Recommendations for Web Development and Deployment Using Team Developer

Unify Corporation

Table of Contents

Abstract

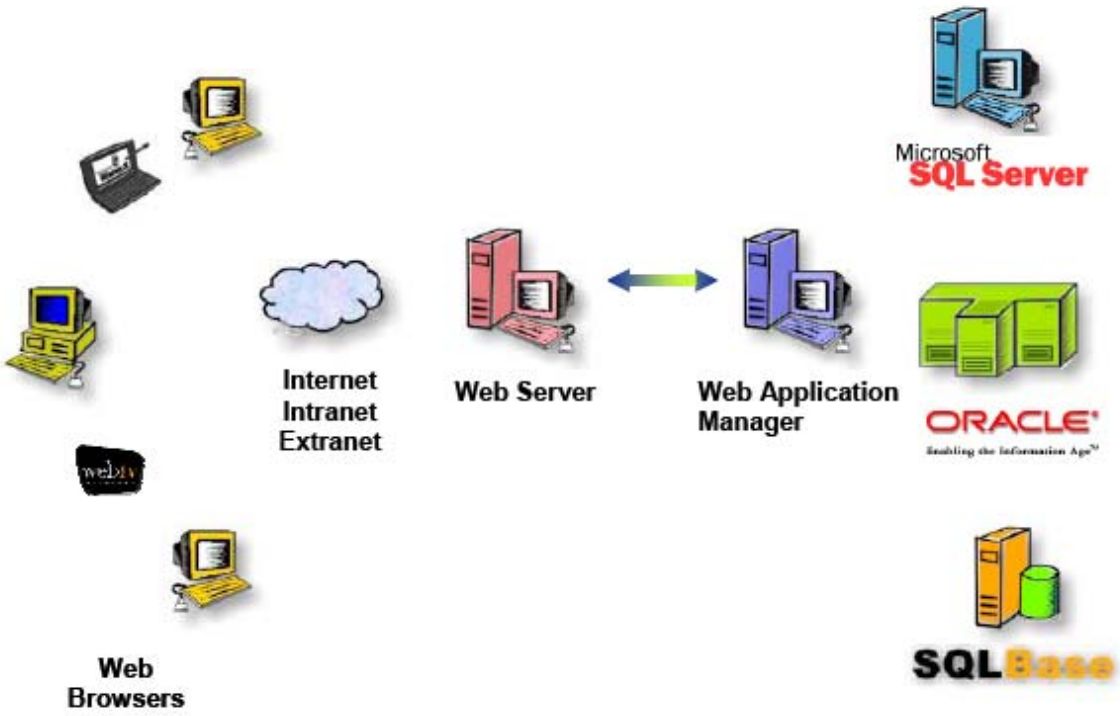
Team Developer is a development environment that allows you to easily build and deploy enterprise-scale client/server or Web applications. There are two ways to develop Web applications with Team Developer.

- **Using our Web Extensions and Web App Manager** for small-scale Intranet web applications.
- **Using ASP and COM** for enterprise scale applications.

This paper mainly focuses on web development and deployment using Web Extensions (GWE) and Web App Manager (WAM). Team Developer allows programmers to easily build and deploy Web applications

Application Flow in Web Mode

Before diving deep into development and deployment recommendations, it is very important to understand the architecture or the application flow in web mode.
The diagram below describes the flow of the application from thin clients to server and from servers to thin clients via net.



Development

Mode of Application

Dedicated Mode

Advantages of dedicated applications are:

- Development is easy, as the programmer doesn't have to deal with state maintenance.
- User authentication is much easier as it only needs to be done once.

Team Developer's Web Extensions offers two modes of Web application development. In dedicated mode, after fulfilling a request from a browser, the application service continues to run until the programmer-defined exit point or the timeout is reached. In this mode, an instance of an application when invoked by a browser is assigned specifically to the instance of the browser. Dedication is bi-directional: all submissions from the browser are directed at the specific application instance, and the application instance only receives submissions from the associated browser. Web App Manager manages this dedication automatically.

Disadvantages of dedicated applications are:

- It is associated with an inbound connection, which limits the number of users that can be connected to an application at a time.
- For every user, Web Application Manager will have to load a process and this can cause the machine running Web Application Manager to run out of resources.
- Even on a very powerful machine, many users hitting it can easily reach the window handle limitation imposed by Windows operating system.
- Fault recovery is poor to nonexistent. If the server that is handling your session has a problem you cannot proceed.
- Because of lack of state management end users cannot use browser features - for example, Back and Forward buttons on the browser.

Re-useable Mode

Advantages of re-useable applications are:

- Reduced load on the application server. Each instance of a re-useable application can support a number of physical users.
- Users can take advantage of the flexibility of the browsers. For example, users are able to use back button of the browser (assuming correct state management is in place).
- The server doesn't have to keep the session alive.
- You can use web-like interface features such as interacting text links or frames. This is not possible in dedicated mode.
- Provides improved scalability.

With re-useable mode, once the application has finished servicing a request from a browser and HTML has been rendered to the requesting browser, the application instance is effectively idle until the next request is received. Also, the application instance is open to requests from browsers other than the current one. This makes the application re-useable.

Disadvantage of re-useable applications is:

- Statelessness of re-useable applications makes development more involved, as the developer has to deal with state management.

Our Recommendation

Considering the advantages and disadvantages discussed above, we recommend the re-useable mode for Web development. A typical Web application is re-useable. Although it makes development difficult, this gives more flexibility to end-users and creates fewer loads on the application server. As the application is re-useable, unlike the dedicated mode application, it doesn't easily hit the window handle limitation of Windows operating systems.

OLE DB Connectivity is strongly recommended for web applications

Database Connectivity

Team Developer provides different ways to connect to databases:

- OLEDB
- Native routers
- ODBC

OLEDB

OLEDB is developed based on the ActiveX Data Objects (ADO) interface. OLEDB provides consistent, high performance access to data, specifically in a distributed model such as the Web deployment of Team Developer. Moreover, using OLEDB, one can access relational database and also access non-relational data such as an XML data source.

Native routers

Team Developer is popular for its native routers, which are still available for backward compatibility.

ODBC

Team Developer has an ODBC router that enables developers to talk with any ODBC data source.

Our Recommendation

For Web application development using Team Developer, we strongly recommend the use of OLEDB connectivity for data access.

Maximum number of Users
We recommend a maximum of 75 users per Web Application Manager (WAM). There should be a maximum of five processes per application with a maximum of 15 users per process. When there are more than 75 users, you should consider having more application servers, each running a copy of WAM, to handle user requests. Depending on the maximum number of users expected, you should decide the number of application servers (WAMs) to be deployed.

To achieve higher levels of throughput, one could partition the application into logical business processes. This means that there will be multiple executable files all triggered at appropriate locations from the Browser front-end.

Development Time –vs- Scalability
As mentioned above, the main drawback of the dedicated type of applications is scalability (each user will have to have a dedicated process running on the server) and the main drawback of a re-useable application is that it requires more development time. When migrating from client server to Web, making them dedicated Web applications is much easier and quicker than making them reusable applications. With dedicated applications, individuals don't have to worry about maintaining state and this makes application development quicker. This is mainly because it keeps track of its context like traditional client server applications.

Our Recommendation
In a typical Web environment, the number of concurrent users is high and we recommend re-useable applications in such cases. However, if the application is a very small single form application (e.g. survey applications) and you don't expect many concurrent hits, then a dedicated application may be adequate.

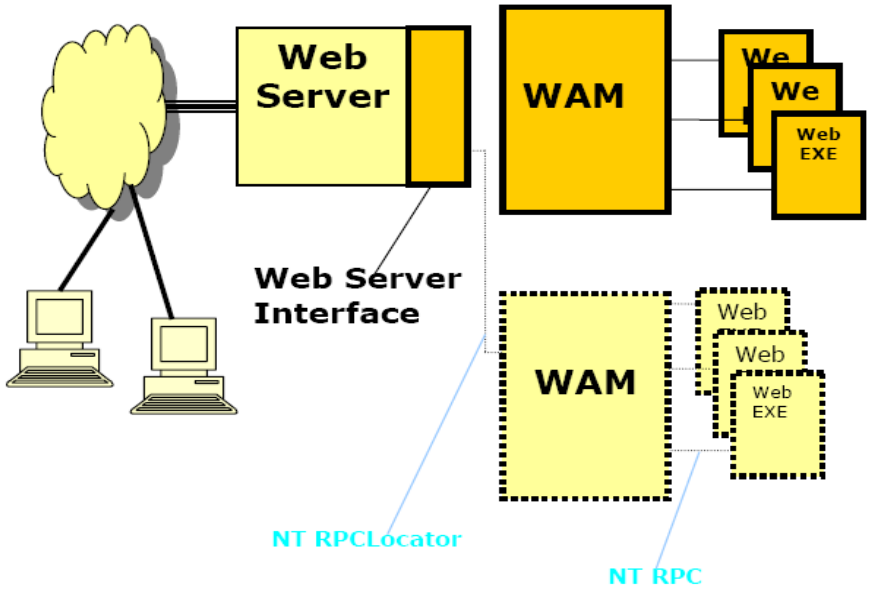
Migration from Client-Server to Web
In the client-server environment the clients are rich clients. Typical client-server environment is two-tier; applications reside on the client and databases reside on the server. Client-server applications are persistent and they keep track of their context. Due to this nature, jumping from form to form or going back to the caller form can be easily accomplished. In addition, creating windows-like interfaces (for example: menus, toolbars, drag and drop features, etc.) can be easily accomplished.

In Web mode, the clients are thin clients with an Internet browser. Thin clients are server dependant. Typical Web environment is more distributed with three/more-tier. Thin clients are expected to have only an Internet browser, the application server will host the app manager and applications, and a database server will host database. To achieve maximum benefits from the distributed model and new technologies like COM/MTS, one would convert their business logic into COM servers and run on a separate server, typically on a MTS server. Due to the non-persistent nature of the Web applications, some client-server centric features such as traversing back and forth between forms, drag and drop, toolbars, status bars and menu interfaces are not so easy to implement although it is not impossible.

Our Recommendation
It is very important to spend time and effort planning the migration. Important considerations are separating business logic from Graphical User Interface (GUI) and designing user interface to replace client-server like user interface with web like user interface.

Deployment

Environment Setup
The diagram below shows a typical deployment environment with Web Application Manager(s).



Technical details regarding the application flow in a WAM deployed environment are discussed below.

Process flow

- User will enter http://webserver/scripts/cwisapi.dll?Service=Service in the browser.
- Request will go the Web server and Web server will look for the scripts directory in its Web server directory.
- It will then look for a file called cwisapi.dll. This DLL has the ability to transfer the request to the Web App Manager via another layer, cwifixx.dll.
- Once Web App Manager receives the request, it will look for the application service name specified in the URL and do the necessary processing, including accessing the database, to create the HTML.
- Once the request is processed, WAM creates the HTML string, which is rendered to the requesting browser via the Web server.

Note: cwisapi.dll is specific to IIS. We support Netscape and other PC based Web servers via different interface DLLs.

This setup can be achieved in four different ways

- A thin client with browser and a server machine that holds the Web server, WAM and database server.
- A thin client with browser, a server with Web server and WAM and another server with database server.
- A thin client with browser and three servers - one for Web server, one for WAM and one for Database server.
- UNIFY supports IIS, Netscape and any CGI-based Web server
- Many WAM servers can be configured. If redundant services are configured on multiple WAM servers and the applications are stateless, excellent fault tolerance is achieved. This can also improve scalability. Shifting processing to the middle tier (WEB app) can ease database load (you might cache information, etc.)

Our Recommendation

We recommend having separate servers for Web server, WAM and database. This way the load will be distributed across three different servers and better performance can be achieved. For improved fault tolerance and scalability, we recommend multiple WAM servers and these WAM servers should be part of the same domain.

Supported Web Servers

Unify supports IIS, Netscape or any other PC based CGI compatible Web servers. Three different Web server interface files are used to handle communication between different Web servers and WAM.

- IIS - Web server interface used on IIS to communicate with WAM is cwisapi.dll.
- Netscape - On Netscape Web server cwnsapi.dll must be used for communication.
- Any other PC based CGI compatible Web servers (for example, Apache must use cwcgi.exe)

Scalability

Considering the above recommended setup for deployment, when there are more users than a single WAM could handle, you can deploy more WAMs so the requests will be load balanced across WAMs. When there is more than one WAM installed, each WAM will recognize the existence of other WAMs. However, a request will be transferred to a different WAM only when the first one hit is not able to handle the request. So the requests are not always evenly distributed. To overcome this issue, you can write a simple application accessing internal functions of WAM to redirect requests evenly and this is discussed in the next section, Load Balancing.

Load Balancing

Right now Web App Manager will transfer a request to another WAM when the original one that receives the request can't handle it. This happens because it has hit some resource limit (applicable only for stateless applications). It doesn't actually check for the number of users accessing WAM at a time to spread user requests

evenly across WAMs. However, this can be initiated programmatically. A simple application accessing internal functions of the Web Extension will be able to enable real load balance for a particular application.

A very important function to know is from the external DLL cwadi21.dll (please note that the DLL name changes according to the version of Team Developer) and the function name is CenAdmConnect. The function declaration for CenAdmConnect is as follows.

Function: CenAdmConnect
Description: APSUSERAPI DWORD APSAPI CenAdmConnect (LPCSTR pszComputerName, handle_t * phServerManager,
LPDWORD pdwRunningMode);
Export Ordinal: 0
Load Balancing may be achieved programmatically
Returns
Number: DWORD
Parameters
String: LPCSTR
Receive Number: LPHANDLE
Receive Number: LPDWORD

Using this function, user requests can be redirected to any server. The other important function that can be used to decide when and to which server the request should be forwarded is CenAdmQueryService. The function declaration for CenAdmQueryService is as follows.

Function: CenAdmQueryService
Description: APSUSERAPI DWORD APSAPI CenAdmQueryService (LPCSTR pszComputerName, handle_t hServerManager, LPCSTR pszServiceName,
PtrPtrCenServiceDesc ppServiceDesc);
Export Ordinal: 0
Returns
Number: DWORD
Parameters
Number: HANDLE
Number: HANDLE
String: LPCSTR
Receive Number: LPHANDLE

Using the above function, one can find out the current active processes and maximum allowed processes for a server. Based on these findings, requests can be redirected to a server when required.

A raw application with just cWebMgrRaw form should be enough for program redirection. Function WebMgrCreate can be coded as follows. A list of WAM

servers, service names, interface dlls and Max Service can be defined under this function.

Function: WebMgrCreate
Actions
Set sServer[0]= ‘SERVER1’
Set sServer[1]= ‘SERVER2’
Set sServer[2]= ‘SERVER3’
Set sInterface[0]= ‘/scripts/cwisapi.dll’
Set sInterface[1]= ‘/scripts/cwisapi.dll’
Set sInterface[2]= ‘/scripts/cwisapi.dll’
Set sService[0]= ‘ex1’
Set sService[1]= ‘ex2’
Set sService[2]= ‘ex3’
Set nMaxServices = 3
Set nActualService = 0

In order to query information regarding a particular application service, a local function called QueryServiceInfo is coded using external call CenAdmQueryService() and is as follows:
Function: QueryServiceInfo
Parameters
String: p_sServer
String: p_sAppName
Receive Number: p_nCurrProc
Receive Number: p_nMaxProc
Static Variables
Local variables
Number: nError
CServiceDescriptor: oServiceDescriptor
Actions
Set nError = WAMQueryService(hServer, p_sAppName, oServiceDescriptor)
Set p_nMaxProc = oServiceDescriptor.PropGetMaxProcesses( )
Set p_nCurrProc = oServiceDescriptor.PropGetCurProcesses( )

…where CServiceDescriptor is a functional class containing many functions related to applications service. Functions PropGetMaxProcesses() andPropGetCurProcesses() will be used to obtain the Max Processes defined for a particular application service and the number of processes currently being used, respectively.

Then under ProcessRequest function, you need to work out the logic of when the request should be transferred and to which server. The sample ProcessRequest function is as follows:

Function: ProcessRequest

```
Actions
Set bFirst = TRUE
Set nOldRedirectTo = -1
Loop
Set nRedirectTo = SalNumberMod( nActualService, nMaxServices )
If nOldRedirectTo = nRedirectTo
Set bRedirect = FALSE
Break
If bFirst
Set nOldRedirectTo = nRedirectTo
Set bFirst = FALSE
Set nActualService = nActualService + 1
Call ConnectWAM( sServer[nRedirectTo] )
Call QueryServiceInfo( sServer[nRedirectTo], sService[nRedirectTo], nCurProc,
nMaxProc )
Call DisconnectWAM( )
If nCurProc < nMaxProc
Set bRedirect = TRUE
Break
If bRedirect
Set sHTML = '<HTML>
<meta http-equiv="refresh" content="0; URL=http://'|| sServer[nRedirectTo] ||
sInterface[nRedirectTo] || '?Service=' || sService[nRedirectTo] || '">
<HEAD>
<TITLE>BALANCEIT! 2.0</TITLE>
</HEAD>
<BODY>
<a>to be redirected...</a>
</BODY>
</HTML>'
Else
Set sHTML = '<HTML>
<HEAD>
<TITLE>BALANCEIT! 2.0</TITLE>
</HEAD>
<BODY>
<a>Sorry, no free services available, try later or contact your administrator!</a>
</BODY>
</HTML>'
Call PutBytes( sHTML, SalStrLength( sHTML ), 'text/html', TRUE )
Call ReuseApp( )
```

The raw Web application described above will be able to handle load balancing very well. Another external function from cwadi21.dll that may be useful in the process is CenAdmDisconnect.

Function: CenAdmDisconnect
Description: APSUSERAPI DWORD APSAPI CenAdmDisconnect (handle_t *
phServerManager);
Export Ordinal: 0
Returns
Number: DWORD
Parameters
Receive Number: LPHANDLE

About Unify

Unify is a global provider of software development technology and solutions that helps IT customers participate in Service-Oriented Architecture (SOA). Unify's productive and re-liable development tools, migration solutions and databases enable organizations to build and modernize business essential applications for SOA. Composer for Lotus Notes offers a complete, like for like, production to production migration solution for Lotus Notes applications. Unify's award-winning NXJ Developer enables IT teams to be extremely productive, learn new technologies fast and deliver Web services-based applications on time and on budget. The Team Developer, SQLBase, DataServer, ACCELL and VISION product families enable cross-platform rapid development on Java/J2EE, Linux or Windows. Unify has a rich heritage in delivering rich, cost-effective technologies to its thousands of IT customers and ISV, VAR and distributor partners. Unify is headquartered in Sacramento, Calif., and can be reached at (916) 928-6400 or by visiting www.unify.com.

Unify Corporation
2101 Arena Blvd., Suite 100
Sacramento, CA 95834
USA
Phone: 1.916.928.6400
Toll Free: 1.800.468.6439
Fax: 1.916.928.6404
Munich: +49 8 115 55430
United Kingdom: +44 (0)1753 245 510
France: +33 (0)1 34 58 28 30