

# CIRCUIT CELLAR®

THE MAGAZINE FOR COMPUTER APPLICATIONS

## FEATURE ARTICLE

Mark Taylor

# Embedded Systems *n*-Formation

## Implementing an *n*-Tiered Client-Server Architecture

In corporate environments, who knows what depends on who knows who, and who knows who depends on who who is. As Mark shows us, the *n*-tiered architecture was designed to make sure the right people have access to the information they need.



As the structure and relationships of information in the corporate environment become more complex, it is becoming increasingly more important to design systems that support access to the ever-growing diversity of resources. People now have significant computing power available to them on their desktops.

Technological advances have enabled professionals of all disciplines to use systems developed for their particular needs, from standard PCs running Windows to high-end graphics workstations running Unix. These powerful systems, combinations of sophisticated hardware and software, emphasize the importance of information in today's corporate environment. In such an environment, the goal is to provide individuals with easy access to the necessary set of resources and information.

With the evolution of the worldwide web, corporations are taking advantage of web technology to provide distributed access to enterprise data. Corporate networks connect a wide range of re-

sources, including desktop notebooks, PCs, and workstations, as well as database and corporate-application systems.

Web browsers on desktop systems provide access to information on internal corporate networks and the Internet. This flexibility gives individuals significant resources to accomplish a wide variety of tasks.

Web-based architecture has evolved out of the generic client-server architecture. This architecture provides a model for network applications that is flexible, powerful, and takes advantage of Internet technology.

The model is the basis for the significant increase in the number of web applications, providing access to massive online databases through online magazines, newspapers, catalogs, and other information-related applications. The structure for this architecture is referred to as *n*-tiered, which is loosely defined as the separation of the user interface, the application, and the database components.

Embedded systems are becoming part of the critical path for web business. Therefore, the corporate network topography must include these resources. To accomplish this, the solution must use the existing, available technology-of-choice and apply the *n*-tiered architecture to an embedded-system environment. The task presents significant challenges, but the solution provides easy integration of embedded systems into the corporate network environment.

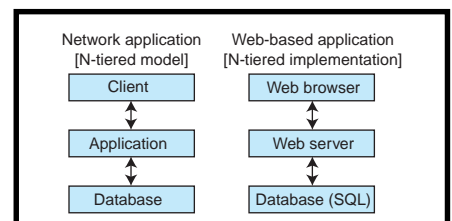


Figure 1—The *n*-tiered architecture defines the functional separation of user interface and data in general terms, dividing a network-based application into client, application, and database tiers.

## n-TIERED ARCHITECTURE

The software applications available for common corporate-environment systems are sophisticated and powerful, complementing the potential inherent in evolving Internet/Intranet technologies.

The technology of distributed computing has evolved to the point that end-user systems now support a standardized user-interface that provides remote access to data from a wide variety of resources. Web browsers, standardized on the hypertext markup language (HTML) and the hypertext transport protocol (HTTP), provide a graphical user interface for locating and displaying data distributed across the network.

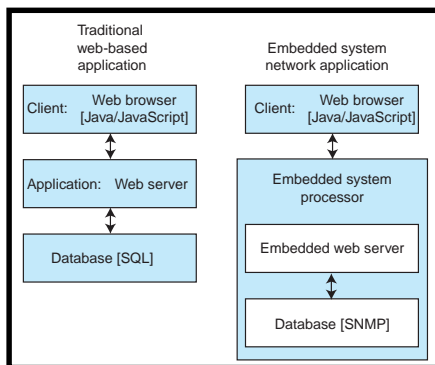
As shown in Figure 1, the client tier encapsulates user interface functions. The application tier supports data processing associated with client requests, and provides the connection between the client tier and the database tier. And last, the database tier provides data storage.

## DATABASE

The *n*-tiered model of client, application, and database tiers is implemented as a web-based application that includes web browser (client tier), web server (application tier), and database system. Evolving from the client-server technology, the *n*-tiered architecture takes advantage of the web to standardize the client system user interface. It's not limited to the foundation of the three-tiered model but can be expanded to include other tiers as appropriate to the particular application. Each tier can be implemented to run on its own system.

A web browser provides an easy and familiar means of navigating and viewing remote data. Web browsers support the HTML standard, resulting in a consistent look and feel and providing a target environment that lends itself to low-overhead development processes. Web-browser architecture also provides a platform for Java and JavaScript applications customized for client/user interface support.

In the *n*-tiered model, web servers are part of the application tier. A web server provides network communication with client browsers acting as the agent for application data. The application



**Figure 2**—For example, the most efficient way to provide remote web-based access to an Ethernet switch, is to run the web server on the Ethernet switch system, manage the interface between the web server and the SNMP MIB agent for the Ethernet switch database, and create a web page interface that effectively presents the Ethernet switch data.

tier generally executes on its own system, providing the link between the web browser and the database.

Database systems make up the third tier of the general *n*-tiered architecture. The database tier is often implemented using an SQL interface to provide access to large databases, which generally occupy their own systems.

## THE EMBEDDED ENVIRONMENT

*n*-tiered architecture provides an opportunity for embedded systems to participate in the corporate network. If the *n*-tiered model can be successfully applied to embedded systems, individuals at remote locations can use web browsers to view data from network devices and to manage the network devices.

The conventional application of *n*-tiered architecture uses significant resources for large application servers and large database systems. However, embedded systems impose strict limitations on resources.

The application and database tiers are often implemented as components that run on the same processor. The web server must be capable of being ported to an embedded-system environment without having a negative impact on the embedded system.

Embedded-systems architecture requires that ancillary tasks (e.g., a web server) execute in nonpreemptive, asynchronous mode to allow the primary, mission-critical tasks (data acquisition and device manipulation) to execute to completion.

In the embedded environment, the database tier often takes the form of a management protocol such as simple network management protocol (SNMP). The implementation of the application tier must support the interface to the device data (see Figure 2).

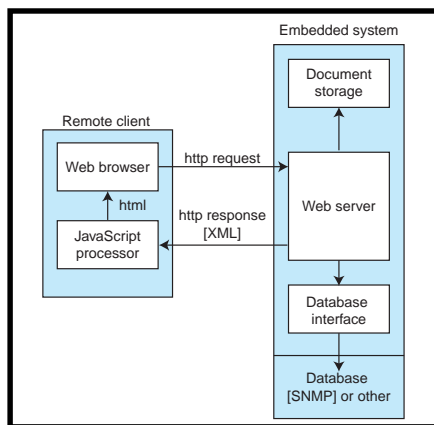
## RESOURCE OPTIMIZATION

Resource optimization is the crucial consideration in applying *n*-tiered architecture to embedded systems. The priorities in embedded-system environments require components that optimize the use of both memory and time. Applying *n*-tiered architecture to embedded systems requires a flexible system that optimizes the components in the client and application tiers.

Two elements have significant impact on accomplishing the goal of resource optimization. First, do as much processing as possible before run time, creating a run-time environment that is as efficient as possible.

Compiler technology provides significant advantages and opportunities when it's applied to web content. By compiling web content, it's possible to perform optimization on the content. A compiled web content database eliminates the need for the server to interpret the HTML stream, which provides a higher level of performance. Compiler technology also enables you to compress web content, resulting in space optimization.

Second, at run time, use client-side processing where possible to support instant feedback for end users and to reduce the load on the server. If the



**Figure 3**—The embedded web server system provides optimization for the client and for the server with components that can withstand the rigor of the embedded-system environment.



**Photo 1**—The tab semantics provide navigation between the Configure web content and the Statistics web content. The form fields in the Addresses section display the device data.

server can offload user interface tasks to the client system, then the embedded processor load is reduced. If the client can generate feedback where the data is entered, the number of messages between the client and the server are reduced, resulting in a more user-friendly interface.

## ***n*-TIERED IMPLEMENTATION**

An implementation of an *n*-tiered architecture for embedded systems supplies components that contribute to the client and application tiers of the *n*-tiered model. In the embedded-systems environment, the application and database tiers are combined on the embedded system, creating a need for a solution that has a minimal impact on the run-time environment (see Figure 3).

Because the server runs on the device, resources are limited. To meet the requirements of the embedded-system environment, the embedded web server should be optimized to provide resource savings for time and space. Using compiler technology makes it possible to achieve this sort of optimization.

There are three ways that compiler technology affects optimization. First, there is no interpreting or reading of HTML content at run time. The compiler can separate the dynamic content from the static content, which lets the web server copy the static information into network buffers with no interpretation required. For HTML forms, this process can provide a considerable savings because all form data can be indexed ahead of time, eliminating the need for any interpreting or look-

ups.

Second, there is no server-side interpreting of dynamic content. All server-side dynamic content such as scripting with Javascript or query optimizations for SQL can be compiled. Compiling ahead of time eliminates the need for any parsing or interpreting at runtime and provides the opportunity to perform other optimizations.

And lastly, run-time can be optimized through compression. Using compression on web content is important for saving space in an embedded environment. Compression algorithms can be tuned for run-time memory and CPU utilization.

A compiler provides an opportunity to balance the different resource requirements so that the compression/decompression process is decompression light and compression heavy. For example, it's possible to do multipass dictionary-based compression on the resource-rich host, which permits efficient run-time decompression without searching.

## **CLIENT-SIDE OPTIMIZATION**

Additional resource savings are realized when the embedded web-server system can provide high-level support for web-page graphics. HTML requires a significant number of statements to encode complex graphics to manage web content. By using an XML-based language to define user interface components for use in web pages, the embedded web server system provides an easy method of defining complex graphical components and interactions.

For example, in Figure 4, the <TABBAR> and <TABITEM> XML

tags define a set of tab pages, analogous to a set of file cards with protruding tabs for labels. With these tags, it's possible to condense several pages of related web content into one visual page, using the tabs to navigate between the subdivided web content.

Figure 4 uses both XML and HTML. The XML tags define the tab page context for the web-content display and provide the layout look and feel for the display. The HTML defines the form that is used to display the device data. The <GROUPING> XML tags in the form declaration displays a border around the set of input fields.

The run-time support for the web-page display in Photo 1 is implemented as a JavaScript component that executes on the web-browser system. The JavaScript processes the XML web-page content, producing HTML that the browser interprets for display. Using the <TABBAR> XML, the JavaScript component generates the substantial amount of HTML required to construct the interface. By implementing a JavaScript processor, the server load is reduced, minimizing the impact on the embedded-system processor.

## **NOW YOU KNOW**

Using *n*-tiered architecture in the embedded-system environment requires more than a cursory application of the existing web-based technology. A successful solution will provide easy integration of embedded systems into the corporate network environment.

To do this effectively, the embedded web server system provides:

- A set of optimized components that have minimal impact on the embedded system
- A web server that supports the latest standards for HTTP and HTML
- A method of easily generating user interfaces for the web-based application

To accomplish these goals, the embedded

```

<TABBAR>
  <TABITEM LABEL='Configure'>
    <CONTENT>
      <FORM EMWEB_NAMESPACE='configdb'>
        <GROUPING LABEL='Addresses'>
          <INPUT NAME='ipaddress' LABEL="IP Address:"/>
          <INPUT NAME='subnetmask' LABEL="Subnet Mask:"/>
          <INPUT NAME='broadcast' LABEL="Broadcast:"/>
        </GROUPING>
      </FORM>
    </CONTENT>
  </TABITEM>

  <TABITEM LABEL='Statistics'>
    <CONTENT>
      ...
    
```

**Figure 4**—Using tabs enables you to include and access plenty of information on a single viewable page.

web server system has to be designed and developed specifically for the embedded-system environment.

**Circuit Cellar, the Magazine for Computer Applications.** Reprinted by permission. For subscription information, call (860) 875-2199 or [subscribe@circuitcellar.com](mailto:subscribe@circuitcellar.com).