Практическое занятие № 1
Программы, выполняющиеся на клиент-ПК и сервере. Насыщенные Internet-приложения. Серверные web-приложения и web-сервисы.

Цель работы:

1. Изучение свойств программ, выполняющихся на клиент-ПК и сервере
2. Изучение свойств насыщенных Internet-приложений, серверных web-приложений и web-сервисов.

Оборудование и программное обеспечение:

1. ПК, локальная сеть.
2. ОС MS Windows 7, офисный пакет MS Office, браузер MS Internet Explorer.

План занятия:

1. Освоение теоретической части занятия в соответствии с разделами 1 - 4.
2. Выполнение практической части занятия.
3. Повторение и закрепление основных изученных понятий и терминов.
4. Ответы по контрольным вопросам.

1. Программы, выполняющиеся на клиент-ПК

Одним из типов программ, предназначенных для выполнения на клиент-ПК, являются сценарии, например, ***JavaScript*** (***VBScript***). Исходный текст сценария представляет собой часть Web-страницы, поэтому сценарий JavaScript передается клиенту вместе с документом, в состав которого он входит. Обрабатывая HTML-документ, браузер обнаруживает исходный текст сценария и запускает его на выполнение.

Ко всем программам, которые передаются с сервера на клиент-ПК и запускаются на выполнение, предъявляется одно общее требование: эти программы должны быть лишены возможности обращаться к ресурсам компьютера, на котором они выполняются. Такое требование вполне обосновано. Ведь передача по сети и запуск Java-апплетов и JavaScript-сценариев происходит автоматически без участия пользователя, поэтому работа этих программ должна быть абсолютно безопасной для компьютера. Другими словами, языки, предназначенные для создания программ, выполняющихся на клиент-ПК, должны быть абсолютно непригодны для написания вирусов и подобных программ.

2. Программы, выполняющиеся на сервере

Код программы, работающей на сервере, не передается клиенту. При получении от клиента специального запроса, предполагающего выполнение такой программы, сервер запускает ее и передает параметры, входящие в состав запроса. Средства для генерации подобного запроса обычно входят в состав HTML-документа.

Результаты своей работы программа оформляет в виде HTML-документа и передает их Web-серверу, а последний, в свою очередь, дополняет полученные данные HTTP-заголовком и передает их клиенту.

3. Насыщенные Internet-приложения

Насыщенное интернет-приложение (***Rich Internet application***) – еще один подход, который заключается в использовании Adobe Flash или Java-апплетов для полной или частичной реализации пользовательского интерфейса, поскольку большинство браузеров поддерживает эти технологии (как правило, с помощью плагинов).

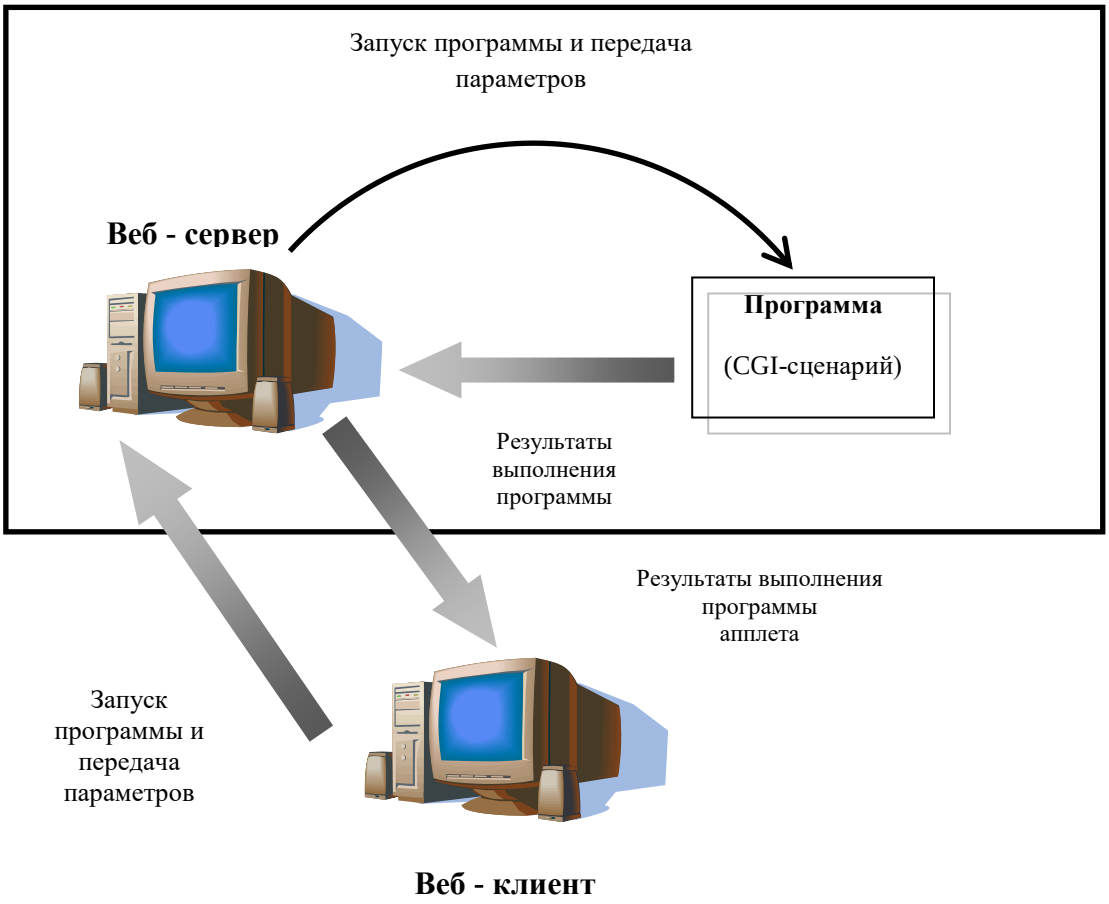
Возникновение данного подхода обусловлено тем, что в рамках Web-приложений с "тонким" клиентом взаимодействие пользователя с приложением реализуется в существенной степени через сервер, что требует отправки данных на сервер, получение ответа от сервера и перезагрузку страницы на стороне клиента.

При использовании Java-апплетов в состав HTML-документа включается специальный дескриптор, описывающий расположение файла, содержащего код апплета, на сервере. После того как клиент получает HTML-код документа, включающего апплет, он генерирует дополнительный запрос серверу. После того как сервер пересылает клиенту код апплета, сам апплет запускается на выполнение.

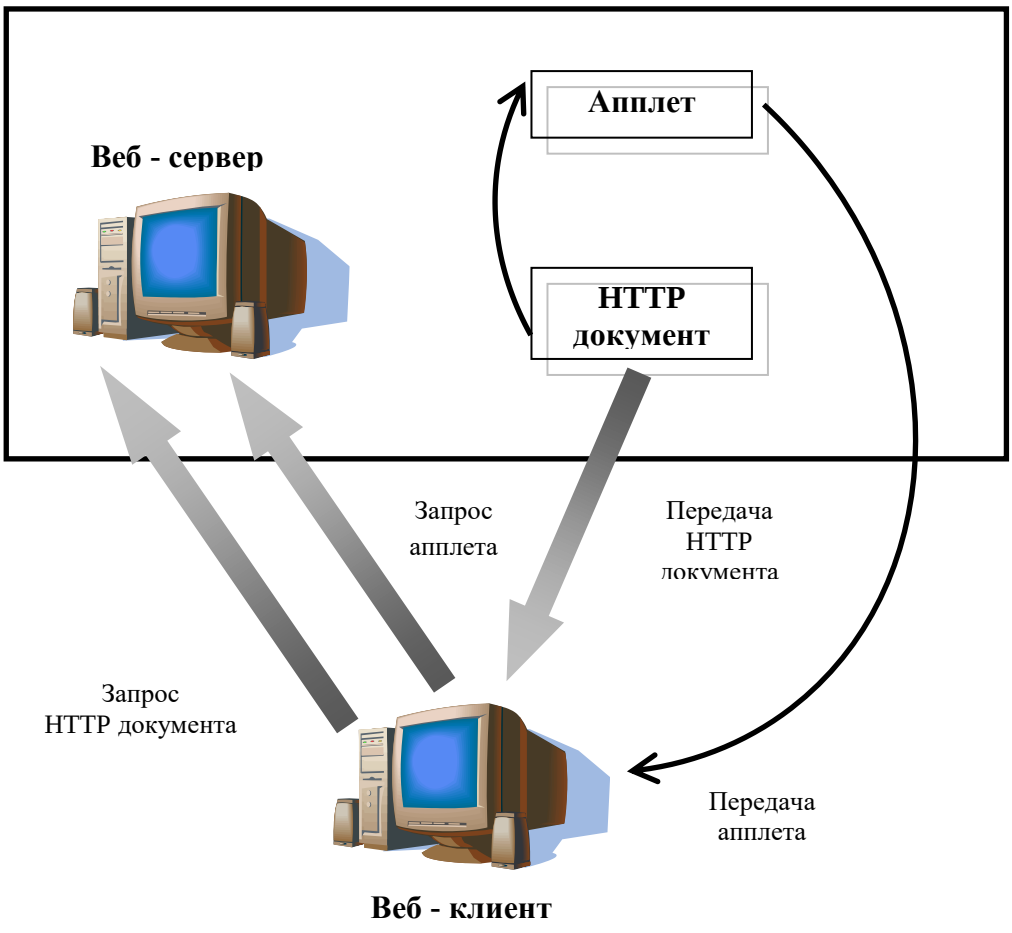
Практическая часть к разделам 1-3

Используя полученные понятия и определения, проиллюстрируйте процессы:

3.1. Взаимодействие клиента с программой, выполняющейся на сервере.



3.2. Передачу клиенту Java-апплета.



При использовании насыщенных Internet-приложений приходится сталкиваться со следующими проблемами:

- необходимость обеспечения безопасной среды выполнения («песочница»);
- для исполнения кода должно быть разрешено исполнение сценариев;
- потеря в производительности, т.к. выполняются на клиентской стороне;
- требуется много времени на загрузку;

Для разработки насыщенных Internet-приложений используются пакеты *Curl*, *Adobe Flex* и *Microsoft Silverlight*.

4. Серверные web-приложения и web-сервисы.

В корпоративных сетях компаний функционирует огромное число разнородных бизнес-приложений, созданных в разное время на базе разных технологий. Задача заключается в том, чтобы объединить разнородные Web-приложения и системы в единую среду. Эту задачу решает **Web-интеграция** с помощью следующих подходов:

- ***Интеграция на уровне представления***. Пользователь взаимодействует с приложением, ему дается *доступ к пользовательскому интерфейсу удаленных приложений*.
- ***Интеграция на уровне функциональности*** - обеспечение прямого доступа к бизнес-логике приложений. Это достигается непосредственным взаимодействием приложений с *API* (программному интерфейсу приложений) или же взаимодействием посредством *Web-сервисов*.
- ***Интеграция на уровне данных*** - доступ к одной или нескольким *базам данных*, используемых удаленным приложением.
- ***Комплексная интеграция***. Коммерческие решения по Web-интеграции, как правило, включают все три типа интеграции

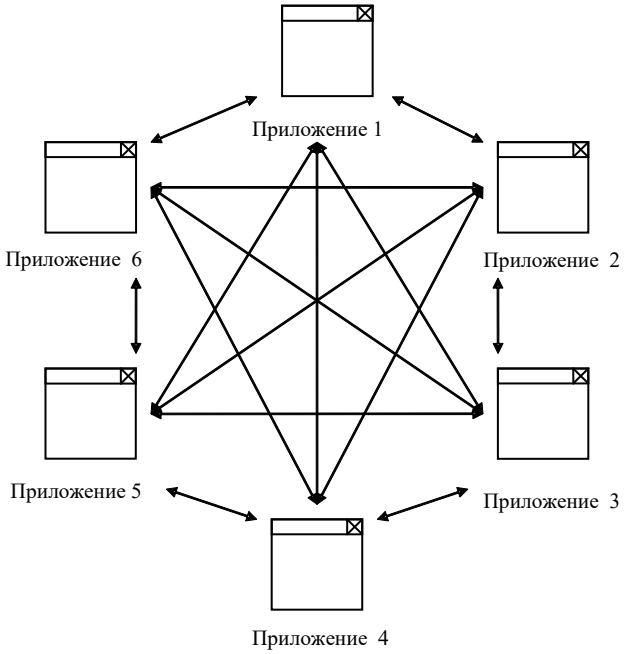
Использование веб-интеграции выгодно по многим причинам:

- *Веб-интеграция* позволяет развертывать информационные системы на базе сторонних приложений без необходимости разбираться в их родительских системах, программных средах и архитектурах баз данных.
- *SOA* и *веб-сервисы* используют программный язык и платформо-независимые интерфейсы между приложениями корпоративной инфраструктуры ИТ. Это дает очевидные преимущества в поддержке, управляемости, развертывании информационных сетей.
- Веб-интеграция позволяет конструировать комплексную функциональность, комбинируя разнородные компоненты посредством протоколов веб-сервисов.
- Веб-интеграция позволяет использовать веб-сервисы разработчиков.
- Веб-интеграция позволяет развивать программные интерфейсы приложений через протоколы веб-сервисов без программирования.

Для веб-интеграции обычно используется коммерческое ПО или популярные технологии, такие как *PHP/Python/Perl, XForms, SOAP* и т.д.

4.1. Интеграция на основе XML

Большое количество систем, стандартов и технологий приводит к тому, что эффективно связать разные источники данных в одну систему не получается. Даже такие, на первый взгляд однородные источники, как системы управления базами данных, применяют языки запросов и форматы представления выбираемой информации, которые редко полностью совместимы между собой. Как следствие, проекты интеграции в таких условиях требуют больших усилий - требуется вникать в детали различных баз данных, протоколов, операционных систем и так далее. В результате интеграция нескольких приложений или систем реализуется по схеме, представленной ниже:



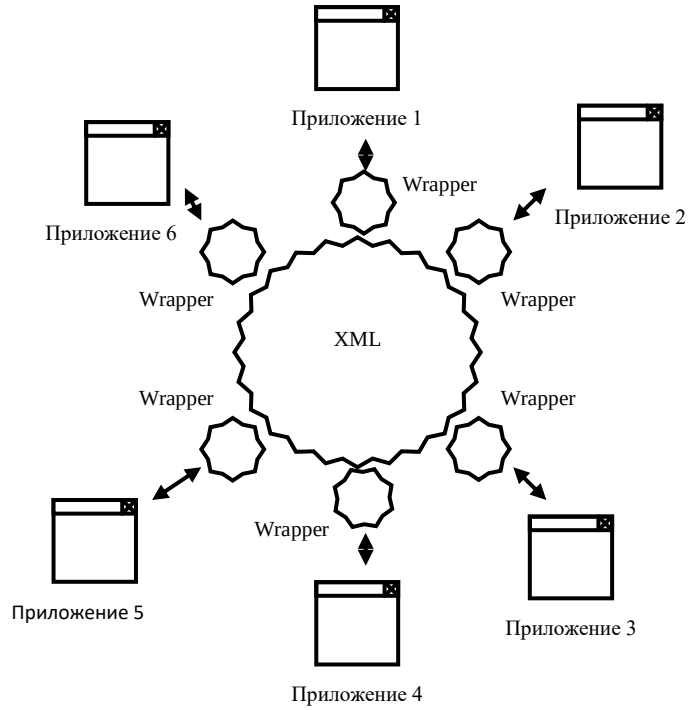
Заставить разные системы работать вместе - чрезвычайно трудоемкая задача. Идея использования XML в интеграции информационных систем сводится к созданию общего XML-языка, которым могла бы пользоваться каждая из них.

Такое решение сразу же намного упрощает проект. Вместо реализации взаимодействия между каждой парой систем следует всего лишь научить каждую из них "говорить" на XML языке. Иначе говоря, все сводится к разработке нескольких *врапперов* (wrapper - упаковщик, программное средство создания системной оболочки для стандартизации внешних обращений и изменения функциональной ориентации действующей системы), которые будут переводить со стандартного XML-языка интегрированной системы на язык, понятный каждой системе в отдельности.

- средства разработки и стандартные библиотеки для XML существуют практически на всех платформах и для большинства популярных языков программирования;
- методы работы с XML достаточно стандартны для того, чтобы в разных системах можно было пользоваться одинаковыми приемами;
- информация, оформленная в виде XML, может обрабатываться не только машинами, но и человеком (что намного облегчает отладку).

В принципе, интеграция по XML-схеме не отличается коренным образом от интеграции на основе любого другого общего стандарта. Вместе с тем, она имеет целый ряд весомых преимуществ:

- XML языки не зависят от аппаратных и программных платформ, что позволяет связывать разнородные системы;
- выразительная мощность XML достаточно велика для того, чтобы описать данные практически любой сложности;



Интеграция на основе XML практически реализуется в рамках протоколов:

- *XML-RPC*. Это протокол удаленного вызова процедур с передачей данных в формате XML через TCP-порт 80, т.е. HTTP -порт.
- *WDDX* (Web Distributed Exchange). Представляет собой механизм обмена сложными структурами данных по протоколу HTTP. Протокол базируется не на структурах, а на событиях.
- *ebXML* (electronic buisiness XML) – XML для электронного бизнеса. Основное назначение – предоставление открытой XML-инфраструктуры, обеспечивающей безопасное глобальное использование информации электронного бизнеса.
- *Веб-сервисы (веб-службы)*.

Веб-сервисы (веб-службы)

Веб-сервис (web service) — программная система, имеющая идентификатор URI, и общедоступные интерфейсы которой определены на языке XML. Описание этой программной системы может быть найдено другими приложениями, которые могут взаимодействовать с ней в соответствии с этим описанием посредством сообщений, основанных на XML, и передаваемых с помощью интернет-протоколов. Веб-служба является единицей модульности при использовании *сервис-ориентированной архитектуры* приложения.

Сервис-ориентированная архитектура (SOA, service-oriented architecture) — модульный подход к разработке программного обеспечения, основанный на использовании сервисов со стандартизированными интерфейсами.

В основе SOA лежат принципы многократного использования функциональных элементов ИТ, унификации типовых операционных процессов . Компоненты программы могут быть распределены по разным узлам сети, и предлагаются как независимые и слабо связанные, заменяемые сервисы-приложения.

Интерфейс компонентов SOA-программы осуществляет инкапсуляцию деталей реализации конкретного компонента (ОС, языка программирования и т. п).

SOA хорошо зарекомендовала себя при построении крупных корпоративных программных систем. Целый ряд разработчиков и интеграторов предлагают инструменты и решения на основе SOA (например, платформы Microsoft .NET , IBM WebSphere, SAP NetWeaver, Diasoft и др.).

Веб-сервисы .NET имеют следующие достоинства:

- *Открытость стандартов*. В веб-сервисах отсутствуют какие-либо скрытые или недоступные элементы. Каждый аспект технологии, от способа поиска веб-сервисы до ее описания и организации связи с ней, определен общедоступными стандартами.
- *Межплатформенность*. Язык программирования, который позволяет создавать XML-документы и отправлять информацию посредством HTTP, позволяет взаимодействовать с любым веб-сервисом. Можно получать веб-услугу из системы, отличной от .NET.
- *Простота*.
- *Поддержка сообщений на понятном человеку языке*. Переход от двоичных стандартов, применяемых в COM и CORBA, к XML-тексту позволил упростить исправление ошибок и обеспечил возможность осуществлять взаимодействие с веб-сервисами по обычным каналам HTTP.