the templates, the scripts. **THERE IS NO MECHANISM FOR DIRECT MIGRATION** from the old 0.2.13 property sheets to the new ones. Most of the properties have the same names though, so it should be fairly easy. 10 minutes tops to migrate if all your customizing has been in the property sheets.
5. You can leave these property sheets in your `/custom` folder, or you can use the ZMI and cut/paste them into your CalendarX instance folder so that they are out of the way.
6. If you want more than one CalendarX instance, you will definitely want to use the ZMI and cut/paste your property sheets into the local instance folders... that way each calendar has its own look, feel, and behavior (what events it locates, etc.).

Hope that helps.

C. Upgrading from 0.1 branch versions of CalendarX.

Ugh. No idea at this moment. Probably a bit like the ones above. Sorry, but write me if you are having trouble. Don't upgrade, just replace the olde one and start fresh.

Appendix B: Guide to property sheets in CalendarX

Property sheets are the tools that allow calendar administrators to configure CalendarX easily, without touching the source code. Easy access to these properties allows a great deal of flexibility in how CalendarX looks and what events it displays. Here is a guide to each of the properties associated with the property sheets, what you can accomplish with it and how to set it properly.

Administrators: Changing the properties associated with each property sheet is best performed in the usual Zope/Plone way... navigate in the ZMI to your /portal_skins/CalendarX folder and click on the name of the property sheet of interest to take you to a view of the properties on the sheet. Then click the button to "customize" that property sheet, which will place a copy of the property sheet in the /custom folder. To change property values, click on the **Properties** tab of the property sheet, and adjust values as desired, then click the Save Changes button at the bottom of the page.

If there are multiple CalendarX instances in a portal, then a property sheet in the /custom folder will override default values established in the filesystem. If different settings are required for just one CalendarX instance, then a copy of the appropriate property sheet can be placed inside the desired CalendarX folder (using the ZMI top copy/paste the property sheet there).

Developers: Property sheets are found in the /Products/CalendarX/skins/CalendarX folder of your distribution.

Officially, these property sheets are objects of a CMF class called FSPropertySheets (File System Property Sheets). The FSPropertySheet class currently (as of Zope 2.7.3) has deficiencies in its ability to handle lines and text properties, so CalendarX patches the methods used to handle these using the so-called "monkey patch" method so that FSPropertySheets can use lines and text properties. The monkey patch can be examined in the `__init__.py` file (the part that says "fixing the 'lines' property for CMF FSPropertiesObject"); this monkey patch provides the bulk of the `__init__.py` code.

If you want to change the default settings for CalendarX for your entire portal, or for all your portals (if running more than one Plone on your server), you can change the settings in the FSPropertySheets in the CalendarX folder in your INSTANCE_HOME/Products folder. Then all instances of CalendarX reading from that /Products folder will receive those defaults, unless overridden locally as described above in the Administrator's section.

Overview: Here are the names and basic function of the property sheets. The remainder of this appendix will cover details of the properties of each sheet in this order. I've described these in alphabetical order, but #2 (CX_props_calendar) is the most commonly used one.

1. **CX_props_addeventlink:** If you wish to have an "Add New Event" link displayed in the subject bar, configure the link here.
2. **CX_props_calendar:** controls most basic calendar functionality, including

- what types of events are shown, how the Subject bar is displayed, etc.
3. **CX_props_css**: provides many opportunities for changing the colors and fonts displayed in the calendar.
 4. **CX_props_custom**: does nothing. If you add new functionality to your calendar and need properties, put them in here.
 5. **CX_props_eventdisplays**: allows you to use different icons and CSS classes based on the Subject of each event.
 6. **CX_props_popup**: checkboxes to select which text is displayed in the rollover text box associated with each event displayed in the calendar.
 7. **CX_props_subcalendar**: If you have subcalendars beneath your main calendar, you will need to configure them here.

1. CX_props_addeventlink

The "Add New Event" link is a means of making it easier for calendar users to add new events to your calendar. It places a link in the SubjectLinks bar that takes users to an appropriate place to add events. Because there are many places/ways that Plone allows you to add events, we've provided several options for how to control who gets what link. Your suggestions (and code) for more options are gratefully accepted.

=== List of Attributes ===

title *string*

Leave this title attribute alone.

showAddEventLink *boolean*

Check this to include an "Add New Event" link in the SubjectLinks bar.

Controls for this link are below. If more than one of the boolean controls below are checked, the ones below will take priority over the ones above. For example, if both useANEFolder and useRolesAndFolders are checked, but the current user does not have one of the specified Roles, then the link target will fall back to the specified ANEFolderPath. The order of these priorities is determined by the code in the Python script "getAddNewEventURL". If no match is found, then a blank string will be returned to the macro, and a condition there will cause NO "Add New Event" link to be shown. This way you can restrict display of this link to only certain users or to users with certain roles. The first two choices (useMemberFolder and useANEFolder) are shown to all Authenticated users, if selected.

useCreateObjectOnClick *boolean*

createObjectOnClickCommand *string*

Together, these two properties tell the link to instantiate a new Event

object in the target folder. Check this if you want to have the link automatically initiate editing of a new event for the user. Uncheck this property if you want the Add New Event link to simply take the user to a target folder without starting a new Event object automatically.

The `createObjectOnClickCommand` string is the command that is carried in the query string of the link's URL target if you are using the `useCreateObjectOnClick` property. The default string is:

```
createObject?type_name=Event
```

which will create a new CMF Event object in the target folder of the link. If you have a different event type that you would like to create, replace the meta_name "Event" with the appropriate meta_name of your desired event type.

If you use this feature, it is advisable to also set your portal to use the `portal_factory` for initiating Events, so that if a user clicks on the link to start a new event but then decides not to finish it, the event will not be abandoned half-finished. `Portal_factory` will simply create a temporary version and then delete it if left unfinished by the user.

useMemberFolder *boolean*

Check this so that the link will take users to their default Member folder where they can add Events.

useMemberSubfolder *boolean*

memberSubfolderPath *string*

Check this property if you want events to be instantiated in a subfolder of a user's Member folder. For example, if all your users are musicians in bands (or groupies perhaps) and they post their band gigs on the calendar, then you might want all the events to be saved in a specific subfolder, such as `"/Members/username/gigs"`. The proper format for the target folder is relative to the `/Members/username` folder and should start with a slash, e.g.: `"/gigs"`

NOTE: ONLY use this if you are certain that users WILL have the named subfolder in their user folder. Otherwise it will return 404, page not found error, or something closely related.

useANFolder *boolean*

ANFolderPath *string*

Together, these allow you to specify a single folder that will be the target of the link. The proper format for the target folder is relative to the `portal_root` and should start with a slash, e.g.: `"/somefolderintheroot/thefolderforevents"`

useUsersAndFolders *boolean*

listOfUsersAndFolders *lines*

Together, these allow you to specify a combination of a username and a corresponding single folder that will be the target of the link for that specific user. The proper format for each line is as follows:

```
"username|folderpath"
```

where the "pipe" character (a vertical slash) is used as a separator between the username and folder path. The proper format for the target folder is relative to the portal_root and should start with a slash, e.g.:

```
"/somefolderintheroot/thefolderforevents"
```

An example with two possible role|folder lines:

```
lupa|calendar/specialevents
```

```
davos|calendar/drearyevents
```

If no matching username is found, the priority rules described above will take over to find a suitable target for the user.

useRolesAndFolders *boolean*

listOfRolesAndFolders *lines*

Together, these allow you to specify a combination of a Role and a corresponding single folder that will be the target of the link for all users with that Role. The proper format for each line is as follows:

```
"rolename|folderpath"
```

where the "pipe" character (a vertical slash) is used as a separator between the role name and folder path. The proper format for the target folder is relative to the portal_root and should start with a slash, e.g.:

```
"/somefolderintheroot/thefolderforevents"
```

An example with two possible role|folder lines:

```
Manager|calendar/specialevents
```

```
Member|calendar/ordinaryevents
```

The lookup stops when a matching role is found for the user. For example, if the Manager logs in, the link for the Manager will target the folder called "specialevents", even though the Manager is also (likely) a Member of the site. If no matching role is found, the priority rules described above will take over to find a suitable target for the user.

2. CX_props_calendar

CX_props_calendar controls most basic calendar functionality, including what types of events are shown, how the Subject bar is displayed, and

many other attributes of the calendar's look and feel.

=== List of Attributes ===

title string

Leave this title attribute alone.

dayOfWeekToStart int

Value indicates what day of the week the Month and Week views begin on. Sunday = 0, Monday = 1, etc.

defaultView string

Name of the default view to be displayed: month, weekbyday, weekbyhour, or day.

useAdvancedQuery boolean

If checked, CalendarX will use the AdvancedQuery product for making queries to the catalog to find events. Use this if you want to override those query methods in your skin folder. I find AdvancedQuery significantly easier to use in building complex queries, but it offers no other advantages for general use in CalendarX.

dayViewStartHour string

Hour of day for CalendarX to BEGIN display for day view and for weekbyhour view. 8 = 8am = 08:00, 20 = 6pm = 20:00. For a 24 hour calendar, set this to 0 (zero).

dayViewEndHour string

Hour of day for CalendarX to END display for day view and for weekbyhour view. 8 = 8am = 08:00, 20 = 6pm = 20:00. Must be later than dayViewStartHour. For a 24 hour calendar, set this to 24.

hoursDisplay string

Code to tell the "hoursdisplay" macro how to display the hours in your calendar views. Currently two possibilities and they only affect the left column display of hours:
"12ampm" = 12 hour display, with am or pm. Ex. 6 pm
"24.00" = 24 hour display, with period. Ex. 20.00

showHighlightFullEvent boolean

If checked, CalendarX will show the full extent of Events on the calendar, even without rolling over with the mouse. Default is a blue color, can be changed in CSS property sheet.

NOTE! If you use this, you might also want to disable the

labelEventsOnlyAtStart property. Disabling labelEventsOnlyAtStart means that the events in the Month view that span several days will show labels for each of those days, instead of only on the first day of the event.

showJumpToDateWidget boolean

If checked, a date-picking widget will show up at the top and bottom of the calendar near the Next/Previous links. This widget lets users pick a date and jump to it, instead of using multiple Next, Next, Next click, or manually typing the date into the URL querystring.

useNumericMonthInJumpToDateWidget boolean

If checked, the Jump-To-Date widget will show a numeric month value (ex. "2") instead of an abbreviation of the month (ex. "Feb"). These abbreviations are pulled from the python DateTime module, not coded into CalendarX code, in the getMonthName.py script.

showPublicPrivateLink boolean

If checked, the *Public* vs *My Events* link will be shown in the Subject Bar. This link allows users to switch between viewing all the published events, or ONLY their own private events. If your calendar is mainly for viewing by anonymous users, you probably don't need this. Default is OFF because this is a nice feature, but not a commonly chosen one.

useMultiSubjects boolean

If checked, the Subject category picker is a checkbox-style form, allowing users to select multiple subjects for viewing. If unchecked, it switches to (an older) single subject chooser that only allows one Subject category at a time. Default is ON for this new-style Multi-Subject chooser, because it is ever so much nicer.

showSubjectBar boolean

If checked, the Subject bar is shown, and if unchecked, it will disappear from view. Default is ON. Decomplicates the calendar if you don't want to use Public/Private or Subject categories.

useCalendarHelp boolean

Check this attribute to show a View tab for "Calendar Help". This brings up a new view page that is intended for you to use for help in case you have neophyte users who could use some calendar help. I've added some code that brings up one page of help for "Members" and a different page of help for "Anonymous" users. This could easily be extended to show

different help screens for other Roles. See the "help" view page template for more information. The help text is quite minimal, so feel free to expand it for your users. Feel free to send me a copy of your nice help files, too!

includeReviewStateVisible boolean

Check this attribute to include events where the review state is 'visible' as well as 'published'. This is useful for calendars where the only users are trusted users and going through the publishing workflow only adds unnecessary complication.

In particular, this could work well even on a site with many untrusted users. In that case, create a calendar for the trusted users that uses a repurposed Event with a new portal_type name. Use the 'restrictToThisListOfTypes' attribute to make this new calendar ONLY read this one type of Event. Then use a getNotAddableTypes.py script to restrict the use of this type of Event to your trusted users (as a role, or a group, or whatever). See the HowTo on plone.org for use of getNotAddableTypes.

showPendingLink boolean

Check this attribute to show a link in the subjectbar that, when clicked, tells the calendar to display events with "pending" state as well as the other events (published, and visible if includeReviewStateVisible has been selected). The link is not a toggle; to get out of the mode where the pending events are showing, simply click any other link on the calendar.

This link ONLY shows up for Calendar Managers. Who is a Calendar Manager? User status as a Calendar Manager is determined by the isCalendarManager.py script. It is easily customized, but as a default is set to allow users with the "Manager" role. If this role is adequate for you, leave this script as is. An example is included in the script to show how to look up group membership to determine Calendar Manager status.

showOnlyEventsInMonth boolean

Check this attribute to restrict the Month view to display events ONLY in the current calendar month, and NOT those events that occur in the days before or after the month begins and ends (ie., if the month view shows the 30th and 31st of the previous month on the calendar, events will NOT be displayed for those dates).

labelEventsOnlyAtStart boolean

Check this attribute to put labels on the month view ONLY on the first day of an event that lasts multiple days. Default is SELECTED. Unselect this attribute if you'd like the event title and datestring to appear on each calendar day that the event is on (ie., a four day event will have the label show up on the calendar four times, on each of the four days of the event).

NOTE! Disabling this (to show events on EVERY day) will only work if the showHighlightFullEvent property is turned ON (selected). It would make no sense (to me) to have the event labeled, but not highlighted. So make sure you use these together. No harm if you don't, but it won't behave the way you might have expected. It pays to read the documentation.

listOfSubjects lines**restrictToThisListOfSubjects boolean**

Together, these two attributes allow you to control the choice of what categories of events to display on your calendar.

List of the Subjects in your CalendarX, for use in creating the macro that displays them on your calendar.

1. LEAVE "listOfSubjects" BLANK, if you want to just use the list of Subjects that is available from already created Events your Plone site.
2. LIST SUBJECTS ONE PER LINE, exactly as they are present in your portal_metadata, in the order you want them displayed. The default values included here are the default values that come with CMF Event and AT Event types.

If "restrictToThisListOfSubjects" is checked, a query for "ALL" subjects is restricted to the Subjects in your listOfSubjects attribute.

If unchecked, "ALL" will return all events found, regardless of their Subject.

Use of this feature allows you to segregate certain events pertaining to certain Subjects to unique calendarx instances.

This also means that if checked, the calendar WILL NOT pick up events that do not have a Subject selected.

ADVANCED FEATURE: Each line in the listOfSubjects can also be a Comma-Separated-Values list (CSV) where each line becomes a list of subjects for viewing. This becomes very useful in the case where you have many Subjects, but would like to combine several of them at a time and use a Nickname (or abbreviation, or acronym) to show up in the Subject menu.

For example: In a calendar for a school with five grade levels, and

four classes of children in each grade level, you could try something like this in your listofSubjects:

```
Class1a,Class1b,Class1c,Class1d
Class2a,Class2b,Class2c,Class2d
Class3a,Class3b,Class3c,Class3d
Class4a,Class4b,Class4c,Class4d
Class5a,Class5b,Class5c,Class5d
```

and then use this in the listofSubjectTitles below:

```
Class 1
Class 2
Class 3
Class 4
Class 5
```

In this way, your subject menu is less cluttered, but it is easy for your users to check one of these to filter and see only the events for children of Class 1 age. You may also want to use SubCalendars with this, so that users can drill all the way down and see ONLY events associated with Class1c, Class4b, etc.

eventTypes list

restrictToThisListOfTypes boolean

Together, these two allow you to restrict what types of content objects will be picked up on your calendar. Put one portal_type per line in the eventTypes attribute, and check "restrictToThisListOfTypes" if you wish this feature to be activated. If unchecked, no check is done on the Type index, regardless of the content of the eventTypes attribute. Usage: For example, this feature means you can create a new Event type for certain users, and then use getNotAddableTypes.py to restrict which users can add those special Event types, which gives even more control over different calendar instances in your Plone site.

This is very useful in subcalendars, where each subcalendar may have a different event Type.

listOfPaths list

restrictToThisListOfPaths boolean

Together, these two allow you to restrict where (the paths) to event content objects that will be picked up on your calendar. This means you can restrict viewing to Events found in certain folders. To use this feature, use a full path exactly as found in your path index. An example:

```
/clients/companyplonesite/Members/fred
/clients/companyplonesite/staff
```

These two paths represent folders where Events can be stored that will show up on the calendar, if restrictToThisListOfPaths is checked. ONLY those events in these two paths will be found. Events in fred's personal folder and events in the staff folder, and any folders deeper than that will be picked up for display on this calendar. For example, if there is a meetings folder inside the staff folder, events inside that folder will also be displayed. If you are having any trouble with this property, please go to the portal_catalog, click on the Catalog tab, and find one of the events that *should* show up on the calendar. Look near the bottom of the page to see what path is being indexed by the "path" index, and use that as the path to the folder that you will use in listOfPaths.

restrictToThisFolder boolean

This property restricts the calendar so that events are ONLY shown if they are found within or beneath the parent folder of the CalendarX instance.

Example: add a CalendarX instance as /Members/lupa/cal. If you set this property to true, then only events found within /Members/lupa and any subfolders therein will be shown on this calendar instance.

This could also be accomplished manually with the restrictToThisListOfPaths property, but this property helps in the special case where you want to allow your users to create a private calendar for their own area. In that case, you should probably set this property to "1" in the property sheet on the filesystem, so that all calendars created by your users will have this property by default.

ADVANCED NOTE: as currently implemented, the restrictToThisFolder option trumps (overrides) the similar restrictToThisListOfPaths property. In other words, in the four query scripts, the restrictToThisListOfPaths property is evaluated first, and if restrictToThisFolder is also selected, the second one (restrictToThisFolder) takes precedence and overrides the first property. To change this precedence, simply go into these query scripts and rearrange the two-line calls for each one so that their order is reversed.

listOfSubjectTitles list

useSubjectTitles boolean

Together, these two attributes allow you to use sensible (e.g., shorter) titles on the Subject Bar for your Subject categories. If your Subject category names are long, three or four subjects can produce an unwieldy list for your users to select from. Instead, use these attributes to include a list of shorter titles for each of your Subjects.

IMPORTANT Be sure to use this feature in conjunction with the "listOfSubjects" attribute above. Be certain that both lists have the

exact same number of entries, so that there is a single corresponding SubjectTitle for each Subject. If these do not match, the calendar may show an error. Simply test your calendar after any changes to this attribute to be certain that your calendar is working without error.

3. CX_props_css

CX_props_css controls the many adjustments that you can make to control the colors and fonts for displaying text in various parts of the calendar. These properties are used in the calendar.css and other CSS files.

Use the Properties tab to adjust the colors and fonts and such throughout the calendar.css and other CSS files.

Default values (include below) are set to nearly match those of a stock Plone 2.0.x install. Or if not match, at least nicely complement. It isn't perfect, but overall things work OOTB. The font sizes are all in percentages instead of fixed sizes, so that the calendar fonts can resize with the rest of the Plone site if the small Normal LARGE style-sheet widget is used.

=== List of Attributes ===

title *string*

Don't mess with this title. Leave it alone. [actually, I don't care.]

**** **Section 1:** ****

VIEW: These attributes control aspects of the "view" tabs, (month, week, day, etc. view template names) right at the top where you choose which view of the calendar is showing.

viewTabsBorderColor default = #8cacbb

View tabs border color.

viewTabsBackgroundColor default = #dee7ec

View tabs background color.

viewFontBaseSize default = 95%

View tabs font size.

viewFontFamily default = "Lucida Grande", Verdana, Lucida, Helvetica, Arial, sans-serif

View tabs font family.

viewTabsFontColor default = #436976

View tabs font color.

**** **Section 2:** ****

Subject Bar: These attributes control aspects of the subject bar, where the Private/Public and MetaCalendar Subject choices are found.

subjectBarBorderColor default = #8cacbb
Color of the border around the subject bar, set in calendar.css. Also goes around the My:Public choices.

subjectBarBackgroundColor default = #dee7ec
Color of the background for subject bar and My:Public bar.

subjectFontFamily default = "Lucida Grande", Verdana, Lucida, Helvetica, Arial, sans-serif
Font family for the subject bar and My:Public bar.

subjectFontSize default = 97%
Font size for the subject bar and My:Public bar.

subjectBarFontColor default = #436976
Font Color for the for subject bar and My:Public bar.

**** **Section 3:** ****

Header: These attributes control aspects of the header area, where the previous and next date arrows, and the calendar date are displayed, at the bottom and top of the calendar. Code for this is generated in the "prevnextcurrentlinks" macro.

headerCenterFontSize default = 135%
"July 2004" header font size.

headerSideFontSize default = 93%
"previous" and "next" links font size.

headerFontFamily default = Verdana, Helvetica, Arial, sans-serif
Font Family name for "prevnextcurrentlinks" macro (prev, next, date header, footer)

headerFontColor default = #436976
Color of the font for header.

headerHeight default = 35px
Height added to base for "prevnextcurrentlinks" macro (prev, next, date header, footer)

headerMarginBottom default = 15px
Bottom margin pixels for "prevnextcurrentlinks" macro (prev, next, date header, footer)

headerMarginTop default = 5px
Top margin pixels for "prevnextcurrentlinks" macro (prev, next, date header, footer)

**** **Section 4:** ****

Continuing: These attributes control aspects of the Continuing Events section, where events that start before or extend across the entire period of view are shown.

continuingHeaderFontSize default = 90%
Continuing events box, header font size

continuingHeaderFontFamily default = Verdana, Helvetica, Arial, sans-serif
Continuing events box, header font family

continuingOuterBorderColor default = #B3CFD9
Continuing events box, outer border color

continuingOuterBorderWidth default = 1px
Continuing events box, outer border width

continuingHeaderBorderColor default = #436976
Continuing events box, inner border color

continuingHeaderBorderWidth default = 1px
Continuing events box, border width

continuingHeaderBackgroundColor default = #8CACBB
Continuing events box, background color

continuingRowEventBackgroundColor default = #DEE7EC
Continuing events box, color if an event is present

continuingRowNoEventBackgroundColor default = #F7F9FA
Continuing events box, color if no event

continuingRowHeight default = 5px
Continuing events box, row height, added to event bottom

**** **Section 5:** ****

Cal: These attributes control aspects of the Main Calendar display.

***CONTROL OF THE MAIN CALENDAR

calBorderColor default = #B3CFD9

Main calendar, border color

calBorderWidth default = 1px
Main calendar, border width

***CONTROL OF THE TR TAGS

calTableRowOddBackgroundColor default = #F7F9FA
Main calendar, for certain views where odd/even vary in color,
this controls the ODD row (in TR tags).

calTableRowEvenBackgroundColor default = #DEE7EC
Main calendar, for certain views where odd/even vary in color,
this controls the EVEN row (in TR tags).

***CONTROL OF THE TH TAGS

calTableHeaderBackgroundColor default = #8CACBB
Main calendar, TH tags background color.

calTableHeaderBorderColor default = #436976
Main calendar, TH tags border color.

calTableHeaderBorderWidth default = 1px
Main calendar, TH tags border width.

calTableHeaderFontColor default = #FFFFFF
Main calendar, TH tags font color.

***CONTROL OF THE EVENT FONTS

calEventFontSize default = 85%
Main calendar, TH tags font size.

calEventFontFamily default = Verdana, Helvetica, Arial, sans-serif
Main calendar, font family for the event listings.

calEventPendingTextColor default = #436976
Main calendar, text color for the pending event listings.

calEventPrivateTextColor default = #821513
Main calendar, text color for the private event listings.

calEventPublishedTextColor default = #466A06
Main calendar, text color for the published event listings.

calEventVisibleTextColor default = #436976
Main calendar, text color for the visible event listings.

***CONTROL OF THE TD TAGS (daily cells in the month view, etc)

calTableDataFontColor default = #000000
Main calendar, TD tag text color, but I don't know if it actually controls anything at this time.

calTableDataBorderColor default = #DEE7EC
Main calendar, TD tag border color.

calTableDataBorderWidth default = 1px
Main calendar, TD tag border width.

calTableDataNoEventBackgroundColor default = #F7F9FA
Main calendar, color when a cell has NO EVENT

calTableDataEventBackgroundColor default = #DEE7EC
Main calendar, color when a cell has an EVENT

calTableDataOutOfMonthBackgroundColor default = #FFFFFF
Main calendar, in the MONTH view when the day shown is NOT a part of the month, this controls the background color.

calTableDataOutOfMonthBorderColor default = #F7F9FA
Main calendar, in the MONTH view when the day shown is NOT a part of the month, this controls the border color.

calTableDataOutOfMonthBorderWidth default = 1px
Main calendar, in the MONTH view when the day shown is NOT a part of the month, this controls the border width.

calTableDataSpanDayFontColor default = #000000
Main calendar, in the MONTH view, text color of the date (ie., "3" on June 3 cell).

calTableDataHeightMonthView default = 105px
Main calendar, in the MONTH view, this controls the empty height of a daily cell.

calTableDataHeightDayView default = 35px
Main calendar, in the DAY view, this controls the empty height of a daily cell.

calTableDataHeightWeekbydayView default = 105px

Main calendar, in the WEEKBYDAY view, this controls the empty height of a daily cell.

calTableDataHeightWeekbyhourView default = 30px

Main calendar, in the WEEKBYHOUR view, this controls the empty height of a daily cell.

calTableDataFontSizeHour default = 130%

Main calendar, in the DAY and WEEKBYHOUR views, this controls the font size of the HOUR displayed (ie., "8am" or "13:00")

4. CX_props_custom

CX_props_custom is an empty property sheet that you can use to add new properties if you are a developer customizing CalendarX. By using this sheet, rather than adding new properties to the existing sheets, you should find it easier to upgrade to new versions of CalendarX because your special settings will be here, rather than in one of the new, updated property sheets of the new CalendarX version. The CX_props_custom property sheet will always be empty in the CalendarX distribution.

=== List of Attributes ===

title *string*

Don't mess with this title. Leave it alone. [actually, I don't care.]

[that's all. add new properties yourself inside this property sheet.]

5. CX_props_eventdisplays

Use the Properties tab to adjust the attributes of the Event as displayed on the calendar views. These properties control the icons available for display and also the CSS classes. You can choose to control icons and CSS classes according to the Subject of the Event, or according to the Type of the Event.

=== List of Attributes ===

title *string*

Leave this title attribute alone.

useSubjectIcons *boolean*

listOfSubjectIcons *lines*

If checked, this will cause the views to choose an icon for each event based on the Subject names found in the list. The list consists of a lines attribute where each line consists of a Subject and an icon ID, separated by a pipe (|) character. For example:

```
Work|event_work_icon.gif
```

 where Work is the subject.

Your subject names should (must!) match the actual subjects you use for your events, or this method will not work well. Actually, it will just pull the default event_icon.gif from the Plone skin if there is any problem finding a matching subject name or icon ID.

This property is handy for making your events more visibly recognizable in your calendar page. The default icon size is 16x16 pixels, with some white (or clear) pixel space on the right and left sides. I haven't tested it with larger icons, but keeping to a modest size might be a good idea.

Add your event icons into your /portal_skins/custom folder, or put them directly into your calendar instance folder for (slightly) better performance.

useSubjectCSSClasses boolean

listOfSubjectCSSClasses lines

If checked, this will cause the views to choose a CSS class for each event based on the Subject names found in the list. The list consists of a lines attribute where each line consists of a Subject and a CSS class name, separated by a pipe (|) character. For example:

```
US Holiday|event_usholiday
```

 where Work is the subject, and event_usholiday is the CSS class name.

Your subject names should (must!) match the actual subjects you use for your events, or this method will not work well. Actually, it will just pull the default event_published CSS class from the Plone skin if there is any problem finding a matching subject name or icon ID.

This nicely allows you to apply styles like font color to your event listings according to the Subject of the event. Put your custom styles into your calendar.css stylesheet, or into a customized version of the ploneCustom.css stylesheet if you prefer (the sample ones I've created for default use are found in calendar.css). An example of a CSS class listing I used to use a blue text color for the "Appointment" event subject:

```
A.event_appointment {  
  COLOR: #0000CC;
```

```

    TEXT-DECORATION: none;
}
A.event_appointment:hover {
    COLOR: #0000FF;
    TEXT-DECORATION: none;
}

```

Additionally, if you want the Subjects in the Subject listing at the top of the calendar to reflect these same CSS classes, you have to add these too. Example ones (for Appointment, etc.) are in calendar.css for you to customize. They are in a SPAN tag and look like this:

```

TABLE.caltabs TD.barright2 SPAN.event_appointment {
    COLOR: #0000CC;
    TEXT-DECORATION: none;
}

```

*** ONE MORE NOTE about these. Make sure in your listOfSubjectCSSClasses and listOfSubjectIcons lines properties that you use the actual SUBJECT name and not any Subject nicknames you might use for display (as defined in the listOfSubjectTitles property in CX_props_calendar. Those labels won't work here, only the actual subject will work.

*** AND ONE MORE NOTE. The following two properties (useEventTypeIcons and useEventTypeCSSClasses) take precedence over useSubjectTypeIcons and useSubjectCSSClasses if, for unknown reasons, BOTH have been selected. There's no real reason to select both... only one can work at a time, and I chose to make the EventType ones take priority. Go figure.

useEventTypeIcons boolean

listOfEventTypeIcons lines

If checked, this will cause the views to choose an icon for each event based on the Event Type (portal_type). The list consists of a lines attribute where each line consists of an Event Type and an icon ID, separated by a pipe (|) character. For example:

Event|event_icon.gif where Event is the portal_type.

useEventTypeCSSClasses boolean

listOfEventTypeCSSClasses lines

If checked, this will cause the views to choose a CSS class for each event based on the Event Type (portal_type). The list consists of a lines attribute where each line consists of a Subject and a CSS class name, separated by a pipe (|) character. For example:

AT Event|atevent_class where AT Event is the portal_type, and
atevent_class is the CSS class name.

6. CX_props_popup

Use the Properties tab to adjust the attributes of the rollover PopUp text that displays when you rollover each event displayed on the calendar. Just check off the ones that you want to show up.

Technical Note: These values represent the metadata that are stored in the portal_catalog for each event. If you want to display other information about the event that is NOT in the metadata, you can read the howto on the CalendarX.org website that explains how you can add other attributes as well.

==== List of Attributes ====

title *string*

Leave this title attribute alone.

showPOPTitle *boolean* Checked = Show this info.

Whether to show the "Title" of the event.

showPOPType *boolean* Checked = Show this info.

Whether to show the "Type" string, telling what type of event object is showing up in the calendar.

showPOPSubject *boolean* Checked = Show this info.

Whether to show the "Subject" of the event. Currently implemented to show all the Subjects that are associated with an event -- because events can have more than one Subject selected.

showPOPStart *boolean* Checked = Show this info.

Whether to show the "Start" date and time of the event.

showPOPEnd *boolean* Checked = Show this info.

Whether to show the "End" date and time of the event.

showPOPCreator *boolean* Checked = Show this info.

Whether to show the "Creator" (Plone username) for this event.

showPOPCreated *boolean* Checked = Show this info.

Whether to show the "Created" date of the event.

showPOPModified *boolean* Checked = Show this info.
Whether to show the "Modified" date of the event, when the last time this event was edited.

showPOPState *boolean* Checked = Show this info.
Whether to show the review "State" of the event, such as published, visible, etc.

showPOPDescription *boolean* Checked = Show this info.
Whether to show the "Description" of the event.

7. CX_props_subcalendar

Subcalendars allow for interesting possibilities for special calendars, especially very busy ones. A subcalendar is a folder inside (nested) in a Main calendar folder. I have only tried a single level of nesting (one main calendar with multiple subcalendars), and these controls focus on menu behavior reflecting that use case.

The properties in this sheet control the basic behavior of subcalendars.

The first two properties are used by the main calendar, and the second two are used by subcalendars. These properties have been developed with a Resource Scheduling calendar application in mind, but other possibilities are certainly possible. Your suggestions (and code) for more options are gratefully accepted.

ALSO: You will need to carefully consider what properties to use in your CX_props_calendar property sheets for both the MAIN and SUB calendars. Examine the Resource Scheduling calendar example provided in order to see what some of the possibilities are.

=== List of Attributes ===

title string

Leave this title attribute alone.

useSubCalendarSubjectMenu boolean

For MAIN Calendars: Check this property to signal that (1) there are subcalendars below and (2) hence, use the special Subcalendar Menu for the Subject Links that allows you to both (either) filter on the subcalendars as well as click on the Subject (subcalendar name) to drill down and view that subcalendar alone.

listOfSubCalendars **lines**

For MAIN Calendars: This is a list (one per line) of the names of the subcalendars. The menu choices uses this list for display of links to the subcalendars.

isSubCalendar **boolean**

For SUB Calendars: Check this property if this calendar folder is a subcalendar. This controls the style of menu displayed for subcalendars versus non-subcalendars.

nameOfSubCalendar **string**

For SUB Calendars: The name of this subcalendar. This is displayed in the Subject Links area.

Appendix C: Guide to python scripts in CalendarX

Python scripts perform the bulk of the logic in handling calendar queries, manipulation of objects, and so on. A Python script is conveniently customizable by Developers from within the ZMI, or they can be customized on the filesystem. For example, if desired a Developer can create a customized filesystem version of CalendarX complete with changes to the Python scripts that control where and how the queries are made to find events for the calendar. Then any (all) CalendarX instances within the portals using this custom CalendarX product will utilize the custom queries. If the Developer wants one instance of CalendarX to use a different version of the query script, then by simply placing a customized version of the desired Python script inside the CalendarX folder, this local script instance inside the CalendarX folder will override the default behavior and use the new Python code instead.

We'll start with (1) a full listing of all the Python scripts used in CalendarX, followed by (2) examination of how a page template uses one Python script to return values, and then (3,4,5) we'll take a closer look at a couple of the CalendarX Python scripts and see how they work. By examining them closely you can see how queries are assembled before calling the `portal_catalog`, and you can learn how we call the CalendarX property values out of the property sheets and macros out of other page templates.

1. Alphabetical listing of Python scripts in CalendarX, with brief job descriptions:

`getAddNewEventURL`.

Returns a URL for the Add New Event link based on property sheet values

`getCXAttribute`

Returns an attribute (property) from the appropriate CalendarX property sheet.

`getCXEventsBefore`

Chooses which means of querying the catalog for Events before a given date.

`getCXEventsBetween`

Chooses which means of querying the catalog for Events between two dates.

`getCXMacro`

Returns a path to the macros from within the CalendarX default skin property sheets.

`getDaysOfMonth`

Returns the number of days in a given month.

`getDaysOfTheWeek`

Returns list of names of the seven days of the week, in a proper order for use on the calendar month view, or week view. Starting day is set in `CX_props_calendar`.

`getDictCommon`

Returns a dictionary of useful objects for the calendar views

`getDictDay`

Returns a dictionary of useful objects for the Day view

`getDictMonth`

Returns a dictionary of useful objects for the Month view

`getDictWeekbyday`

Returns a dictionary of useful objects for the Weekbyday view

`getDictWeekbyhour`

Returns a dictionary of useful objects for the Weekbyhour view

`getEndOfDay`
Returns a `DateTime` object for the end of the Day view showing on the calendar (defaults to one second before midnight). Hour range: 1-24.

`getEndOfMonth`
Returns a `DateTime` object for one second before midnight on last day of the Month

`getEndOfWeek`
Returns a `DateTime` of the last second of the day of the end of the calendar week.

`getEventDictDay`
Returns a dictionary of useful objects for Events for the Day view

`getEventDictMonth`
Returns a dictionary of useful objects for Events for the Month view

`getEventDictWeekbyday`
Returns a dictionary of useful objects for Events for the Weekbyday view

`getEventDictWeekbyhour`
Returns a dictionary of useful objects for Events for the Weekbyhour view

`getEventIcons`
Returns an icon object from the skin based on the Subject or Event Type

`getEventsBeforeAQ`
Queries catalog to retrieve events between two dates using `AdvancedQuery`.

`getEventsBeforeZC`
Queries the catalog for Events before a start date using `ZCatalog` query dictionary.

`getEventsBetweenAQ`
Queries catalog to retrieve events between two dates using `AdvancedQuery`.

`getEventsBetweenZC`
Queries the catalog for Events between two dates using the `ZCatalog` query.

`getSubjectCSSClasses`
Returns a CSS class name based on the event Type (portal type)

`getMonthName`
Returns a month name given a month integer, or a month list

`getNumOfDays`
Returns an int for number of days or dates between start and end. a "day" is 24 hours, a "date" is calendar day (midnight)

`getNumOfHours`
Returns integer number of hours from start of calendar to event. works for weekbyhour or day view.

`getStartOfDay`
Returns a `DateTime` object for the beginning of the Day view showing on the calendar (defaults to midnight).

`getStartOfMonth`
Returns a `DateTime` object for midnight on 1st day of the Month

`getStartOfMonthToShow`
Returns a `DateTime` for the first of the Month view showing on the calendar (end of the previous month, usually).

getStartOfWeek
 Returns a `DateTime` object for the start of the first day of the week, for use on the calendar month view.

getSubjectCSSClasses
 Returns a CSS class name based on the event Subject

isCalendarManager
 Returns True (1) if current user is considered a "Calendar Manager"

listUnique
 Returns list of unique values from a list of values not unique and if `listOfSubjects` is True, creates a unique list of subjects from that attribute

makeCSV
 Returns a CSV (comma separated variables) string from a list.

queriesSubtract
 subtracts `q2` from `q1`: looks for objects in `q1` that are NOT in `q2` and builds a `q1new` list of those objects, and returns `q1new`

queriesUnique
 Returns Brain of unique events from a Brain of events not unique

2. A brief overview of Python script use in CalendarX view templates:

CalendarX strives to make as much of the functionality of the calendar closely accessible to the Plone developer for customization. That's why so little of the functional Python code is located in the `CalendarXFolder` class, and so much of it is present in the `/portal_skins/CalendarX` skins folder. From there, you can easily customize the behavior of your CalendarX instances, and even make each of several CalendarX instances behave differently from the others.

Here's the basic story of how CalendarX creates a view.

1. Call the URL
(e.g., `http://mycalendar.biz/calendar/month? currentDate=2005/01/10&xmy=0&xsub=ALL`)

This calls the `month.pt` template and initiates the view generation process with three calendar parameters passed through the URL: `currentDate`, `xmy`, and `xsub`.

2. The month page template gathers a number of variables together at the beginning that can be used throughout the page template ('global' variables). Code for this looks like this:

```

<body>
<!-- Defining global variables -->
<div metal:fill-slot="main"
    tal:define="
        MODIFIED string:mod 0.4.12 use calendarPrint.css;
        viewname string:month;
        DateTime python:modules['DateTime'].DateTime;
        Dict python:here.getDictMonth();
        url here/absolute_url;
        ampm Dict/ampm;
        etc....
  
```

One by one, here's what those lines do:

1. `MODIFIED` defines a string... in this case it serves simply as a comment. Each of the page templates has a similar string at the beginning to show how it differs from previous versions of the template. Another common string I use in page template code is `"COMMENT"`.
2. `viewname` defines a string `'month'`. This parameter is passed to other scripts for `'month'` specific needs.
3. `DateTime` defines a Python module called `'DateTime'`. This is used as an abbreviation so that wherever else (such as macro code) that we need to call a `DateTime` function, the module will be available for us.
4. `Dict` defines a dictionary of useful information. There are many parameters that get defined in each of the templates, and some of these are common to all the templates. What we do here is call a template-specific Python script called `getDictMonth.py` that in turn calls another script called `getDictCommon.py`, and this returns a dictionary of many variables, as you will see in the following statements.
5. `url` defines the URL of the current page by calling `here/absolute_url`.
6. `ampm` defines a Boolean (`True/False`) value based on the value of the `'hoursDisplay'` property found in the `CX_props_calendar` property sheet (`True` if `hoursDisplay = 'ampm'`, and `False` if `hoursDisplay` equals anything else). This value is returned to the month template from `'Dict'`, the dictionary we got from `getDictMonth` script (and in turn, this value comes from the `getDictCommon` script). In earlier versions of `CalendarX`, these variables were all defined right here in the page template code. In this case the code would have looked like this:

```
ampm python:test(here.getCXAttribute('hoursDisplay') == '12ampm',1,0)
```

and there would have been many, many such definitions one after the other, some more complex than others. Instead, all (well, at least much) such Python logic has been moved out of the template and into a few scripts to clean up the template code and so that such code only needs to be changed in one place (rather than in many page templates).

Cleaning and rearranging the code in `CalendarX` is an ongoing battle/chore/drama. Currently, there are still several places where code could and should be moved out of the page templates and into other Python scripts, but hasn't yet. It doesn't mean `CalendarX` is bad... it just means I haven't gotten to everything yet. `CalendarX` is not, by the way, model code for how to build Products in Plone.... it is full of less-than-best practices. But with each new branch of the code I try to make improvements in the architecture, and get more good advice. Meanwhile, it works, and I'm happy to maintain it and make gradual improvements.

- 3. `getCXAttributes.py` -- How CalendarX gathers properties from the property sheets.**

- 4. `getCXMacro.py` -- How CalendarX finds its macros.**

- 5. How CalendarX queries for events (`getCXEventsBetween.py` and friends).**

Appendix D: Guide to page template views and macros in CalendarX

There are four views in CalendarX: Month, WeekByDay, WeekByHour and Day. Each template has a similar overall style, with a number of defined variables at the top, followed by calls to CSS and Javascript files, and then code to generate the views. Much of the detailed code has been pulled out of these templates into macros and Python scripts to make it easier to maintain and manage. Additionally, most of the important features of the views that can be made configurable have been pulled out and put into configurable properties to minimize the amount of customization required to create a unique calendar for your Plone site. These properties are described in Appendix B. The Python scripts are described separately in Appendix C.

Macros are simply page template code snippets that are called in from the view templates -- this separates the code so that it only needs to be changed in one place, rather than in each template, when a common change should propagate through all the views. The macros for CalendarX are all located in `CX_props_macros.pt`, in the `/skins/CalendarX` folder of your CalendarX product distribution, and on your site this can be found and customized from the `/portal_skins/CalendarX` folder. Here is a brief description of these macros in the order they are found in `CX_props_macros`, with a brief description of each macro's purpose.

1. `samplemacronamehere` - a sample macro that is unused anywhere. Copy and paste this to use as a starting point for macro code of your own.
2. `caltabsforviews` - the view tabs at the top of the calendar: month, weekbyday, weekbyhour, day, help.
3. `prevnextcurrentlinks` - the text and links at the top/bottom of calendar: previous month, next month, the Date string (e.g., June 2006), and the Jump To Date widget.
4. `monthdaysofweek` - orders days of the week properly for the month view
5. `hoursdisplay` - displays hours on the day, weekbyhour views.
6. `popuptextbox` - generates the rollover popup text boxes for each event. You can control what appears here through the `CX_props_popup` property sheet, and you can further customize it here, if desired.
7. `eventlister` - displays the event text and link in the calendar cells for each event. This macro runs once for each event listed on a view.
8. `copyright` - Powered by CalendarX link... comment the code out here if you don't want to show this on your website ;-)
9. `subjectlinks` - shows choices for Subject categories, plus other widgets. This is a long, complicated macro. In the new 0.5 branch I've created several versions of this macro so that you can choose different layouts just using the property sheets. You could do the equivalent here... just copy the entire macro and rename the original one `subjectlinks_ORIGINAL`. Then modify your new `subjectlinks` macro and the views will pick up the new one here.

Read through Appendix C on Python scripts to learn how `getCXMacro.py` helps find the macro needed.

Month view:

The Month view displays a month of events at one time, with a standard calendar appearance (4 to 6 weeks of seven days in each week). Events within each day are arranged in order of starting time.

There are several configurable properties that ONLY affect the month view, using the CX_props_calendar property sheet (see Appendix B for details on each):

- showOnlyEventsInMonth: this controls whether out-of-month events are shown or not (e.g., events on January 31 for a February calendar view)
- labelEventsOnlyAtStart: this controls whether the title of a multi-day event is shown on each of the days, or just on the first day of the event.

Here is a typical month view:

The screenshot displays the CalendarX for iPhone web interface. At the top left, the logo "CalendarX for iPhone" is visible, along with the version "Running: 0.4.12(stable)". A search bar is located at the top right. Below the header, there are navigation links: "home", "the guide", "news", "forums", "issues", "links", "about", and "members". A secondary navigation bar includes "Home", "my folder", "my preferences", "junks", "phone setup", and "log out".

The main content area shows a month view for January 2005. The calendar grid has columns for days of the week (Monday to Sunday) and rows for dates. Events are listed within the grid cells. For example, on January 1st, there is an event "CalendarX 0.4.12(stable) is released" from 05:17 am to 05:45 am. On January 7th, there is a "Meeting with David" from 12:00 am to 11:00 am. On January 10th, there are two events: "CalendarX 0.4.12(stable)" from 01:30 am to 01:30 am, and "CalendarX 0.3.9(beta) is released" from 06:15 pm to 07:40 pm. On January 16th and 17th, there are "Happy New Year" events from 06:27 pm to 08:00 pm. On January 17th, there is a "Workshop" from 10:00 am to 10:00 am. On January 14th and 15th, there are "Jan 14 - Jan 15" events. On January 14th and 15th, there are "Jan 14 - Jan 15" events.

At the bottom of the calendar grid, there are navigation controls: "<< previous month", "January 2005", "2005", "Jan", "22", "Jump", and "next month >>".

At the very bottom, it says "Powered by CalendarX, a iPhone Calendar."

WeekByDay view:

The WeekByDay view displays a week of events at one time, with each day having one block of events, arranged in the same order as the starting time of each event. The starting day of the week (Sunday, Monday, etc.) is configurable by the Administrator, using the dayOfWeekToStart property of the CX_props_calendar property sheet.

Here is a typical weekbyday view:

The screenshot shows the CalendarX for Linux web interface. At the top left, it says "CalendarX for Linux" and "Running: 0.4.12(stable)". Below this is a navigation menu with links like "home", "the goods", "news", "howto", "issues", "faq", "links", "members". A search bar is located at the top right. The main content area is titled "weekbyday" and shows a weekly calendar for January 9, 2005, to January 15, 2005. The days are listed from Sunday to Saturday. Events are shown as colored blocks on the calendar days. For example, on Monday, January 10, 2005, there are two events: "CalendarX 0.4.11(stable) (start: 01:20 pm - Jan 10, 2005 | end: 01:55 pm - Jan 10, 2005)" and "CalendarX 0.3.4(dev) is released (start: 09:15 pm - Jan 10, 2005 | end: 07:48 pm - Jan 10, 2005)". On Friday, January 14, 2005, there is an event "aaa (start: 08:47 am - Jan 14, 2005 | end: 08:58 am - Jan 15, 2005)". On Saturday, January 15, 2005, there is an event "aaa (start: 08:47 am - Jan 14, 2005 | end: 08:58 am - Jan 15, 2005)". The interface includes a sidebar with "new & updated" items, a "previous week" and "next week" navigation, and a "Powered by CalendarX for Linux CalendarX" footer.

WeekByHour view:

The WeekByHour view is the most complicated view... it displays a week of events arranged hour by hour through each day. In this respect it is a bit like seven day views arranged side by side. The view can be set to show the entire 24 hour day (midnight to midnight), or in the default setting from 8am to 8pm, or for any set interval (using the dayViewStartHour and dayViewEndHour properties in the CX_props_calendar property sheet). The listing is arranged in one-hour blocks, and multiple events within the hour are arranged in order of starting time.

The starting day of the week (Sunday, Monday, etc.) is configurable by the Administrator, using the dayOfWeekToStart property of the CX_props_calendar property sheet. The hour display on the left side of the view can be configured as 12 hour (am/pm) display, or as a 24 hour display (using the hoursDisplay property in CX_props_calendar). Events that start on this day but before the starting display time (e.g., a meteor shower event at 4am) will appear in the Continuing events block at the top, and later events (late parties) will show up at the bottom of the page in a Later events block. Rolling over these events with the mouse will highlight the appropriate day they start on.

Here is a typical weekbyhour view:

The screenshot shows the CalendarX for Home web application in the WeekByHour view. The interface includes a navigation menu with links like 'home', 'the goods', 'news', 'features', 'screens', 'links', 'about', and 'members'. A search bar is located at the top right. The main calendar grid displays a week from January 9, 2005, to January 15, 2005. The grid is organized by hour on the left side, ranging from 8 am to 7 pm. Two events are visible: 'CalendarX 0.4.11(stable)' on Monday, Jan 10, from 01:33 pm to 01:53 pm, and 'CalendarX 0.5.0(dev) is released' on Monday, Jan 10, from 06:13 am to 07:43 am. A tooltip for the 01:33 pm event shows its subject 'conversion' and description: 'CalendarX-0.4.11(stable) was released Monday. It provides non-critical bugs that also prevents events to show up if naming a Personal calendar, and that also certifies in web view CSS elements to be properly rendered by the user's browser from the CSS property sheet. That's all we could find broken in the first 10 days of 0.4.11.' The footer of the page reads 'Powered by CalendarX, a Home Calendar.'

Day view:

The Day view displays one day's worth of events. The view can be set to show the entire 24 hour day (midnight to midnight), or in the default setting from 8am to 8pm, or for any set interval (using the `dayViewStartHour` and `dayViewEndHour` properties in the `CX_props_calendar` property sheet). The listing is arranged in one-hour blocks, and multiple events within the hour are arranged in order of starting time.

The starting day of the week (Sunday, Monday, etc.) is configurable by the Administrator, using the `dayOfWeekToStart` property of the `CX_props_calendar` property sheet. The hour display on the left side of the view can be configured as 12 hour (am/pm) display, or as a 24 hour display (using the `hoursDisplay` property in `CX_props_calendar`). Events that start on this day but before the starting display time (e.g., a meteor shower event at 4am) will appear in the Continuing events block at the top, and later events (late parties) will show up at the bottom of the page in a Later events block.

Here is a typical day view:

The screenshot shows the CalendarX for iPhone web interface. At the top left is the logo "CalendarX for iPhone" with a version indicator "Running: 0.4.12(stable)". A search bar is located at the top right. Below the logo is a navigation menu with links: "try it", "home", "the goods", "news", "howtos", "issues", "links", "sites", "members". A breadcrumb trail reads "you are here: home > calendarx-0.4.12 stable demo > day".

The main content area is titled "day" and shows the date "January 10, 2005 (Monday)". Navigation controls include "previous day", "next day", and "Jump" buttons. A table displays events in one-hour blocks:

Hour	Events
8 am	
9 am	
10 am	
11 am	
12 pm	
1 pm	CalendarX 0.4.11(stable) (start: 01:30 pm - Jan 10, 2005 end: 01:50 pm - Jan 10, 2005)
2 pm	
3 pm	
4 pm	
5 pm	
6 pm	CalendarX 0.5.0(dev) is released (start: 06:15 pm - Jan 10, 2005 end: 07:40 pm - Jan 10, 2005)
7 pm	

At the bottom of the interface, it says "Powered by [CalendarX, a Plone Calendar.](#)"

Appendix E: Guide to CSS and JavaScript code in CalendarX

CSS is used in CalendarX. So is JavaScript. This is good.

Without touching the page templates, you can configure nearly every visible color and font shown in CalendarX. That's what the plethora of properties in `CX_props_css` is all about. Try it, you'll see.

The JavaScript is much more complicated. Even I'm not sure what everything does, and I've been over it all and through it more than twice or thrice.

For more good information, read the source code. Thank Limi for Plone CSS, and Oliver for PloneCalendar CSS, and thank Oliver for PloneCalendar JavaScript, which together make nice popup text boxes and rollover highlighting quite possible.

[[more to come...]]

Appendix F: About Python, Zope, Plone, and CalendarX 0.5 branch

Python is a programming language designed for clean object oriented programming with high-level commands, a rich collection of libraries, and an elegant syntax that allows rapid development. Python is an open source project.

Zope is an application server built on the strength of the Python language, with authentication management and an object database that is ideal for web development. Zope provides a powerful base for rapid and maintainable web application development that scales well with inexpensive hardware. Zope is an open source project.

Plone is an object-oriented content management system (CMS) built on the powerful Zope application server, and featuring rapid development of custom content types and a flexible, standards-compliant web interface. Because of its strong base in Zope and Python, there simply is no other professional quality CMS more flexible and powerful than Plone. Plone is an open source project.

CalendarX is a web-based calendar product that works with Plone portals. CalendarX is pretty good, and it's getting better all the time. CalendarX is an open source project.

And as one raving fan puts it:

"i love calendarx coz...
it does what it says on the tin."

What more can you ask for?

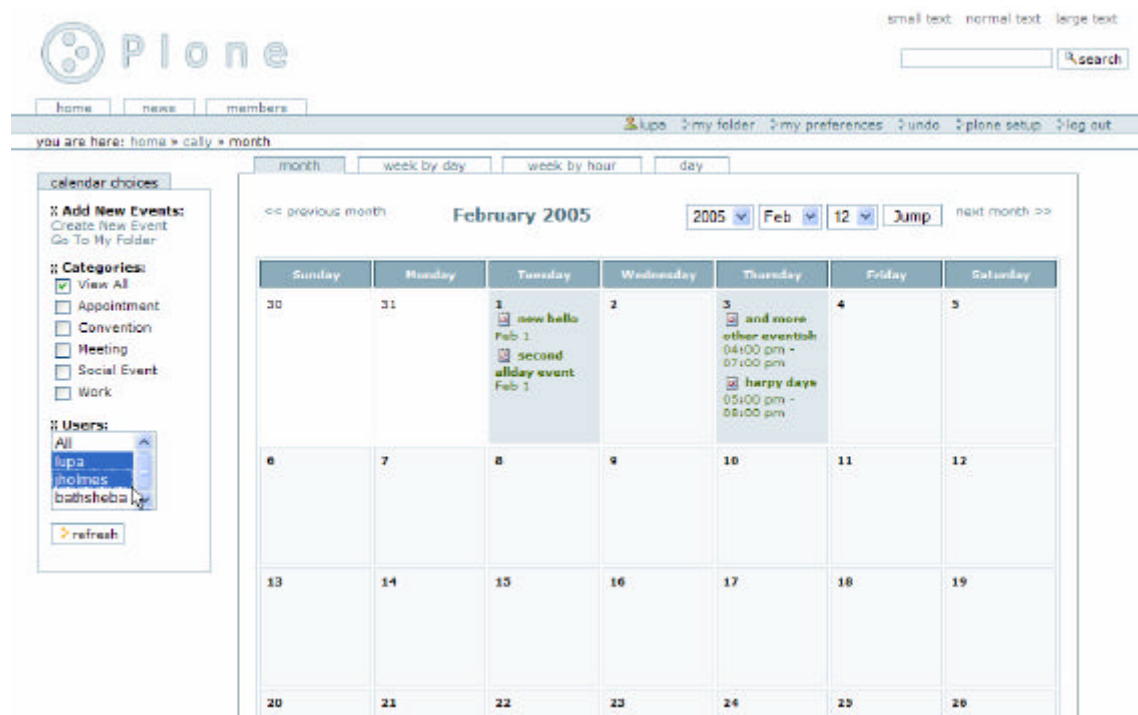
OK... maybe iCalendar integration, hotsync with your Palm or Outlook or Evolution, recurring events, better group functionality, a portlet version, lots more new kinds of views to choose from, clickable icons to let you start a new event from any date or time within the calendar, easily customizable new Event types, PopUp event detail windows instead of the default Plone views, skinning for a standalone calendar that really speeds things up by peeling away some of the Plone extras, maybe a few others.

But, of course, it doesn't say any of those things on the tin.

So, now we introduce the 0.5 branch of CalendarX. All new development is currently focused on the 0.5 branch, currently (mid-Feb 2005) at version 0.5.1(dev). Each release of the development branch is as bugfree as possible (I don't release a tarball until all critical bugs have been squashed), and many times development releases are used in production Plone sites (that was certainly the case for the 0.4 branch, and I think it may already be the case for some 0.5 branch sites). However, I don't recommend investing significant work customizing a production calendar based on the development branches simply because I make no guarantee that the API and internals of the development branch will remain stable into the future... once I make it up to

the alpha or beta stage, you can be pretty certain that nothing major is going to change and any changes to the CalendarX branch beyond that will be bugfixes and non-critical feature improvements. That's my caveat... but that said, there are folks out there (me included) that use the development branch releases in production use all the time, and aren't afraid to jump in and modify things to make it work the way we want it to. Go for it!

Here's a screen shot of 0.5.1 branch of CalendarX, default release with portlet_cx_choices:



Some of the new features in 0.5 branch of CalendarX:

1. Control of calendar choices can use a portlet, cleaning up the calendar header nicely.
2. A month-view portlet_calendarx is available to replace the stock portlet_calendar of Plone, and it draws its behavior from any designated CalendarX instance in your portal.
3. Greatly expanded property selection (nearly 100 new properties) for controlling what events are displayed on your calendar, and who can see them, including some group functionality.
4. A User-Picker widget (shown above) to let you select a few Users and see only their events on the calendar. Group-Picker widget to follow sometime in the future.
5. Speed. CalendarX-0.5 branch is now significantly faster than the 0.4 branch.

No manual yet for the 0.5 branch, but the new features are documented in the /docs folder, just as they are for the 0.4 branch. Happy scheduling!

+lupa+