WebObjects Overview

Mac OS X Server > WebObjects

2007-07-11

Contents

Figures

Introduction to WebObjects Overview

Important: The following tools are deprecated and no longer supported in WebObjects 5.4 and later: EOModeler, RuleEditor, WebObjects Builder, WOALauncher, and Java Client. WebObjects templates are not available for creating new projects in Xcode on Mac OS X v10.5 and later.

WebObjects is an application server with tools, technologies, and capabilities to create Internet and intranet applications. It has an object-oriented architecture that promotes quick development of reusable web components. WebObjects is extremely scalable and supports high transaction volumes.

WebObjects solves some common problems—dynamic webpage generation, user forms and input, state management, and interface with databases—that usually consume most of a your development time. This allows you to focus on presentation, your data model and business logic.

This document introduces the architecture, technologies, development tools, and development approaches. After reading this document, you will be able to make an informed decision about which approach to use.

Who Should Read This Document

WebObjects Overview is written for developers who want to start using WebObjects and need to choose an approach. However, anyone interested in WebObjects technology will benefit from reading this document.

This document does not assume you have a background in object-oriented programming. However, WebObjects is based on object-oriented frameworks written in Java. You should be familiar with object-oriented programming if you intend to write WebObjects applications.

An advantage of WebObjects is the database connectivity and rapid prototyping tools it provides. To fully appreciate WebObjects, you should have some understanding of databases, although this document doesn’t require it.

Because WebObjects provides several distinct approaches to developing applications, this document discusses them one by one and compares their pros and cons to help you decide which approach is appropriate for your application.

Organization of This Document

This document has the following chapters:

- "About Web Objects" (page 9) introduces the technologies of WebObjects and how they fit together. The technologies include APIs and tools for creating HTML-based, desktop, and web services applications. WebObjects also supports some J2EE integration.

- "Choosing Your Approach" (page 19) summarizes the pros and cons of each approach and suggests a development process to evaluate which approach or combination of approaches is appropriate for your application.

There is also a glossary, "Glossary" (page 25), which contains definitions of common object-oriented programming, web, and database terms used in this document.

See Also

Read *Getting Started with WebObjects* after reading this document for links to specific documents corresponding to your learning path.

All WebObjects documentation is located in `/Developer/ADC Reference Library`—double-click `index.html` and navigate to the Internet & Web category to view your local copies of WebObjects documentation. Otherwise, view the latest WebObjects documentation online at http://developer.apple.com/documentation.

WebObjects developer tools are located in `/Developer/Applications/WebObjects` and sample programs are located in `/Developer/Examples/JavaWebObjects`.

About Web Objects

From an information-technology perspective, WebObjects is a scalable, high-availability, high-performance application server. From the viewpoint of a developer, though, WebObjects is an extensible object-oriented platform upon which you can rapidly develop and deploy applications that integrate existing data and systems. WebObjects allows you to build applications that leverage the connectivity that the Internet or an intranet provides using a multitiered client-server architecture.

The web was created to simplify access to electronically published documents. Originally webpages were just static text pages with hyperlinks to other documents. However, they quickly evolved into highly graphical animated presentations. Along the way, a degree of interactivity was introduced, allowing people browsing the web to fill out forms and thereby supply data to the server.

WebObjects allows you to take the next logical step. With it, you can produce full-fledged applications for use either across the Internet or within a corporate intranet. Users not only fill out forms but can author content stored in back-end databases. By tracking user sessions and preferences, you can offer a custom user experience much like a desktop application.

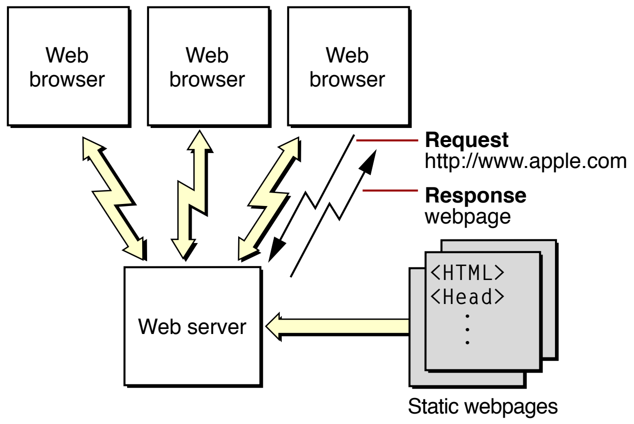
These applications can be web-based, and thus accessible through a web browser, or can have the full interactivity of a stand alone desktop application. Your application can also provide web services to other web applications.

Dynamic Web Publishing

Much of the content on the web is textual or graphical material that doesn't change much over time. However, there is growing demand for websites that publish ever-changing content—for example, breaking news stories, up-to-the-minute stock quotes, and the current weather.

The architecture of a static website is shown in Figure 1-1. A user's web browser requests pages using Uniform Resource Locators (URLs). These requests are sent over the network to the web server, which analyzes each request and selects the appropriate webpage to return to the user's browser. This webpage is simply a text file that contains HTML code. Using the HTML tags embedded within the file received from the HTTP server, the browser renders the page. Because HTML in its simplest form is just a markup language, there is no interactivity with a static page.

Figure 1-1 A static website



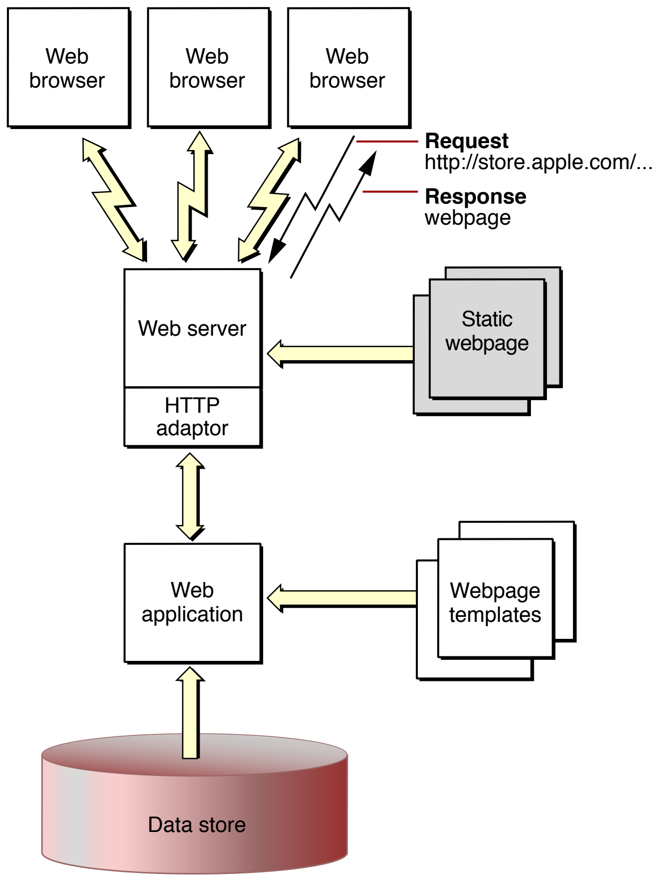
Static websites are easy to maintain. There are a number of tools in the market that allow you to create webpages with a relatively small amount of effort. And, as long as the content of your pages doesn't change too often, it isn't difficult to keep them up-to-date.

However, if the content on your pages needs to be "live"—for example, updated continually from multiple sources—or interactive—for example, depends on user preferences or queries—then it would be too labor intensive to maintain all the possible static webpages.

WebObjects allows you to quickly and easily publish dynamic content over the web. You create webpage templates that indicate where on the webpage the dynamic content is placed. WebObjects fills in the content when the page needs to be generated in response to a request. The information your applications publish can reside in a database or other data-storage medium or it can be generated at the time a page is accessed. The pages are also highly interactive—you can fully specify the way the user navigates through them and what data they can view and modify.

Figure 1-2 shows a WebObjects-based website. Again, the request (in the form of a URL) originates from a web browser. The web server detects that the request should be handled by a WebObjects application, and passes the request to an HTTP adaptor. The adaptor packages the incoming request in a form the WebObjects application can understand and forwards it to the application. Based upon webpage templates you define and the relevant data from the data store, the application generates a webpage that it passes back through the adaptor to the web server. The web server sends the page to the web browser, which renders it.

Figure 1-2 A dynamic publishing website



This type of WebObjects application is referred to as **web-based**, since the result is a series of dynamically generated HTML webpages.

Instead of using an HTTP adaptor, you can deploy applications through servlet containers. This approach allows you to take advantage of your servlet container's application deployment facilities. For more information on this approach, read *WebObjects J2EE Programming Guide*.

Different Client-Server Applications

Although the majority of websites publish static content, the number of sites that publish dynamic content is growing rapidly. Many enterprises use intranets, the Internet, or both to provide easy access to dynamic content. Online stores selling books, music, or computers are examples of an Internet client-server application.

Client-server applications offer huge advantages over traditional applications. Users don't have to install the application on a client computer, which not only saves client disk space but ensures that the user always has the most up-to-date version of the application. Also, the client computers can be Macs, PCs, or anything that can run the client, a web browser, with the necessary capabilities.

WebObjects allows you to develop three different types of Internet applications: web applications, Java Client applications, and web services. Web applications are analogous to Common Gateway Interface (CGI) applications and consist of dynamically generated webpages accessed through a web browser. Java Client moves part of your application to the client-side computer and enlists Sun's Java Foundation Classes (JFC) to give it the complete user interface found in a more traditional desktop application. Web services uses the same back-end components of web applications and Java Client applications to provide services to other web applications. Rapid prototyping tools are available for each of these types of client-server applications.

Web Applications

You can create a web application quickly and easily with WebObjects. WebObjects provides many HTML-based elements that you can use to build your web application's interface. These elements range from simple user interface widgets (for example, submit buttons, checkboxes, and tables) to elements that provide for the conditional or iterative display of content.

You can also define web components. These are webpage templates that you can use to define your web application's design. Web components can contain any of the layout elements mentioned earlier as well as other web components. For example, you can create a toolbar component that provides a link to your main website and to a search webpage. Then, as you create other components, you include the toolbar component in them. When you develop a support page for your website that you want all the other components to use, the toolbar component is the only place you need to add a link to it.

Web components encapsulate more than the layout of a webpage. They also encompass a Java file that links the component's elements and subcomponents into a coherent entity. You put application-specific business logic in the Java class of a web component.

For more information on web applications, read *WebObjects Web Applications Programming Guide*. Read *WebObjects Builder User Guide* for how to construct reusable web components.

Desktop Applications

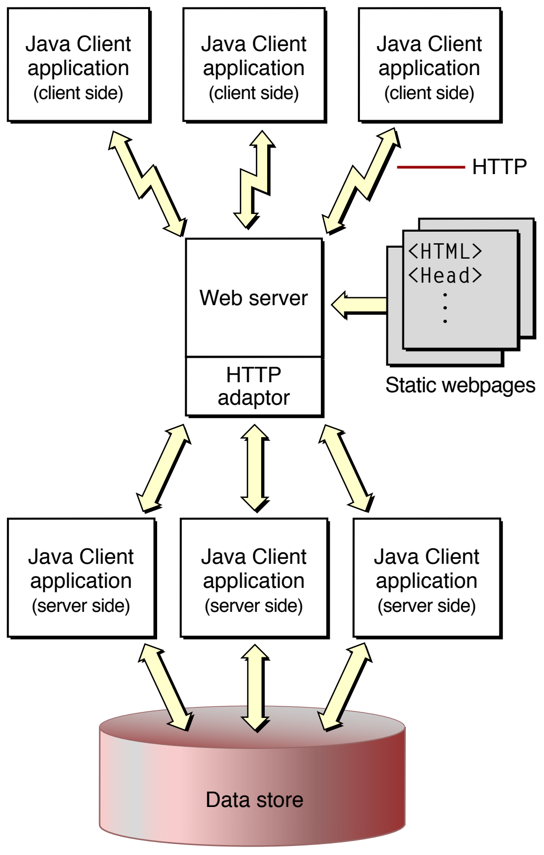
When you need the fast and full-featured user interface of desktop client-server applications, you can partition your application so that a portion of it—including all or part of the user interface logic—runs in Java directly on the client. Client-server communication is handled by WebObjects. WebObjects applications that are partitioned in this way are known as Java Client applications.

Java Client distributes the objects of your WebObjects application between the application server and one or more clients—typically Java applications. It is based on a distributed multitier client-server architecture where processing duties are divided between a client, an application server, a database server, and a web server. With a Java Client application, you can partition business objects containing business logic and data into a *client side* and a *server side*. This partitioning can improve performance and at the same time help to secure legacy data and business rules.

Figure 1-3 illustrates a Java Client application in which the client portion is running as an application installed on the user's computer. Java Client applications, just like web applications, can communicate with the application server using HTTP requests. In addition, Java Client passes objects between the portion of your application residing on the user's computer and the portion of your application that remains on the application server.

For more information on desktop applications, read *WebObjects Java Client Programming Guide*.

Figure 1-3 Java Client applications in action


Java Client application (client side)
Java Client application (client side)
Java Client application (client side)
HTTP
Web server
HTTP adaptor
<HTML>
<Head>
Static webpages
Java Client application (server side)
Java Client application (server side)
Java Client application (server side)
Data store

Web Services

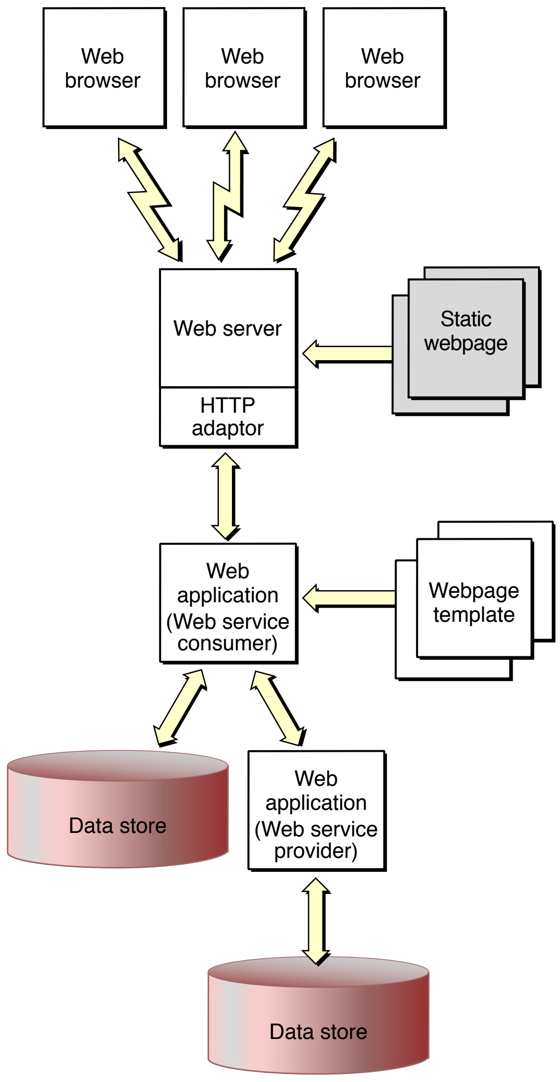
Web services is an innovative implementation of distributed computing. WebObjects allows you to expose class methods as web service operations. Web services provide an efficient way for applications to communicate with each other. Based on Simple Object Access Protocol (SOAP) messages that wrap Extensible Markup Language (XML) documents, web services provide a flexible infrastructure that leverages the ubiquitous HTTP (or HTTPS) over TCP/IP. This means that your organization probably has all the hardware and software infrastructure needed to deploy web services.

But web services provide more than an information-exchange system. When an application implements some of its functionality using web services, it becomes more than the sum of its parts. For example, you can create a web service operation that uses a web service operation from another provider to give its consumers (also known as *service requestors*) information tailored to their needs. Web service operations are similar to the methods of a Java class; a provider is an entity that publishes a web service, while the entities that use the web service are called consumers.

Web applications as well as Java Client applications can take advantage of web services. Figure 1-4 shows a dynamic-publishing website that uses web services.

For more information on web services applications, read *WebObjects Web Services Programming Guide*.

Figure 1-4 A dynamic publishing website using web services



Rapid Prototyping

WebObjects provides power and flexibility. A certain degree of complexity, however, accompanies these features. For many applications, whether web-based or Java Client–based, it's more important initially to develop the application quickly than strive for maximum customization. As an example, a simple data-browsing and editing application, intended only for internal use by a system administrator, probably wouldn't warrant the same degree of effort you would put into an application accessible by the general public. To simplify the development of applications like the former, WebObjects includes a set of rapid-prototyping technologies: Direct to Web, Direct to Java Client, and Direct to Web Services.

These three technologies correspond to the three types of applications described in "Different Client-Server Applications" (page 11) but are similar in approach. Each technology creates a different type of application: a web application by Direct to Web, a Java Client application by Direct to Java Client, and a web services application by Direct to Web Services. These technologies use a data model as the base upon which an application is created. In addition, they are useful not only for creating simple data-browsing applications or web services, but for creating early prototypes. Because they allow customization on various levels, they are also well-suited for creating shipping applications.

Direct to Web

Direct to Web is a system for creating web applications that access a database. All Direct to Web needs to create the application is a model of the database, which you can build using EOModeler or the Xcode EO model design tool.

Direct to Web uses information from a data model to dynamically generate webpages. Consequently, you can modify your application's configuration at runtime—using the Web Assistant—to hide objects of a particular class, hide their properties, reorder properties, and change the way they are displayed without recompiling or relaunching the application.

Out of the box, Direct to Web generates webpages for nine common database tasks, including querying, editing, and listing. To do this, Direct to Web uses a task-specific component called a template that can perform the task on any entity. The templates, in conjunction with a set of rules (which you can customize), are the essential elements of a Direct to Web application.

A Direct to Web application is highly customizable. For example, you can change the appearance of the standard templates, mix web components with webpages generated by Direct to Web, and create custom web components and Direct to Web templates that implement specialized behavior.

You can also freeze pages and edit them using WebObjects Builder. By doing so, you improve performance but lose the ability to change the page using the Web Assistant. Usually, you freeze pages just before deploying your Direct to Web application.

For more information on Direct to Web, read *WebObjects Direct to Web Guide*.

Direct to Java Client

Like Direct to Web, Direct to Java Client generates a user interface for common database tasks using rules to control program flow and provides an assistant that allows you to modify your applications at runtime. The applications produced by Direct to Java Client have rich desktop-class user interfaces. In addition, Java Client applications can take advantage of the processing power of the client computer to perform operations such as sorting or filtering a list of items received from the server.

For more information on Direct to Java Client, read *WebObjects Java Client Programming Guide*.

Direct to Web Services

Direct to Web Services allows you to create a web service that lets its clients access data in your data store by invoking web service operations. Although this approach is similar to Direct to Web and Direct to Java Client in its use of a data model and rule sets, the target users for web service applications are other applications, not people.

You use the Web Services Assistant to determine which data entities are accessible by your web service clients and the type of operations they can execute on them, such as search, insert, delete, and update. You accomplish this without writing a single line of code.

For more information on Direct to Web Services, read *WebObjects Web Services Programming Guide*.

Enterprise Objects

Dynamic WebObjects applications are more powerful if you can populate the generated HTML, Java Client messages, or web service responses with data obtained from a back-end database. A back-end database can provide read and write access to a shared repository of web content, including data used by your web application such as customer records.

WebObjects uses Enterprise Objects to encapsulate the details of accessing tables and records in a database. Your application uses enterprise objects, created from Java classes, to read and write data. You specify the mapping between these enterprise objects and the database using an entity-relationship model. You create the entity-relationship model using either EOModeler or the Xcode EO model design tool. Enterprise Objects seamlessly takes care of all your queries, read/write access, and faulting.

Your enterprise objects are true models in the Model-View-Controller paradigm that encapsulate data and business logic. Enterprise object classes represent the part of your application that won't change regardless of which application-development approach you choose.

Enterprise Objects enables the Direct to Web, Direct to Java Client, and Direct to Web Services technologies. These technologies use the entity-relationship model to abstract rules and templates for controlling what data can be viewed and modified.

For more information on Enterprise Objects, read *WebObjects Enterprise Objects Programming Guide*. For details on how to use EOModeler, read *EOModeler User Guide*. Read *Xcode 2.2 User Guide* for how to create a model in Xcode.

The WebObjects Advantage

WebObjects provides a number of key technologies that give it a significant advantage over other application servers.

Streamlined Database Access

Much of the data that is (or could be) presented on the web already exists in electronic form. Not only can it be a challenge to create a website or web application to present your data using conventional tools, accessing the data itself could be difficult. Some products rely on manually created or assistant-generated Structured Query Language (SQL) code, leading to database-specific code that is difficult to optimize. WebObjects avoids these problems by using Enterprise Objects, a model-based mechanism for cleanly instantiating business objects directly from database tables. Enterprise Objects handles all the interactions with the database including fetching, caching, and saving. This allows you to write your business logic against actual objects independent of the underlying data store. You can modify schemas, add or change databases, or even use a totally different storage mechanism without needing to rewrite your application.

Separation of Model, View, and Controller

An ideal web application–development system that is also object-oriented simplifies maintenance and encourages code reuse by enforcing a clean separation of models (data store), views (webpages), and controllers (Java code). This separation is inherent in the WebObjects programming, which uses reusable web components to generate webpages directly from enterprise-object instances without the need to embed scripts or Java code inside the webpages themselves. A web component contains a webpage template, which you—or a professional webpage designer—can design and edit using standard webpage authoring tools. A component can also implement custom behavior using a separate Java source file. Neither the template nor the Java source file includes data-model–specific information.

State Management

The HTTP protocol used on the web is inherently stateless; that is, each HTTP request arrives independently of earlier requests, and it is up to web applications to determine which of the active users sent it. Therefore, most web applications of consequence—as well as some of the more interesting dynamic publishing sites—need to keep state information, such as login information or a shopping cart, associated with each user session.

Without using cookies, WebObjects provides objects that allow you to maintain information for the life of a particular user session, or longer. This makes it particularly easy to implement an application like an online store: you don't have to do anything special to maintain the contents of the user's shopping cart or other data over the life of the session. In addition, your online store could even monitor customer buying patterns and then highlight items a particular buyer is likely to be interested in the next time she visits your site.

Modular Development

The power of WebObjects comes from a tightly integrated set of tools and frameworks, facilitating the rapid assembly of complex applications. At the heart of this system is Xcode, an integrated development environment (IDE) that manages your Java business logic and tracks data models, web components, and supporting files. As mentioned earlier, WebObjects also includes powerful assistants and frameworks that allow the rapid creation of web, Java Client, and web services applications directly from the database. Advanced developers can tap into the WebObjects API, allowing virtually unlimited customization and expandability.

Pure Java

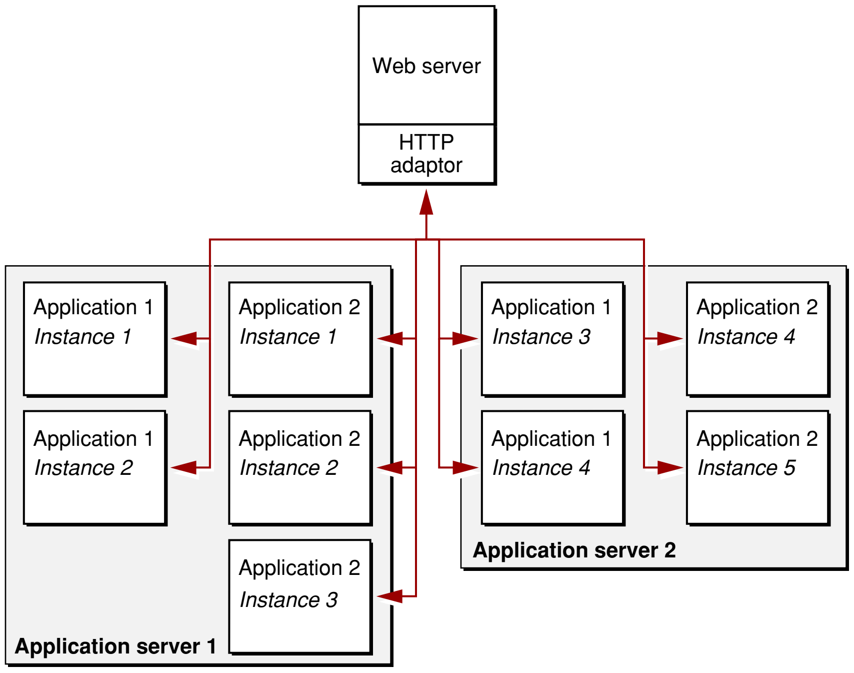
WebObjects applications are 100% Pure Java, which means they can be deployed on any platform with a certified Java virtual machine.

Scalability and Performance

Static websites and traditional client-server applications have one advantage: They both leverage the power of the client platform, minimizing the load on the server. It doesn't take all that much processing power to serve a set of static webpages. Dynamic applications, although a tremendous advance over static websites, require additional server power to quickly access the changing data and construct the webpages or Java Client user interface.

The WebObjects application server is both efficient and scalable. With WebObjects, if more power, reliability, or failover protection is needed, you can run multiple instances of your application, either on one or on multiple application servers (see Figure 1-5). You can choose from one of several load-balancing algorithms (or create your own) to determine which application instance each new user should connect to. And, either locally or from a remote location, you can analyze site loads and usage patterns and then start or stop additional application instances as necessary. Load balancing is a very powerful feature of WebObjects that allows you to add more server capacity as the need arises without needing to implement a load-balancing algorithm yourself. Read *WebObjects Deployment Guide Using JavaMonitor* for more deployment options.

Figure 1-5 Multiple instances of two applications



Choosing Your Approach

Before you get started using WebObjects, you need to decide which development approach to use. There are many choices available but it's not difficult to choose given your requirements. It's not difficult to change your approach either because your enterprise objects are the same. You should consider the following issues when making a decision:

- Are you planning to deploy over the Internet or an intranet?
- What are your user interface requirements?
- How quickly do you need to develop the application?

This section briefly describes the different approaches—the advantages, disadvantages, and typical use of each approach—including a discussion of rapid prototyping and combining approaches.

Enterprise Objects

First decide if you are using Enterprise Objects. You use Enterprise Objects if you need to access data from a back-end database and use that data to drive the content generated by your application. All types of WebObjects applications including HTML-based, Java Client, and web services can use Enterprise Objects to model their data. Enterprise Objects is extremely scalable supporting large data stores and clients—it is used by many successful online stores. The only reason not to use Enterprise Objects is if you don't have persistent data or you have a proprietary data store that is incompatible with Enterprise Objects. In the later case, you might still be able to use Enterprise Objects by writing your own adaptor.

Web Applications

Creating web applications that are HTML-based is the most popular choice because any user on any computer with a web browser can access your website. Typically, web applications don't require the user to download software. You can create login pages and track user sessions in a web application but you don't have to use this feature—your users can access pages directly. You can combine dynamic with static pages to improve performance, too. You can use JavaScript, Flash, and Quicktime to improve the user experience. Because of this, web applications is a common choice for public websites/Internet applications with a large and broad user base—such as community websites and online stores.

Java Client

In contrast, Java Client applications are more suitable for intranet applications. Java Client is a popular choice for enterprises that have some control over the client computers. Choose this option when downloading client-side software is not a problem.

The user must wait for client-side software to download and the quality of the user's Java virtual machine determines whether the application runs correctly and efficiently. Java Web Start can help in making Internet deployment of Java Client applications more user friendly.

Java Client can provide a better user interface similar to other desktop applications. The user interface response time may be better in a Java Client application also because some computations can take place on the client side.

Web Services

Web services are used when you don't have a user interface and just want to provide a service to other web applications. For example, a medical system might have many different components used for billing, patient records, and scheduling. Each of these components might have its own data storage and web interface. However, data may need to be shared across different components while at the same time controlling access to sensitive data—controlling read/write permissions. In this case, web services can be used to control access from one application to another and perform common tasks, such as fetching demographic information about a patient. Typically, web applications can extend their usefulness by providing web services to other web applications including Dashboard widgets on Mac OS X.

You can provide and consume web services on the Internet or an intranet. However, due to the emerging nature of web service technology, you should take into account security issues before making web services available on the Internet.

Choosing web services doesn't exclude other approaches. Fortunately, using Direct to Web Services you can create a web services application easily from an existing web or Java Client application. All you need is your business logic framework and EO model.

Rapid Prototyping

You can use the rapid prototyping tools—Direct to Web, Direct to Java Client, or Direct to Web Services—to create an application faster and with less effort. All you really need to create a rapid prototype is an EO model and optionally, your business logic.

Choosing one of the rapid prototyping tools doesn't exclude any of the other approaches either. Your business logic framework and EO model can be the same for all types of WebObjects applications. However, using a rapid prototyping tool for the production system has some advantages and disadvantages.

In all cases, the rapid prototyping tools are great for testing your model and business logic. Often these prototypes help define system requirements and identify design problems. Later, they can be used to debug an application. You can use it to inspect and edit your data too. Because of this, prototypes make ideal data entry and administration tools. Rapid prototyping may be your only choice if your EO model is evolving and it is impossible to maintain hundreds of custom pages. For these reasons, prototypes are often used throughout the lifetime of a project.

The user interface generated by a rapid prototyping tool is not, however, flexible or fancy enough for high-profile websites. The webpages of a Direct to Web application are generated from templates—for example, a query, inspect, edit, list and master-detail template per entity. The layout and flow of the website is predetermined—you just customize each of the entity pages using the Web Assistant. You can freeze pages and use WebObjects Builder to modify the look, but you need a combined approach to change the control flow.

Your decision whether or not to use Direct to Java Client is similar. If your Java Client user interface has specific layout and flow requirements, then don't use Direct to Java Client. Keep in mind that the Direct to Java Client approach—including the user interface it generates—is designed expressly for viewing and editing databases, especially large ones. If your application requires this capability, you will find the Direct to Java Client user interface well-suited for the task.

If a rapid prototyping tool doesn't meet your needs, consider combining approaches as described in "Combining Approaches" (page 21).

Combining Approaches

WebObjects does not confine you to a single approach. You can switch your approach as you develop your application or combine it with another approach. This is possible in WebObjects because the business logic is encapsulated in enterprise objects and not in the application.

For example, the web application and Direct to Web approaches can be combined in many ways. You can start with a Direct to Web application and freeze templates and pages to create a custom look. You can also replace the main pages with your own login, tunnel, and navigation pages. You create all these custom pages using WebObjects Builder.

You can also use Direct to Web reusable components, located on the palette in WebObjects Builder, in any web application. The components—corresponding to inspect, edit, and list pages—can be customized using the Web Assistant. If your application employs forms and lists that work with enterprise objects, these components can save you a tremendous amount of time.

You can also mix Java Client and Direct to Java Client applications. If you're developing a Java Client application and you need a Direct to Java Client controller—for example, a window that edits an enterprise object—you can easily instantiate one. Also, you can freeze an interface in Direct to Java Client and edit it with Interface Builder.

It's not uncommon to implement several approaches in parallel until you determine which is the best for your application. This is a typical development path:

1. Create your EO model
2. Create a Direct to Web application
3. Create a Direct to Java Client application (as a comparison)
4. Choose an approach and create your custom application
5. Create a Direct to Web Services application

Using Direct to Web Services is just another way to extend the usefulness and lifetime of your application. Usually after several releases of your application, you discover some valuable services that you can provide to other applications. If you use Enterprise Objects, you can easily create and deploy web services using Direct to Web Services.

Document Revision History

This table describes the changes to *WebObjects Overview*.

Date	Notes
2007-07-11	Updated for WebObjects 5.4.
2006-01-10	Completely revised. This book now focuses on choosing an approach.
2005-04-29	Fixed a link to the SSL Specification.
2005-02-03	Fixed various errors and typos.
2003-05-14	Corrected programming-interface usage example in "Publishing the Calculator Class as a Web Service."
2002-09-01	Revised to reflect changes made in WebObjects 5.2.
	Added chapter about Web services support, including Direct to Web Services.
	Changed references to *HTML-based application* to *Web application*.
	Combined contents of “HTML-Based Applications” and “Direct to Web” chapters in one chapter, "Web Applications."
2002-01-01	Revised to reflect changes made in WebObjects 5.1.
	Combined contents of “Java Client Applications” and “Direct to Java Client Applications” chapters in one chapter, "Desktop Applications.".
	Added chapter on J2EE support.
2000-12-01	First version of *Inside WebObjects: WebObjects Overview*.

Glossary

business logic The rules associated with the data in a database that typically encode business policies. An example is automatically adding late fees for overdue items.

CGI (Common Gateway Interface) A standard for interfacing external applications with information servers, such as HTTP or web servers.

class In object-oriented languages such as Java, a prototype for a particular kind of object. A class definition declares instance variables and defines methods for all members of the class. Objects that have the same types of instance variables and have access to the same methods belong to the same class.

column In a relational database, the dimension of a table that holds values for a particular attribute. For example, a table that contains employee records might have a LAST_NAME column that contains the values for each employee's last name.

database server A data storage and retrieval system. Database servers typically run on a dedicated computer and are accessed by client applications over a network.

Direct to Java Client A WebObjects development approach that can generate a Java Client application from a model.

Direct to Java Client Assistant A tool used to customize a Direct to Java Client application.

Direct to Web A WebObjects development approach that can generate a web application from a model.

Direct to Web Services A WebObjects development approach that can generate a web service application from a model.

Direct to Web template A component used in Direct to Web applications that can generate a webpage for a particular task (for example, a list page) for any entity.

dynamic element A dynamic version of an HTML element. WebObjects includes a list of dynamic elements with which you can build web components.

enterprise object An object that conforms to the key-value coding protocol and whose properties can map to stored data. An enterprise object brings together stored data with methods for operating on that data.

Enterprise Objects Enterprise Objects is a set of frameworks to build feature-rich database applications that encapsulate your business logic, yet are independent of any particular data source.

entity In Entity-Relationship modeling, a distinguishable object about which data is kept. For example, you can have an Employee entity with attributes such as `lastName`, `firstName`, `address`, and so on. An entity typically corresponds to a table in a relational database; an entity's attributes, in turn, correspond to a table's columns.

Entity-Relationship modeling A discipline for examining and representing the components and interrelationships in a database system. Also known as ER modeling, this discipline factors a database system into entities, attributes, and relationships.

EOModeler A tool used to create and edit models.

faulting A mechanism used by WebObjects to increase performance whereby destination objects of relationships are not fetched until they are explicitly accessed.

fetch In Enterprise Objects applications, to retrieve data from the database server into the client application, usually into enterprise objects.

HTTP adaptor A process (or a part of one) that connects WebObjects applications to a web server.

instance In object-oriented languages such as Java, an object that belongs to (is a member of) a particular class. Instances are created at runtime according to the specification in the class definition.

Interface Builder A tool used to create and edit graphical user interfaces like those used in Java Client applications.

Java Client A WebObjects development approach that allows you to create graphical user interface applications that run on the user's computer and communicate with a WebObjects server.

JFC (Java Foundation Classes) A set of classes that implement graphical user interface components, also called Swing components.

JDBC An interface between Java platforms and databases.

key An arbitrary value (usually a string) used to locate a datum in a data structure such as a dictionary.

method In object-oriented programming, a procedure that can be executed by an object.

model An object (of the EOModel class) that defines, in Entity-Relationship terms, the mapping between enterprise object classes and the database schema. This definition is typically stored in a file created with the EOModeler application. A model also includes the information needed to connect to a particular database server.

Model-View-Controller An object-oriented programming paradigm in which the functions of an application are separated into the special knowledge (model objects), user interface elements (view objects), and the interface that connects them (the controller object).

object A programming unit that groups together a data structure (instance variables) and the operations (methods) that can use or affect that data. Objects are the principal building blocks of object-oriented programs.

record The set of values that describes a single instance of an entity; in a relational database, a record is equivalent to a row.

relational database A database designed according to the relational model, which uses the discipline of Entity-Relationship modeling and the data design standards called normal forms.

relationship A link between two entities that's based on attributes of the entities. For example, the Department and Employee entities can have a relationship based on the `deptID` attribute as a foreign key in Employee, and as the primary key in Department. This relationship would make it possible to find the employees for a given department.

reusable component A component that can be nested within other components and acts like a dynamic element.

request A message conforming to the Hypertext Transfer Protocol (HTTP) sent from the user's web browser to a web server that asks for a resource like a webpage.

response A message conforming to the Hypertext Transfer Protocol (HTTP) sent from the web server to the user's web browser that contains the resource specified by the corresponding request. The response is typically a webpage.

row In a relational database, the dimension of a table that groups attributes into records.

session A period during which access to a WebObjects application and its resources is granted to a particular client (typically a browser). Also an object (of the WOSession class) representing a session.

table A two-dimensional set of values corresponding to an entity. The columns of a table represent characteristics of the entity and the rows represent instances of the entity.

to-many relationship A relationship in which each source record has zero to many corresponding destination records. For example, a department has many employees.

to-one relationship A relationship in which each source record has exactly one corresponding destination record. For example, each employee has one job title.

transaction A set of actions that is treated as a single operation.

Web Assistant Tool used to customize a Direct to Web application.

Web component An object (of the WOComponent class) that represents a webpage or a reusable portion of one.

Webpage template HTML file that specifies the overall appearance of a webpage generated from a web component.

Web Services Assistant Application used to customize a Direct to Web Services applications.

WebObjects Builder An application used to edit web components.

Index