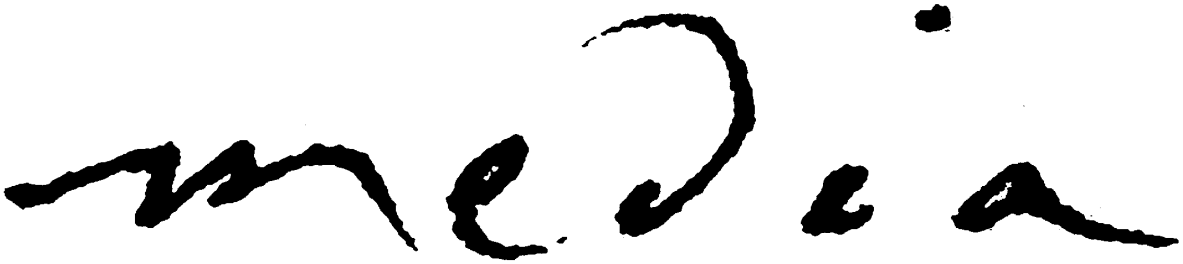
# Performance Tuning

by Theresa Ray of Tensor Information Systems, Inc.

Sponsored by Apple Computer, Inc.
Apple Developer Connection



# Performance Tuning

by Theresa Ray of Tensor Information Systems, Inc.

Performance tuning your WebObjects application is a critical step on the road to successful deployment of a web site. Optimization of a web site is always productive, only costs a moderate amount of development time, and results in a solid, scalable system. By properly coding, testing and administrating your application, your web site can be a great success!

## The importance of performance tuning your application

We've all been irritated by web sites with agonizingly slow response times, or which contain unnecessarily large images that take an eternity to download. You do NOT want your web site to be difficult or frustrating for its users. Proper performance tuning of your WebObjects application can ensure that your users will have a responsive and enjoyable visit to your site

Thorough optimization of a WebObjects application requires many steps. Most applications don't need to be tuned in every possible area. Occasionally, a very large-scale application will require complete top-to-bottom inspection and optimization. This document strives to identify most of the key areas in which a WebObjects application COULD be tuned. You will have to decide which areas of optimization will be most meaningful for your deployment, and implement those areas.

This document is intended to introduce the reader to the topics relevant for performance tuning of a WebObjects application. The details necessary for implementing each topic are thoroughly covered in other documentation, including the WebObjects Developer's Guide and the Enterprise Objects Framework Developer's Guide, and are beyond the scope of this introductory paper. Therefore, appropriate sections include a "recommended reading" list where the reader can find the details necessary to the implement that section's topic.

This document is organized in two sections. Section I focuses on ways to optimize the WebObjects code in order to achieve optimal results. Section II focuses on testing tools and deployment options that will enhance the performance of your web site.

## Section I - Optimizing your WebObjects code

There are many areas in the development of your WebObjects code where choosing the proper configuration can result in an optimally performing web site. By being aware of all the options available and their effect on your application, you can strike a balance between the requirements of your application and optimal performance of your web site.

### Development vs. Deployment Code Techniques

Language Choice - Code written in WebScript is very convenient during development of your application! After making changes to your logic, there is no need to recompile. All you are required to do is reload the page, and voila! Your new code changes have been implemented and are reflected immediately! Well, what's optimal for

development is rarely optimal for deployment. WebScript is interpreted real-time by WebObjects when creating a response or processing a request. As with any interpreted language, scripts run significantly slower than languages that are compiled, such as Java and Objective-C. Therefore, for optimum performance of your web site, you should convert all components which are not extremely minimal in nature from WebScript (or Perl, if you choose to use the third-party WOPerl tool) into Java or Objective-C before deployment.

Component Caching - Additionally, WebObjects by default does not cache its component definitions. Component definitions consist of the .html and .wod files which identify how to render a page within your web site. Caching of these components is off by default so that during development and testing, you are not required to restart the application in order to see changes that have been made to these files. For the vast majority of applications, enabling caching of component definitions before deployment is a highly recommended procedure. To enable component caching in your application, you can invoke the `this.setCachingEnabled(true)` message in the application's *init* method, or by specifing the -c argument on the command line when launching your WebObjects application.

There are a few unusual applications which have very generic components and choose from a set of .html and .wod files in real-time by overriding the *templateWithName:* method. Applications that rely on the *templateWithName:* method implementation should not use component caching.

Logging - During development of your applicaton, you likely used many *logWithFormat:* statements, or had tracing options enabled, in order to gain visibility into your application's behavior without having to invoke the debugger continually. You will want to disable or remove as many unnecessary log statements as you can and turn off all tracing options before deployment, so that your application is not wasting time or file system resources by logging information that is no longer pertinent.

Some logging, of course, is prudent. Most deployment applications have implemented an exception handler to prevent users of the site from seeing the lengthy WebObjects debugging error message when an exception is thrown. Logging the reason for the exception and meaningful variable values in your exception handler will provide insight as to why the users are experiencing difficulty, and will ease troubleshooting of your production application.

## Recommended Reading for Development vs. Deployment Code Techniques

WebObjects Developer's Guide: Chapter Titled "Deployment and Peformance Issues"

## Optimizing HTML

Images - Optimization of HTML is another way to improve the performance of your application. Following good HTML programming practices, such as minimizing the number and size of images within a page, provides a good start on tuning your application. There are, however, some additional areas of analysis necessary for WebObjects programs. Any reusable components that exist in your application must also be considered when determining the size of the page you are returning to your users. You may have minimized all the images on your parent components, but neglected to optimize your navigation bar images, for example.

WORepetitions - When using WORepetitions in your page, you should take a moment to analyze the information contained within the repetition. Is the WORepetition creating table rows for the results of a fetch from the database? If so, you should implement a fetch size limit, or show the results in batches (20 per page, for

example), so that you don't end up trying to return 1000 rows in a table that must be rendered by the user's browser. The WODisplayGroup provides an option to automatically batch-display results from a fetch.

In general, non-optimal HTML contained within a WORepetition can also provide some unwieldy results. For those of you who do not use WebObjects Builder to create your HTML, some web design packages produce notoriously redundant HTML. For instance, instead of setting up the font for the document as a whole, some applications will stick font tags in with every cell for a table. When rows of a table are contained within a WORepetition, the result can be an unnecessarily large HTML file. Even if you use WebObjects Builder to create HTML, it is never a bad idea to take a quick look at the HTML being generated to see if it can be optimized.

### Recommended Reading for Optimizing HTML

WebObjects Developer's Guide: Chapter Titled "Deployment and Peformance Issues"

### Memory Leaks

If your application is written entirely in Java or WebScript, this section does not pertain to you, since WebObjects manages memory allocation and deallocation for these components. If, however, you have developed any of your application in Objective-C, you will want to thoroughly analyze your memory allocation and deallocation, as you are responsible for releasing any memory that you retained. Remember that alloc, retain and copy are all messages that increment the retain count for an object, and which require a release or autorelease message to be sent in order to decrement the retain count and allow the object to be freed.

Forgetting to release the memory associated with an object is termed a "memory leak", and can cause your application's memory requirements to grow over time (how fast depends on the size of the leak), and will result in poor performance. WebObjects provides several tools to assist in memory analysis including MallocDebug for the Mac OS X Server, and ObjectAlloc for NT (found in the Developer/Applications area after installation). Both allow you to observe object allocation/deallocation activity in your application in real time.

Unfortunately, there are no tools currently available for UNIX platforms. However, by monitoring the amount of memory used by your application in conjunction with the number of active sessions, you can get a good indication whether or not you have a memory leak, and how bad that leak is. Finding it requires you to either use MallocDebug on the Mac, ObjectAlloc on NT, or manually review your code.

### Recommended Reading for Memory leaks

WebObjects Memory Management Survival Guide

### Optimizing Database Access

This is a topic worthy of an entire survival guide, and a class on database performance tuning. There are so many details associated with tuning the database access of your application, that they cannot be covered here. However, this guide will address the major topics involved in optimization of database access. For more information, consult additional documentation available for your database, and the recommended reading list below.

Prefetching - Your application will typically require information common for all users, such as a list of states for a name/address form. You may want to prefetch the tables that are used by the application purely for look-up requirements. This will increase the memory required by your application somewhat, but look-up tables are usually fairly small in size, and the time saved by not having to request the information from the database for every user

session can make it a worthwhile investment. Each user session would refer to the application for the common information.

Prefetching can also be useful when a session initializes. There may be information which is not generic enough to prefetch in the application, but which you can identify at the beginning of a session. If it is not a huge amount of information, you may want to consider gathering this information upon the initial request of the user (for instance, information about a previous visit to the site). The response time for the first page of the site will be somewhat slower because of the additional database access, but subsequent requests by that user will be processed faster. Again, it depends on the requirements of your application.

Data model optimization - Optimization of your data model can result in the most significant response time improvement for your application. Particularly for large, complex data models, denormalization can provide very optimal performance. In a normalized data model, all the information is cleanly segmented into relevant tables, using relationships (foreign keys) to join tables containing related information. However, if the information you need is spread across several different tables in the database, the application must perform a fetch for each table in the database from which it needs information.

For example, suppose that you have a data model that has a Customer table, a NameAddress table, a Household table, and a UserProfile table. Unfortunately, some frequently accessed page of your application requires information from all of these tables to display the appropriate interface for the user. The application will have to fetch from each of these tables to gather the information for the display.

In a denormalized data model, all of the information needed to display a single page would optimally be located in a single table in the database. However, this makes the table large, complex, and confusing. Of course, finding the appropriate balance between normalized and denornalized is what you want to do. If you find that you are routinely accessing tables for only one or two attributes, you may want to consider replicating or moving those attributes into another more frequently accessed table.

Helpful hint: You can view or log the SQL statements (the database queries) that EOF generates by opening a shell and typing:

```
defaults write NSGlobalDomain EOAdaptorDebugEnabled YES
```

This command is case sensitive. On NT, this information goes to the Application Event log unless you are running your application in gdb (the debugger) or from a shell. On UNIX and Mach, this information is sent to stdout. Remember to turn this default off when deploying!

Indexing - Ensuring that your database has indexes created for attributes that you commonly use in qualifiers for your fetches is also critical to the performance of your application. In a Customer table, which may contain hundreds of thousands of records, missing one key index on an attribute can result in a query which takes ten minutes or more to execute. Indexes take up space in the database, so you do not want to blindly index every attribute either. You need to analyze the qualifiers that your application routinely uses, and have your database administrator ensure that those queries are optimally tuned.

Fetch size - If you find yourself with queries that take a long time to execute even after optimizing your data model, you may want to investigate using additional EOF options such as supplying fetch hints in your SQL, or canceling a fetch after a certain number of rows is returned.

## Recommended Reading for Optimizing database access

Enterprise Objects Framework Developer's Guide: Section Titled "How Can I Improve Performance?"

## Session management

Session management (or state storage) is one area where the developer can strike an appropriate balance between the application's requirements and optimal performance of the application. A good tuning technique is to store the minimum amount of information required by your application in the session. In an application that will have a large number of simultaneous active sessions, restricting the information stored in the session can make a large difference in the amount of memory used by your application, and thereby increase its performance.

Session storage in memory - Proper choice of state storage can also have a significant effect on an application. By default, WebObjects stores its session information in a memory-resident object in the application. This technique is sufficient for most applications. There is a slight affect on load balancing, and very high-volume web sites might need to consider which option is truly optimal for them.

Load balancing is affected by memory-resident session storage since this technique requires a user of your application to return to the same instance of the WebObjects application as he or she navigates through your site. For example, if you have ten instances of your WebObjects application running, and Mary connects with instance number four on her initial visit to the site, she must return to instance number four as she navigates throughout the site. She cannot move to another instance of the application until she starts a new session.

With the improvements applied in WebObjects 4.0 (including multithreaded I/O, LongRunningResponse pages, improved speed, and the ability to start a user session only when required), the affect of memory-resident session storage has been reduced for even high-volume applications.

Session storage by other means - If you choose to store the session information on the file system, in the database, in hidden fields within the HTML or in a cookie, the user to can connect to any instance of the WebObjects application as he or she navigates through your site. This provides for a truly load balanced system. Of course, these options have both advantages and disadvantages, and storing session information outside of the memory of the application should not be taken lightly (implemented incorrectly, they may end up de-optimizing an otherwise well running application). The WebObjects Developer's Guide is an excellent reference regarding the pros and cons of using these techniques. You have to analyze your application's architecture and your deployment configuration and requirements to determine which mechanism will result in the best possible performance for your application.

Session timeout - Remembering to optimize the timeout for your session is a critical part of optimizing your application. WebObjects 4.0 uses a default session timeout of 3600 seconds, but older versions of WebObjects (3.5 and prior) did not implement a default session timeout. With no session timeout set, every user session that is begun is maintained until the application is terminated. This will cause your application to grow unabated, and result in unacceptable performance. Setting the session timeout by sending the message `this.setTimeOut(3600)` in the session's *init* method results in the session being deallocated (along with all of the memory it required) after 10 minutes (the argument for this method is in seconds). You need to choose the appropriate session time-out for your site.

## Recommended Reading for Session management

WebObjects Developers Guide: Chapter Titled "Managing State"

## Section II - Application testing and deployment

WebObjects provides several tools for testing and deployment of your web application, allowing you to evaluate your application's performance and monitor your production applications with relative ease. There are also a number of parameters that can be optimized when setting up your site to suit your production application's specific needs.

### Testing before deployment

Testing your WebObjects application before deployment is critical for many reasons. First, testing allows you to find and fix any undesirable behavior that your application may possess. Second, testing allows you to analyze the performance of your application, allowing you to further optimize its behavior using the techniques described in Section I of this guide. Analysis of the performance of your application also allows you to determine the number of instances required in your production environment, and hence the resources required by your production system when hosting the application.

Recording & PlaybackManager - When you install WebObjects 4.0 or later, a special Recording feature is included that, when enabled, allows you to watch, record and log a single user session of your WebObjects application. (In versions prior to WebObjects 4.0, WORecording and WOPlayback are provided as examples). You can later use that recorded session as a template for playback against your WebObjects application. Each playback process will loop through the recorded template continuously, and you can start many processes to evaluate the effect of an increased number of requests on your system.

You will probably go through several iterations of record and playback during testing. At first, you may want to record an average session to determine roughly whether your application performs under load as expected. Later, you may want to record a session that provides a maximum load on your application. For example, you may choose to record a session with many database fetches, a session that selects all of the longest-running database fetches possible in your application, or a session that accesses every page in your site.

The Recording feature can only capture a single user session, must not include any backtracking, and does not currently work with frames. To record a WebObjects 4.0 session, start the application on the command line or from Monitor and specify the -WORecordingPath option followed by the location of the recording file (the file must have a .rec extension).

```
MyApplication.exe -WORecordingPath c:\TEMP\MyApplication.rec
```

The PlaybackManager tool is installed in NEXT_ROOT/Library/WebObjects/Applications/PlayBackManager.woa where NEXT_ROOT is platform dependent (c:?Apple for NT, /opt/Apple on Solaris, etc). For more details on the Recording and Playback features of WebObjects 4.0, consult the Recommended Reading list at the end of this section.

WOStatisticsStore - WebObjects provides a statistics store object to record information about a WebObjects application instance in a log file, which can be accessed via the WOStats page or later analyzed by a Common Log File Format (CLFF) standard analysis tool. A partial listing of the information available on the WOStats page includes: number of transactions, average transaction time, average idle time, moving average transaction time, moving average idle time, average session life, number of session created, number of peak active sessions, average transactions per session, and current number of active sessions.

The WOStatisticsStore object is useful both during testing and deployment. Enabling the WOStatisticsStore logs in conjunction with the Recording and PlaybackManager tools mentioned above allows you to analyze the

performance of your WebObjects application under your simulated load conditions, and will help you to decide what optimization steps are required and how many instances of your application are necessary.

Statistics are not enabled by default. To store the information in a log file, you must set the log file path (your Application's *init* method is a good place to do this). For example,

```
this.statisticsStore().setLogFile("/tmp/WebObjects.log", 1);
```

would instruct the WOStatisticsStore object to place the output in a file named WebObjects.log located in the /tmp directory.

When a log file is set, the WOStatisticsStore object records everything returned by the *descriptionForResponse:inContext:* message, which is sent to the top-level response component after the *appendToResponse:inContext:* message. By default, this method returns the page name. Overriding this method allows you to include any additional information desired to the log.

### Recommended Reading for Testing before deployment

WOInfoCenter or Online Documentation: Section Titled "Serving WebObjects"
WebObjects 4.1 Deployment Book (not yet released)

### Monitor application and deployment parameters

WebObjects provides an administration application called Monitor that helps you create the WebObjects public configuration file, start and stop applications, and perform load balancing from a single machine, even if your applications are running on multiple machines. The WebObjects public configuration file contains information necessary for the WebObjects adaptor about your WebObjects applications - how many instances you have of each application, what machine and port number they are running on, etc. (WebObjects 4.0 and later versions install Monitor as a tool directly during installation, while versions prior to WebObjects 4.0 provided Monitor as an example.) The Monitor tool displays a subset of the WOStats information for each application instance for your convenience. When you configure a WebObjects application in the Monitor tool, there are a number of parameters you will specify that affect the performance of your site.

Application cycling - The first of these parameters is titled "Application cycling". An application's performance can be affected if it is allowed to run for long periods of time. As an application runs, the memory it requires will inevitably grow. This memory growth will consume more system resources than are really necessary, and will degrade the performance of your system over time. Therefore, periodically shutting down and restarting the application is recommended. The value you choose for the application cycling parameter determines when the periodic shutdown of your application will occur. Choosing a time during off-peak hours of access for your application is recommended. Alternatively, you may choose to set the minimum active sessions, enable inactivity self-kill, and auto-recover options to allow the application to terminate and restart gracefully after all user sessions are complete and the instance has been idle for some time. This provides a more graceful restart of the application than a scheduled shutdown, which will terminate the application no matter how many users are accessing it at that moment.

Adaptor key - The next parameter you can choose in Monitor that affects the performance of your site is titled "Adaptor key:". Using an NSAPI, ISAPI, or other API-based adaptor provides a performance advantage over the CGI adaptor because the web server can dynamically load the adaptor instead of spawning a new adaptor process for every request and killing that adaptor process after every response.

Instance number - The "Instance number" parameter lets you specify the number for this instance of a WebObjects application. Your application may have many simultaneously running processes, each on a different port (or even a different machine) with a different instance number. The WebObjects adaptor, in conjunction with the configuration file created by the Monitor application, determines how many instances of a given application are running, and dynamically load balances between those instances. You should use the Recording, PlaybackManager tools in conjunction with the WOStatisticsStore object to determine the number of instances required for your application. You do not want to start a large number of application processes unnecessarily, as you will consume more system resources than are really required.

Hostname - The "Hostname" parameter lets you specify the machine running this instance of your application. You do NOT need to have all instances of your application running on the same machine. It may be beneficial to distribute your application instances across a network of machines in order to handle the load for your system in a more optimal fashion.

### Recommended Reading for Monitor application and deployment parameters

WOInfoCenter or Online Documentation: Section Titled "Serving WebObjects"
WebObjects 4.1 Deployment Book (not yet released)

### Listen Queue Depth

Another parameter you can change to alter the behavior of your application is the listen queue depth. This parameter dictates how many requests can be queued up for a given instance of your application before that application starts refusing new requests. For example, if your application has a listen queue depth of five, and it receives one request every two seconds and is able to process one request every second, the listen queue is empty because your application is able to keep up with the load. If your application starts to receive two requests per second, the listen queue will fill up because the application can no longer keep up with the request loading. If your application experiences particularly heavy peak periods followed by a return to a lower request rate, increasing the listen queue depth may be effective in increasing your site's performance (this may be characterized by a sometimes-acceptable, sometimes-slow response rate). If, however, your site's request rate has increased so that the queue is never really empty (and response times are consistently slow), you should start additional instances of your application to better manage the site's load.

### Recommended Reading for Listen Queue Depth

WOInfoCenter or Online Documentation: Section Titled "Serving WebObjects"
WebObjects 4.1 Deployment Book (not yet released)

### System Configuration

Web server - Taking time to optimize your WebObjects application and deployment setup is ineffective if your web server is not also optimized. If requests are being bottlenecked because of a poorly configured web server, it makes no difference how many instances of an application you have running, or how big your server is. Your web master should have configured the web server for the appropriate number of instances (just like you set up the WebObjects application with multiple instances), and with the appropriate queue depth.

Physical configuration of your sever - Determine the size of a single application instance using Monitor or the WOStats page and multiply it by the number of instances you need to run. The result is the amount of physical memory recommended for that machine. Use of virtual memory to cover any limitations in the physical memory

size is acceptable, but is not as optimal. If you require a very large amount of memory for each instance of an application, you may want to review the amount of session information stored by the application.

Database server - If your WebObjects applications primarily access a database for information, hosting the database on a machine separate from the WebObjects applications and web server provides optimal results. If you need to run the database and WebObjects applications on a single machine, try to provide enough physical memory for the requirements of BOTH the database and the WebObjects applications.

## Recommended Reading for System Configuration

WOInfoCenter or Online Documentation: Section Titled "Serving WebObjects"
WebObjects 4.1 Deployment Book (not yet released)

## Conclusion

Correctly designing and implementing your WebObjects application results in a solid foundation for a responsive web site. Use of the techniques described in this guide will assist you in ensuring that your application is tuned and streamlined.

Phone: 817-335-7770
E-mail: theresa@tensor.com
URL: http://www.tensor.com

## Resources…

http://developer.apple.com/techpubs/webobjects
*WebObjects Developer's Guide*
*Enterprise Objects Framework Developer's Guide*
http://www.omnigroup.com/MailArchive/WebObjects
http://www.omnigroup.com/MailArchive/eof
http://www2.stepwise.com/cgi-bin/WebObjects/Stepwise/Sites
ftp://dev.apple.com/devworld/Interactive_Media_Resources
http://www.apple.com/developer
http://developer.apple.com/media
http://til.info.apple.com/

## About the Author…

Theresa Ray is a Senior Software Consultant for Tensor Information Systems in Fort Worth, TX (*http://www.tensor.com)*. She has programmed in OPENSTEP (both WebObjects and AppKit interfaces) on projects for a wide variety of clients including the U.S. Navy, the United States Postal Service, America Online, and Lockheed-Martin. Her experience spans all versions of WebObjects from 1.0 to 4.0, EOF 1.1 to 3.0, NEXTSTEP 3.1 to OPENSTEP 4.2, Rhapsody for Power Macintosh, and yellow-box for NT. In addition, she is an Apple-certified instructor for WebObjects courses.

Tensor Information Systems is an Apple partner providing systems integration and enterprise solutions to its customers. Tensor's employees are experienced in all Apple technologies including OPENSTEP, NEXTSTEP, Rhapsody, EOF and WebObjects. Tensor also provides Apple-certified training in WebObjects, Oracle consulting and training, as well as systems integration consulting on HP-UX. And Oracle.

You may reach Theresa by e-mail: *theresa@tensor.com* or by phone at (817) 335-7770.

**Interactive Media Resources Include:**

- **Interactive Media Guidebooks**
- **Market Research Reports**
- **Survival Guides—Technical "How To" Guides**
- **Comarketing Opportunities**
- **Special Discounts**

# Apple Developer Connection

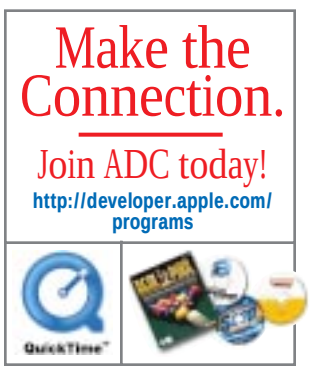

As Apple technologies such as QuickTime, ColorSync, and AppleScript continue to expand Macintosh as the tool of choice for content creators and interactive media authors, the Apple Developer Connection continues its commitment to provide creative professionals with the latest technical and marketing information and tools.

### Interactive Media Resources

Whether looking for technical guides from industry experts or for market and industry research reports to help make critical business decisions, you'll find them on the Interactive Media Resources page.

### Apple Developer Connection Programs and Products

ADC programs and products offer easy access to technical and business resources for anyone interested in developing for Apple platforms worldwide. Apple offers three levels of program participation serving developer needs .




### Membership Programs

*Online Program*—Developers gain access to the latest technical documentation for Apple technologies as well as business resources and information through the Apple Developer Connection web site.

*Select Program*—Offers developers the convenience of technical and business information and resources on monthly CDs, provides access to prerelease software, and bundles two technical support incidents.

*Premier Program*—Meets the needs of developers who desire the most complete suite of products and services from Apple, including eight technical support incidents and discounts on Apple hardware.

### Standalone Products

Apple offers many standalone products that allow developers to choose their own level of support from Apple or enhance their Select or Premier Program membership. Choose from the following products and begin enjoying the benefits today.

*Developer Connection Mailing*—Subscribe to the Apple Developer Connection Mailing for the latest in development tools, system software, and more.

*Technical Support*—Purchase technical support and work directly with Apple's Worldwide Developer Technical Support engineers.

*Apple Developer Connection News*—Stay connected to Apple and developer-specific news by subscribing to our free weekly e-mail newsletter, *Apple Developer Connection News*. Each newsletter contains up-to-date information on topics such as Mac OS, Interactive Media, Hardware, Apple News and Comarketing Opportunities.

### Macintosh Products Guide

The most complete guide for Macintosh products! Be sure to list your hardware and software products in our *free* online database!

## http://developer.apple.com/media

The ultimate source for creative professionals.