Содержание:

Введение

Использование серверов приложений — следующий шаг за клиент-серверной технологией, позволяющий повысить доступность и надежность информационной системы предприятия. Но мне кажется, что самым важным преимуществом сервера приложений является его большая гибкость, что очень ценно для крупных предприятий. Способность предприятия максимально быстро реагировать на изменение состояния рынка и окружающей среды позволяет ему оставаться конкурентоспособным.

Назначение

Для сервера приложений характерны расширенные возможности обработки информации, а взаимодействие с клиентом становится подобным работе приложения. В маркетинге термином «сервер приложений» обычно обозначают предлагаемое продавцами комплексное решение, которое содержит все требуемые компоненты технологий. Для некоторых организаций такой комплексный подход к построению сервера приложений облегчает разработку благодаря унификации разрабатываемых моделей и централизации поддержки.

«Беспроводной» сервер»:

В своей простейшей интерпретации такой компьютер может представлять собой типичный Web-сервер или сервер приложений, который просто знает, как передавать документы, составленные на стандартном для беспроводных устройств языке. Часто в качестве такого языка выступает Wireless Markup Language (WML). Адаптация Web-сервера для работы в качестве беспроводного сервера, способного обрабатывать документы WML-типа, обычно сводится просто к тому, чтобы обучить сервер распознаванию этих документов. Web-серверу требуется только сообщить клиенту, что документ составлен в формате для беспроводных устройств, и на этом

его работа заканчивается.

Технические характеристики

Серверы приложений относят к приложениям промежуточного слоя (middleware). Существует несколько категорий продуктов промежуточного слоя:

1. Message orientated (яркие представители — MQseries и JMS);
2. Object Broker (пример — CORBA и DCOM);
3. Component based — наиболее перспективная категория, на мой взгляд, и это подтверждают известные представители данной категории .NET и EJB.

В настоящее время есть несколько серверов приложений таких крупных компаний, как Sun Microsystems, Borland, IBM, Oracle, и каждый из них отличается набором предоставляемых сервисов (производительность в данном случае учитывать не будем). Эти сервисы облегчают программирование и развертывание приложений масштаба предприятия. Вы можете использовать уже готовые строительные блоки для реализации необходимой бизнес-логики.

Основные решаемые задачи

Инсталляция и администрирование пользовательских библиотек

Процесс инсталляции заключается в размещении пользовательских библиотек в соответствующие каталоги. Возможны два вида пользовательских библиотек: общие и частные. Общие библиотеки пользователей должны находиться в каталоге %AS_DIR%\lib. Там находится библиотека стандартных функций lib.dll. Свои частные библиотеки каждый пользователь должен помещать в специально отведённый для них каталог %AS_DIR%\lib\<имя_пользователя>.

Каждый пользователь имеет доступ только к своим частным библиотекам. Общие библиотеки доступны всем пользователям.

При подключении библиотеки с помощью оператора OPEN LIBRARY <имя_библиотеки> интерпретатор в поисках данной библиотеки просматривает сначала каталог, содержащий частные библиотеки пользователя. Если библиотека с таким именем не найдена, то интерпретатор просматривает каталог, в котором хранятся общие библиотеки пользователей. В случае совпадения имён частной и

общей библиотек, интерпретатор подключает частную библиотеку пользователя.

При вызове функции интерпретатор ищет её в подключенной библиотеке. Если функция не найдена, то интерпретатор ищет её в библиотеке стандартных функций, которая подключается автоматически.

Обмен данными между интерпретатором ASPL и внешними хранимыми процедурами

Обмен данными между интерпретатором ASPL и внешними хранимыми процедурами осуществляется через внутренний стек интерпретатора. Перед вызовом внешней процедуры интерпретатор помещает в стек входные значения процедуры и передает ей управление. Внешняя процедура извлекает входные значения из стека, обрабатывает их, помещает в стек выходные значения и передает управление интерпретатору. Интерпретатор извлекает из стека выходные значения и обрабатывает их.

Создание пользовательских библиотек

Пользовательские библиотеки создаются по общим правилам построения динамически подключаемых библиотек (см. приложение). Наряду с этим есть некоторые особенности, относящиеся к передаче входных и выходных значений библиотечных функций. Любая библиотечная функция должна в явном виде извлекать входные значения из стека и помещать в стек выходные значения. Это объясняется тем, что интерпретатор заранее не может знать о количестве передаваемых параметров. Поэтому перед передачей управления функции интерпретатор помещает в стек входные значения, не проверяя их количество, а функция, получив управление, извлекает входные значения из стека и проверяет их число. После вычисления выходных значений функция должна поместить их в стек. Для того чтобы отличать библиотечные функции от стандартных функций языка C, все функции, входящие в пользовательские библиотеки, должны иметь префикс as_ и быть объявлены следующим образом:

void as_<имя_функции>(void)

Для извлечения входных значений из стека служит функция Pop(). Она возвращает значение типа VALUE, так как в стеке содержатся значения только этого типа. Для того чтобы привести переменную типа VALUE к требуемому типу, нужно воспользоваться одной из функций вида GetAs<Type>(VALUE *), где Type - требуемый тип. Проверка стека на наличие в нём значений осуществляется с

помощью функции IsStackEmpty().

После вычисления выходных значений, они должны быть приведены к типу VALUE с помощью одной из функций вида Make<Type>(VALUE*, <Type>). Затем выходные значения заносятся в стек с помощью функции Push(VALUE*) или Return (VALUE*). Эти функции идентичны.

В качестве примера рассмотрим функцию нахождения синуса as_sin, содержащуюся в библиотеке стандартных функций lib.dll.

```
void as_sin ( void ) /*Объявление функции*/

{

VALUE v;

double f;

if (IsStackEmpty() ) /*Проверка числа параметров*/

AS_ERROR ("Illegal use of sin function.");

v = Pop(); /*Извлечение входного значения из стека*/

f = GetAsDouble (&v); /*Приведение входного значения к типу Double*/

errno = 0;

f = sin(f);

if ( errno != 0 )

AS_ERROR("SIN error."); /*Сообщение об ошибке и выход из функции*/

Drop(&v);

MakeDouble (&v, f); /*Приведение выходного значения к типу VALUE*/

Return ( &v ); /*Занесение выходного значения в стек*/

}
```

Использование пользовательских библиотек

Для использования функций из пользовательской библиотеки соответствующая библиотека должна быть подключена с помощью оператора OPEN LIBRARY <имя_библиотеки>. После этого функции, входящие в библиотеку, могут быть вызваны оператором CALL:

CALL <имя_функции> (<список_входных_значений>) RETURNING <список_возвращаемых_значений>.

Если функция возвращает только одно значение, то её можно использовать в выражениях, например, f = sin(x).

Замечание. При вызове функций, входящих в пользовательские библиотеки, к имени функции не нужно добавлять префикс as_, так как интерпретатор делает это автоматически. Например, рассмотренную выше функцию as_sin следует вызывать CALL sin(x) RETURNING x.

Особенности работы и конфигурирования

Преимущества серверов приложений:

1. Целостность данных и кода:

Выделяя бизнес логику на отдельный сервер, или на небольшое количество серверов, можно гарантировать обновления и улучшения приложений для всех пользователей. Отсутствует риск, что старая версия приложения получит доступ к данным или сможет их изменить старым несовместимым образом.

1. Централизованная настройка и управление:

Изменения в настройках приложения, таких как изменение сервера базы данных или системных настроек, могут производиться централизованно.

1. Безопасность:

Сервер приложений действует как центральная точка, используя которую, поставщики сервисов могут управлять доступом к данным и частям самих приложений, что считается преимуществом защиты. Её наличие позволяет переместить ответственность за аутентификацию с потенциально небезопасного уровня клиента на уровень сервера приложений, при этом дополнительно скрывая уровень базы данных.

1. Поддержка транзакций:

Транзакция представляет собой единицу активности, во время которой большое число изменений ресурсов (в одном или различных источниках) может быть выполнено атомарно (как неделимая единица работы). Конечные пользователи при этом могут выиграть от стандартизованного поведения системы, от уменьшения времени на разработку и от снижения стоимости. В то время как сервер приложений выполняет массу нужного генерирования кода, разработчики могут сфокусироваться на бизнес-логике.

Устанавливаемое программное обеспечение

Сервисы может предоставлять сервер приложений, от этого зависит количество и качество строительных блоков:

1. Webserver — чаще всего включают в поставку самый популярный и мощный Apache;
2. Web Container — позволяет выполнять JSP и сервлеты. Для Apache таким сервисом является Tomcat;
3. CORBA Agent — может предоставлять распределенную директорию для хранения CORBA объектов;
4. Messaging Service — брокер сообщений;
5. Transaction Service — уже из названия понятно, что это сервис транзакций;
6. JDBC — предлагает драйверы для подключения к базам данных, ведь именно серверу приложений придется общаться с БД и ему нужно уметь подключаться к используемой в вашей компании базе;
7. Java Mail — предоставляет сервис к SMTP;
8. JMS (Java Messaging Service) — обработка синхронных и асинхронных сообщений;
9. RMI (Remote Method Invocation) — вызов удаленных процедур.

Это основные блоки, которые может предоставлять тот или иной сервер приложений.

Заключение

Сервер приложений - это набор программного обеспечения, позволяющий распределить обработку данных по сети, организовать специально выделенные серверы для выполнения определенных задач, многозадачный режим выполнения программ пользователя

Многие серверы приложений позволяют реализовать приложения, устойчивые к сбоям. В настоящее время серверы приложений являются основой многих корпоративных решений.

Список литературы

http://www.compline-ufa.ru/tipi-serverov-klassifitatsiya#sp

http://www.km.ru/referats/57A4900A10B04830AB714B4FE978B816

https://www.webkursovik.ru/kartgotrab.asp?id=-158571