mdCalendar

Introduction

Readers and advertisers are often impressed with our unique online calendar product. mdCalendar easily allows for the creation of complete and flexible online calendars. It lets the creator build calendars that contain as much or as little information as desired. The creator may also enhance and customize mdCalendar to fit his needs.

Morris DigitalWorks' mdCalendar is the result of years of continuous development and refinement by a media company for a media company.

This white paper provides an overview of the benefits, requirements, and architecture of mdCalendar.

Benefits of mdCalendar

The mdCalendar system supplies developers with all of the necessary tools for capturing, maintaining, and displaying calendar and event data.

The primary advantages of the mdCalendar system are:

- **Multiple inputs:** There are several points of data entry into mdCalendar: ™TimeLink allows for the manual addition, deletion, and editing of events that appear in the calendar. ™Event data can be imported from text files either interactively or as part of an automated batch process. ™Web Form allows individuals in the community to enter an event into the database. You may then use TimeLink to either edit the event and make it active, or omit the event from the calendar. This highly sought-after feature gives *you* complete control over the content in your calendar.

- **Multiple outputs:** Separation of data from presentation allows for opportunities to extensively reuse data in a variety of formats and presentations. These include, but are not limited to, effortless aggregation of calendar events from multiple calendars, converting the calendar data into print format, and repurposing data to exploit new revenue opportunities.

- **Multiple data uses:** mdBusiness, in collaboration with mdCalendar, allows for a global business resource. Specifically, mdBusiness facilitates the insertion of business data into a business database for use in many of MDW's applications, including mdCalendar.

- **Editorial Control**: TimeLink's flexible event entry system provides the ability to modify each calendar event and schedule quickly and easily. It also allows for user-defined metadata fields, also known as calendar features. Such features may include admission price, parking, or age requirement information.

- **Reduced costs**: Most systems require the intervention of highly trained and paid programmers to select, format, and display data dynamically. mdCalendar puts the tools to retrieve and format data into the hands of Web development staff via Template Manager and Morris Templating Language (MTL).

- **Security:** mdCalendar has built-in user account management for controlled access to data modification.

- **Performance:** Most applications or Web pages strive to be data-driven, but the more dynamic a site is, the heavier the requirements are on the database behind it. mdCalendar, however, exploits extensive database tuning and data and page caching.

- **Flexible Scheduling:** Most calendar products only allow event scheduling that occurs in patterns (i.e. every Saturday, every 3rd day of the month, only on weekends, etc.). This is not the case with mdCalendar. With mdCalendar, you have the capability to assign multiple schedules per event, and to assign multiple times per schedule.

Benefits of dealing with MDW

- **Industry Experience:** We know publishing because we are publishing. And we've been our own harshest critics and most demanding customers. mdCalendar is a product we've worked hard on since 2000. And we've gotten it right.

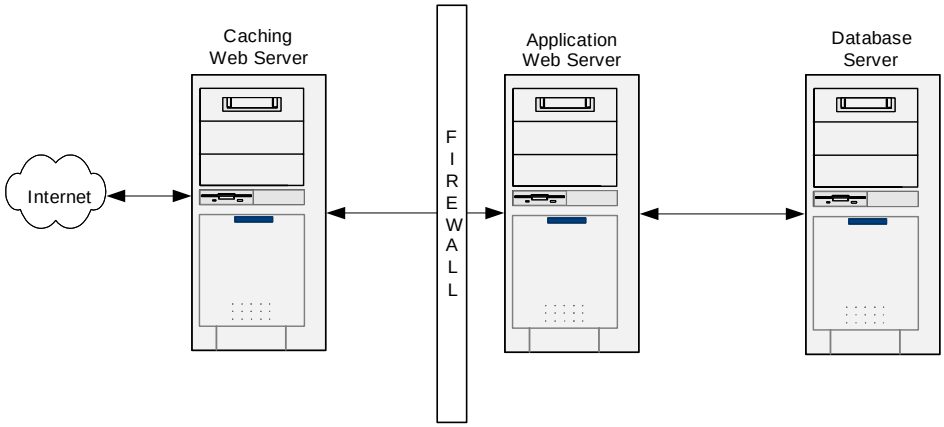
- **Single Vendor Solution:** mdCalendar contains a wide variety of tools, products, and applications, and supports the gamut of calendar needs. The pieces of this package were developed to work together.

- **Maturity of Platform:** As business needs have changed, mdCalendar has expanded to meet those needs with new applications and functionality. But at its core, mdCalendar is a product that has been tested continuously in the real world since 2000.

- **Stability:** mdCalendar was built to answer the needs of Morris' 36 newspapers, 19 shoppers and 20 magazines and specialty publications. MDW has a continuing need for this product and a commitment to its future development and expansion.

System Topology

At this time, please note that there are two methods to running mdCalendar, "ASP" and "Enterprise". "ASP" simply refers to mdCalendar running on our system, while "Enterprise" refers to you running mdCalendar on your system. Either way, the next section describes the platform components we use to run mdCalendar software.



- **Server hardware:**
  MDW uses Hewlett Packard servers. Specifications will vary. There may be some recommended configurations based on client needs.
  - Database server(s)
  - Application server(s)
  - Although not required, MDW recommends caching web server(s).

- **Server-side system requirements:**
  - Oracle Enterprise 9i w/Oracle InterMedia and Java Options
  - Oracle Net9 Client
  - HPUX 11.x or RedHat Linux 7.x or ES 3.0
  - Apache Web server 2.x
  - Java 1.4.

- **Administrative client-side components:**

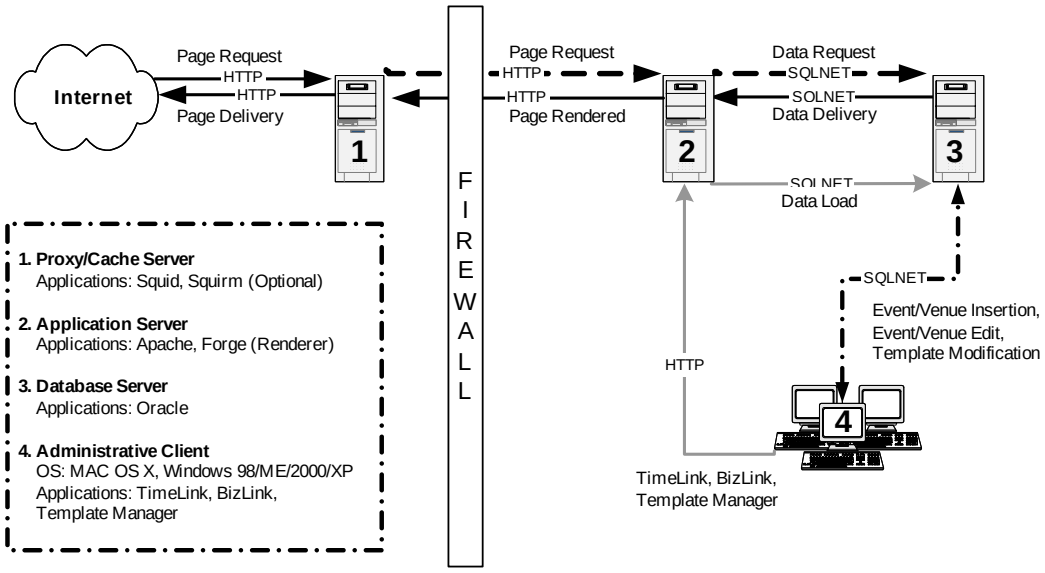
  - mdCalendar's administrative client application, TimeLink, is Java-based. Java runtime environments (JRE - PC; MRJ - Mac) will be required to run these clients.

Utility/Application Overview

The mdCalendar system encompasses all of your data input and rendering. The following is a high-level view of the mdCalendar system. Input is built around the administrative client, TimeLink. There are no minimum requirements for this input, but the richer the event input, the richer the potential output.

Below is a list of the components comprising the backbone of the mdCalendar system and their interaction. The overview contains a short description of each of the components and user benefits.

Example of a typical configuration:



Data Input

- **TimeLink**
  TimeLink resides on one or more Administrative Client machines (4), and lets calendar administrators control their calendar publication (i.e. add, edit, and delete events). Users can search for events or browse by event dates, then edit any facet of the event's data, including text and dates. TimeLink includes the ability to manage account creation and access control for those users designated as super-users. Its built-in security keeps users from even seeing calendars in groups they don't have access permissions for.

- **BizLink**
  All information added to the business database (via BizLink and other mdBusiness applications) is available as venue data in TimeLink.

Data Repository

- **Data Repository**
  The data repository, on the Database Server (3), contains event data, stored procedures for precompiled data retrieval, automated processes for data archiving, and data modification tracking.

Data Rendering

- **Template Manager**
  This resides on one or more Administrative Client machines (4) and allows Webmasters to easily manage a large number of templates written in MTL, MDW's flexible template scripting language (see Appendix for details), by pulling a variety of information from the database.

  MTL (pronounced “MeTaL”) can be inserted into any formatting (HTML, XML, etc.) to control presentation. Template output can be displayed dynamically, constantly calling the newest information directly from the database, or generated to static files on a regular basis.

  TM provides a quick reference to MTL tags, an audit trail for easy template recovery, and user account management. Template Manager is also equipped with a package containing generic, default templates. These default templates contain examples of all the key functionality offered by MTL and the rendering system.

- **Forge**
  The Forge rendering engine resides on the Application Server (2) and processes MTL templates into output, conditionally requesting the necessary data from the Data Repository on the Database Server (3) and formatting the returned records.

  Forge features data caching to reduce load on the database. If the same data is requested two times in a row, Forge only goes to the database once.

  Forge features optimal SQL processing, secure data-entry processing, and static and dynamic page generation as well as email generation.

  Forge transfers files behind the firewall using RPC protocol. It can deliver files locally or to other hosts in the local network and check whether dependent files exist on those local and remote hosts.

- **Internet Requests**

  Page requests from the Internet are received by Squid, which is on the Proxy/Cache Server (1). If Squid has the rendered page for the requested URL in cache, it returns the page immediately to the Web user. If Squid does not have the page cached, it requests the page from Forge, which is on the Application Server (2).

  If Forge has the data required for the page in its cache, it formats the data according to the template's MTL and returns the rendered page to Squid. If Forge does not have the required data, it requests it from the database (3), formats the data according to the template, and sends the rendered page to Squid. (Data and presentation are independent of each other, and multiple templates with variously formatted output might call for the same data. So Forge might have the necessary data in cache from an earlier request, which outputs to a different format.)

  Squid caches the page returned by Forge and sends it to the Web user.

Appendix

MTL was written to put the tools to retrieve and format data into the hands of proficient HTML writers. It was not targeted to programmers, and has therefore been kept relatively simple. It nonetheless contains the tools to author fairly subtle and complex output.

MTL contains three control structures: if, loop, and query. The query structure, which controls the retrieval and output of data, is the heart of MTL.
A simple MTL template:

```
<mtl classifieds.ad property=$property>
     <mtl classifieds.ad.free_text><p>
</mtl classifieds.ad>
```

This example retrieves the first 20 ads from the property or properties specified by $`property`. It consists of an open query tag: `<mtl classifieds.ad property=$property>`, a close query tag: `</mtl classifieds.ad>`, a data tag, and an HTML tag.

The open query tag defines the start of the query control structure and the criteria Forge will use to select ads. The close query tag defines the end of the query control structure. Forge executes everything between the open and the close query tags once for each ad that matches the criteria. In this case, Forge will output the text of the ad, followed by a "<p>."

Pagination is built in to mdClassifieds; so unless the open query tag contains instructions to the contrary, Forge will break the query results into "pages" of 20 ads each.

To be able to get to subsequent pages of ads, or to narrow the selection criteria (from "all ads in $property"), add more search criteria, also known as *query qualifiers*.

```
<mtl classifieds.ad property=$property ads_per_page=$maxrec  page=$page
category_number=$categorynumber classification=$classification>
     <mtl classifieds.ad.free_text><p>
</mtl classifieds.ad>
```

This version makes use of several variables. Typically, the values are passed to the template via key=value pairs in the URL used to access the template. Here's how the arguments look in a URL:

```
…/classifieds-bin/classifieds?property=[your
prop]&maxrec=10&page=2&classification=transportation…
```

Note that no category number has been specified in this URL. Each qualifier in the query is optional. If a value is specified for it, Forge will use it; otherwise it will be ignored.

MTL contains many more query qualifiers and data tags, but a full examination is beyond the scope of this document.