Федеральное агентство по образованию
АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ГОУВПО «АмГУ»

УТВЕРЖДАЮ
Зав. кафердой ИУС
_____А.В.Бушманов
«___»_____2007г.

ПРОГРАММИРОВАНИЕ В ИНТЕРНЕТ

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ПО ДИСЦИПЛИНЕ

для специальности: 230102 – Автоматизированные системы обработки информации и управления
230201 – Информационные системы и технологии

Составитель ассистент Степанов А.Е.

Факультет математики и информатики

Кафедра информационных и управляющих систем

Благовещенск
2007 г.

Печатается по решению
редакционно-издательского совета
факультета математики и информатики
Амурского государственного
университета

А.Е. Степанов

Учебно-методический комплекс по дисциплине «Программирование в интернет» для студентов очной формы обучения специальности 230102 – Автоматизированные системы обработки информации и управления, 230201 – Информационные системы и технологии. – Благовещенск: Амурский гос. ун-т, 2007. – 51 с.

Учебно-методические рекомендации ориентированны на оказание помощи студентам очной формы обучения по специальностям 230102 – Автоматизированные системы обработки информации и управления, 230201 – Информационные системы и технологии, для успешного освоения дисциплины «Программирование в интернет»

СОДЕРЖАНИЕ

1. РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

Федеральное агентство по образованию

АМУРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
(ГОУВПО « АмГУ»)

УТВЕРЖДАЮ
Проректор по УНР

_____Е.С.Астапова
«_____»_____2007 г.

РАБОЧАЯ ПРОГРАММА
по дисциплине «Программирование в Интернет»
для специальности 230201 «Информационные системы и технологии»

Курс	4 курс
Семестр	8 семестр
Лекции	
Лабораторные занятия	15 часов
Экзамен	
Контрольная работа	
Самостоятельная работа	15 часов
Всего часов нагрузки	30 часов

Составитель	СТЕПАНОВ А.Е., ассистент
Факультет	Математики и информатики
Кафедра	ИУС

2007

Дисциплина «Программирование в интернет» для специальности 230201 «Информационные системы и технологии» введена по решению УМСС

Рабочая программа обсуждена на заседании кафедры Информационных и управляющих систем

«_____» _____2007 г., протокол №_____

Заведующий кафедрой _____ А.В.Бушманов

Рабочая программа одобрена на заседании УМС 230201 «Информационные системы и технологии»

«_____» _____2007 г., протокол №_____

Председатель_____ А.В.Бушманов

СОГЛАСОВАНО
Начальник УМУ
_____Г.Н.Торопчина

«____»_____2007 г.

СОГЛАСОВАНО
Председатель УМС факультета
_____Е.Л.Еремин

«_____»_____2007 г.

СОГЛАСОВАНО
Заведующий выпускающей кафедрой
_____А.В.Бушманов

«____»_____2007 г.

1. ЦЕЛИ И ЗАДАЧИ ДИСЦИПЛИНЫ, ЕЕ МЕСТО В УЧЕБНОМ ПРОЦЕССЕ

Сегодня Интернет объединяет множество различных сетей,миллионы компьютеров, более 100 миллионов пользователей по всем континентам.В российском Интернет зарегистрировано свыше 10 миллионов документов. Интернет в целом своём развивается качественно.

Целью дисциплины является изучение методов программирования приложений для Интернет, создание Интернет - приложений на основе языка гипертекстовой разметки HTML и языка программирования высокого уровня Delphi.

2. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ

Дисциплина «Программирование в интернет» для специальности 230201 «Информационные системы и технологии» введена по решению УМСС.

Наименование тем, их содержание, объем

1. Общие сведения и понятия. (4 час)

История Интернет. Семейство протоколов TCP/IP. 7ми уровневая модель сетевого взаимодействия OSI. Локальные вычислительные сети на основе Ethernet. Топологии ЛВС. Основы Ethernet. Введение в теорию интернет-программирования.

2. Delphi и Internet . (2 час)

Статические документы Интернет. Язык разметки гипертекстов. Динамическое создание документов WEB с помощью Delphi. Приложение WEB-сервера.

3.Элементы управления страниц Internet и IntenetExpress. (2 час)

Компоненты для работы с базами данных панели Internet. Подключение к БД. Компонент TPageProducer. Компонент TQueryTableProducer.

4. Разработка компонентов ActiveX. (4 час)

Создание объекта ActiveForm. Тестирование компонентов ActiveX. Страницы свойств. Пользовательские страницы свойств. Создание компонентов ActiveX.

5. Создание многопользовательских распределённых приложений с использованием интерфейса сокетов. (2 час)

Понятие «порта». Типы сокетных соединений. Описание сокетов. Сокеты

клиента. Серверные сокеты. Работа с событиями сокетов.

6. Элементы управления страницы FastNet. (4 час)

Компонент TNMDayTime. Компонент TNMTime. Компоненты TNMMsg, TNMMsgServ. Компонент TNMEcho. Компонент TNMFinger. Компонент TNMFTP. Компонент TNMHTTP. Компонент TNMNNTP. Компонент TNMPOP3. Компонент TNMSMTP. Компоненты TNMStrm, TNMStrmServ. Компонент TNMUDP. Компонент TNMURL. Компонент TNMUUProcessor.

Самостоятельная работа студентов

В качестве самостоятельной работы по дисциплине «Введение в распознавание сигналов» студенты готовят рефераты по следующим темам:

1. Общие сведения о компонентах страницы Internet.
2. Компонент TClientSocket
3. Компонент TServerSoket
4. Компонент TWebDispatcher
5. Компонент TCustomWebDispatcher
6. Компонент TPageProducer
7. Компонент TWebBrowser
8. Общие сведения о компонентах страницы InternetExpress.
9. Компонент TXMLBroker
10. Компонент TMidasPageProducer

Вопросы к зачету

1. История Интернет.
2. Семейство протоколов TCP/IP.
3. Уровни модели OSI.
4. Локальные вычислительные сети на основе Ethernet
5. Топологии ЛВС
6. Основы Ethernet
7. Введение в теорию интернет-программирования
8. Статические документы Интернет
9. Язык разметки гипертекстов
10. Динамическое создание документов WEB с помощью Delphi
11. Приложение WEB-сервера
12. Компоненты для работы с базами данных панели Internet
13. Подключение к БД
14. Компонент TPageProducer
15. Компонент TQueryTableProducer
16. Создание объекта ActiveForm

17. Понятие «порта». Типы сокетных соединений
18. Описание сокетов. Сокеты клиента. Серверные сокеты. Работа с событиями сокетов.
19. Работа с событиями сокетов
20. Элементы управления страницы FastNet
21. Общие сведения о компонентах страницы Internet
22. Компоненты TClientSocket и TServerSoket
23. Общие сведения о компонентах страницы InternetExpress
24. Компоненты TXMLBroker и TMidasPageProducer

Виды контроля

Для проверки эффективности преподавания дисциплины проводится контроль знаний студентов. При этом используются следующие виды контроля:

- *текущий контроль* за аудиторной и самостоятельной работой обучаемых осуществляется во время проведения аудиторных занятий посредством устного опроса, проведения контрольных работ;
- *промежуточный контроль* осуществляется два раза в семестр в виде анализа итоговых отчетов на аттестационные вопросы;
- *итоговый контроль* в виде зачета осуществляется после успешного прохождения студентами текущего и промежуточного контроля.

Требования к знаниям студентов, предъявляемые на зачете

Для получения зачета студент должен посещать занятия, проявлять активность в аудитории, знать теоретический материал в необходимом объеме.

3. УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

Перечень обязательной (основной) литературы

1. Козлов А.В. Программирование в интернет. / Учебник для вузов. – М.: Бином, 2001. – 366 с.

2. Петр Дарахвелидзе, Евгений Марков. Разработка Web-служб средствами Delphi 2003.

Перечень дополнительной литературы

1. Гофман В., Хомоненко А. Delphi 5: . – СПб.:БХВ ,2000.
2. Дейтел Х.М. и др. Как программировать на XML. – М.: Бином, 2001
3. Дуглас Э. Крамер. Сети TCP/IP. – СПб.: Вильямс, 2003.

4. УЧЕБНО-МЕТОДИЧЕСКАЯ (ТЕХНОЛОГИЧЕСКАЯ) КАРТА ДИСЦИПЛИНЫ

Номер недели	Номер темы	Используемые нагляд. и метод пособия	Самостоятельная работа студентов		Формы контроля
			Содержание	часы	
1	1	1,2 – осн. 1,2,5 – доп.	Выбор темы самостоятельной работы	2	
2	1	1,2 – осн. 1,2,5 – доп.			
3	2	2 – осн. 1,2,5 – доп.	Поиск литературы по теме самостоятельной работы	6	собес ед.
4	3	2 – осн. 1,2,5 – доп.			
5	4	2 – осн. 1,14,4,8 – доп.	Работа с литературой и поиск информации в сети Интернет по новейшим достижениям в области программирования в Интернет	7	
6	5	2,3 – осн. 1,10,6 – доп.			
7	6	2,3 – осн. 1,10,6 – доп.			

Условные обозначения:

осн. – основная литература
доп. – дополнительная литература
к.р. – контрольная работа
собесед. – собеседование

2 ГРАФИК СОМОСТОЯТЕЛЬНОЙ РАБОТЫ СТУДЕНТОВ ПО ДИСЦИПЛИНЕ

Содержание самостоятельной работы студентов	Объем самостоятельной работы студентов, в часах	Сроки выполнения самостоятельной работы
Работа с основной, дополнительной, а так же с периодической литературой при подготовке к семинарским занятиям	3	К каждому семинарскому занятию
Общие сведения о сетях передачи данных	2	ПЗ-1
Среда программирования Delphi	4	ПЗ-2
Компоненты, используемые для создания сетевых приложений	4	ПЗ-3
Технологии, используемые при создании приложений для сети «Интернет»	2	ПЗ-4
Итого:	15	

3 ПЛАН-КОНСПЕКТ ЛЕКЦИЙ ПО ДИСЦИПЛИНЕ «ПРОГРАММИРОВАНИЕ В ИНТЕРНЕТ»

Истоки Интернет

Первым документальным описанием социального взаимодействия, которое станет возможным благодаря сети, была серия заметок, написанных Дж. Ликлайдером (J.C.R. Licklider) из Массачусетского технологического института (MIT) в августе 1962 года. В этих заметках обсуждалась концепция "Галактической сети" ("Galactic Network"). Автор предвидел создание глобальной сети взаимосвязанных компьютеров, с помощью которой каждый сможет быстро получать доступ к данным и программам, расположенным на любом компьютере. По духу эта концепция очень близка к современному состоянию Интернет. В октябре 1962 года Ликлайдер стал первым руководителем исследовательского компьютерного проекта в Управлении перспективных исследований и разработок Министерства обороны США (Defence Advanced Research Projects Agency, DARPA). Ликлайдер сумел убедить своих преемников по работе в DARPA — Ивана Сазерленда (Ivan Sutherland) и Боба Тейлора (Bob Taylor), а также исследователя из MIT Лоуренса Робертса в важности этой сетевой концепции.

Леонард Клейнрок из MIT опубликовал первую статью по теории пакетной коммутации [6] в июле 1961 года, а первую книгу — в 1964 году. Клейнрок убедил Робертса в теоретической обоснованности пакетных коммуникаций (в противоположность коммутации соединений), что явилось важным шагом на пути к созданию компьютерных сетей. Другим ключевым шагом должна была стать

организация реального межкомпьютерного взаимодействия. Для исследования этого вопроса Робертс совместно с Томасом Меррилом (Thomas Merrill) в 1965 году связал компьютер TX-2, расположенный в Массачусетсе, с ЭВМ Q-32, находившейся в Калифорнии. Связь осуществлялась по низкоскоростной коммутируемой телефонной линии. Таким образом была создана первая в истории (хотя и маленькая) нелокальная компьютерная сеть. Результатом эксперимента стало понимание того, что компьютеры с разделением времени могут успешно работать вместе, выполняя программы и осуществляя выборку данных на удаленной машине. Стало ясно и то, что телефонная система с коммутацией соединений абсолютно непригодна для построения компьютерной сети. Убежденность Клейнрока в необходимости пакетной коммутации получила еще одно подтверждение.

В конце 1966 года Робертс начал работать в DARPA над концепцией компьютерной сети. Довольно быстро появился план ARPANET, опубликованный в 1967 году. На конференции, где Робертс представлял свою статью, был сделан еще один доклад о концепции пакетной сети. Его авторами были английские ученые Дональд Дэвис (Donald Davies) и Роджер Скентльбьюри (Roger Scantlebury) из Национальной физической лаборатории (NPL). Скентльбьюри рассказал Робертсу о работах, выполнявшихся в NPL, а также а работах Пола Бэрена (Paul Baran) и его коллег из RAND (американская бесприбыльная организация, занимающаяся стратегическими исследованиями и разработками). В 1964 году группа сотрудников RAND написала статью по сетям с пакетной коммутацией для надежных голосовых коммуникаций в военных системах [1]. Оказалось, что работы в MIT (1961-1967), RAND (1962-1965) и NPL (1964-1967) велись параллельно при полном отсутствии информации о деятельности коллег. Разговор Робертса с сотрудниками NPL привел к заимствованию слова "пакет" и решению увеличить предлагаемую скорость передачи по каналам проектируемой сети ARPANET с 2.4 Кбит/с до 50 Кбит/с.

В августе 1968 года, после того как Робертс и организации, финансируемые из бюджета DARPA, доработали общую структуру и спецификации ARPANET, DARPA выпустило запрос на расценки (Request For Quotation, RFQ), организовав открытый конкурс на разработку одного из ключевых компонентов — коммутатора пакетов, получившего название Интерфейсный процессор сообщений (Interface Message Processor, IMP). В декабре 1968 года конкурс выиграла группа во главе с Фрэнком Хартом (Frank Heart) из компании Bolt, Beranek и Newman (BBN). После этого роли распределились следующим образом. Команда из BBN работала над Интерфейсными процессорами сообщений, Боб Кан принимал активное участие в проработке архитектуры ARPANET, Робертс совместно с Ховардом Фрэнком (Howard Frank) и его группой из Network Analysis Corporation проектировали и оптимизировали топологию и экономические аспекты сети, группа Клейнрока из Калифорнийского университета в Лос-Анджелесе (UCLA) готовила систему измерения характеристик сети.

Благодаря тому, что Клейнрок уже в течение нескольких лет был известен как автор теории пакетной коммутации и как специалист по анализу, проектированию и измерениям, его Сетевой измерительный центр в UCLA был выбран в качестве первого узла ARPANET. Тогда же, в сентябре 1969 года, компания BBN установила в Калифорнийском университете первый Интерфейсный процессор сообщений и

подключила к нему первый компьютер. Второй узел был образован на базе проекта Дуга Энгельбарта (Doug Engelbart) "Наращивание человеческого интеллекта" в Стэнфордском исследовательском институте (SRI). (Следует отметить, что частью проекта Энгельбарта была ранняя гипертекстовая система NLS.) В SRI организовали Сетевой информационный центр, который возглавила Элизабет Фейнлер (Elizabeth (Jake) Feinler). В функции центра входило поддержание таблиц соответствия между имена и адресами компьютеров, а также обслуживание каталога запросов на комментарии и предложения (Request For Comments, RFC). Через месяц, когда SRI подключили к ARPANET, из лаборатории Клейнрока было послано первое межкомпьютерное сообщение. Двумя следующими узлами ARPANET стали Калифорнийский университет в городе Санта-Барбара (UCSB) и Университет штата Юта. В этих университетах развивались проекты по прикладной визуализации. Глен Галлер (Glen Culler) и Бартон Фрайд (Burton Fried) из UCSB исследовали методы отображения математических функций с использованием дисплеев с памятью, позволяющих справиться с проблемой перерисовки изображения по сети. Роберт Тейлор и Иван Сазерленд в Юте исследовали методы рисования по сети трехмерных сцен. Таким образом, к концу 1969 года четыре компьютера были объединены в первоначальную конфигурацию ARPANET. Взошел первый росток Интернет. Следует отметить, что уже на этой ранней стадии велись исследования как по сетевой инфраструктуре, так и по сетевым приложениям. Данная традиция не нарушена и в наши дни.

В последующие годы число компьютеров, подключенных к ARPANET, быстро росло.

Одновременно велись работы по созданию функционально полного протокола межкомпьютерного взаимодействия и другого сетевого программного обеспечения. В декабре 1970 года Сетевая рабочая группа (Network Working Group, NWG) под руководством С. Крокера завершила работу над первой версией протокола, получившего название Протокол управления сетью (Network Control Protocol, NCP). После того, как в 1971-1972 годах были выполнены работы по реализации NCP на узлах ARPANET, пользователи сети наконец смогли приступить к разработке приложений.

В октябре 1972 года Роберт Кан организовал большую, весьма успешную демонстрацию ARPANET на Международной конференции по компьютерным коммуникациям (International Computer Communication Conference, ICCC). Это был первый показ на публике новой сетевой технологии. Также в 1972 году появилось первое "горячее" приложение — электронная почта. В марте Рэй Томлинсон (Ray Tomlinson) из BBN, движимый необходимостью создания для разработчиков ARPANET простых средств координации, написал базовые программы пересылки и чтения электронных сообщений. В июле Робертс добавил к этим программам возможности выдачи списка сообщений, выборочного чтения, сохранения в файле, пересылки и подготовки ответа. С тех пор более чем на десять лет электронная почта стала крупнейшим сетевым приложением. Для своего времени электронная почта стала тем же, чем в наши дни является Всемирная паутина — исключительно мощным катализатором роста всех видов межперсональных потоков данных.

Стек протоколов TCP\IP

Сеть Internet - это сеть сетей, объединяющая как локальные сети, так и глобальные сети типа NSFNET. Поэтому центральным местом при обсуждении принципов построения сети является семейство протоколов межсетевого обмена TCP/IP.

Под термином "TCP/IP" обычно понимают все, что связано с протоколами TCP и IP. Это не только собственно сами проколы с указанными именами, но и протоколы построенные на использовании TCP и IP, и прикладные программы.

Главной задачей стека TCP/IP является объединение в сеть пакетных подсетей через шлюзы. Каждая сеть работает по своим собственным законам, однако предполагается, что шлюз может принять пакет из другой сети и доставить его по указанному адресу. Реально, пакет из одной сети передается в другую подсеть через последовательность шлюзов, которые обеспечивают сквозную маршрутизацию пакетов по всей сети. В данном случае, под шлюзом понимается точка соединения сетей. При этом соединяться могут как локальные сети, так и глобальные сети. В качестве шлюза могут выступать как специальные устройства, маршрутизаторы, например, так и компьютеры, которые имеют программное обеспечение, выполняющее функции маршрутизации пакетов. Маршрутизация - это процедура определения пути следования пакета из одной сети в другую.

Такой механизм доставки становится возможным благодаря реализации во всех узлах сети протокола межсетевого обмена IP. Если обратиться к истории создания сети Internet, то с самого начала предполагалось разработать спецификации сети коммутации пакетов. Это значит, что любое сообщение, которое отправляется по сети, должно быть при отправке "нашинковано" на фрагменты. Каждый из фрагментов должен быть снабжен адресами отправителя и получателя, а также номером этого пакета в последовательности пакетов, составляющих все сообщение в целом. Такая система позволяет на каждом шлюзе выбирать маршрут, основываясь на текущей информации о состоянии сети, что повышает надежность системы в целом. При этом каждый пакет может пройти от отправителя к получателю по своему собственному маршруту. Порядок получения пакетов получателем не имеет большого значения, т.к. каждый пакет несет в себе информацию о своем месте в сообщении. При создании этой системы принципиальным было обеспечение ее живучести и надежной доставки сообщений, т.к. предполагалось, что система должна была обеспечивать управление Вооруженными Силами США в случае нанесения ядерного удара по территории страны.

Структура стека протоколов TCP/IP

При рассмотрении процедур межсетевого взаимодействия всегда опираются на стандарты, разработанные International Standard Organization (ISO). Эти стандарты получили название "Семиуровневой модели сетевого обмена" или в английском варианте "Open System Interconnection Reference Model" (OSI Ref.Model). В данной модели обмен информацией может быть представлен в виде стека, представленного на рисунке 2.1. Как видно из рисунка, в этой модели определяется все - от стандарта

физического соединения сетей до протоколов обмена прикладного программного обеспечения. Дадим некоторые комментарии к этой модели.

Физический уровень данной модели определяет характеристики физической сети передачи данных, которая используется для межсетевого обмена. Это такие параметры, как: напряжение в сети, сила тока, число контактов на разъемах и т.п. Типичными стандартами этого уровня являются, например RS232C, V35, IEEE 802.3 и т.п.

Рис. 1. Семиуровневая модель протоколов межсетевого обмена OSI

К канальному уровню отнесены протоколы, определяющие соединение, например, SLIP (Strial Line Internet Protocol), PPP (Point to Point Protocol), NDIS, пакетный протокол, ODI и т.п. В данном случае речь идет о протоколе взаимодействия между драйверами устройств и устройствами, с одной стороны, а с другой стороны, между операционной системой и драйверами устройства. Такое определение основывается на том, что драйвер - это, фактически, конвертор данных из оного формата в другой, но при этом он может иметь и свой внутренний формат данных.

К сетевому (межсетевому) уровню относятся протоколы, которые отвечают за отправку и получение данных, или, другими словами, за соединение отправителя и получателя. Вообще говоря, эта терминология пошла от сетей коммутации каналов, когда отправитель и получатель действительно соединяются на время работы каналом связи. Применительно к сетям TCP/IP, такая терминология не очень приемлема. К этому уровню в TCP/IP относят протокол IP (Internet Protocol). Именно здесь определяется отправитель и получатель, именно здесь находится необходимая информация для доставки пакета по сети.

Транспортный уровень отвечает за надежность доставки данных, и здесь, проверяя контрольные суммы, принимается решение о сборке сообщения в одно целое. В Internet транспортный уровень представлен двумя протоколами TCP (Transport Control Protocol) и UDP (User Datagramm Protocol). Если предыдущий

уровень (сетевой) определяет только правила доставки информации, то транспортный уровень отвечает за целостность доставляемых данных.

Уровень сессии определяет стандарты взаимодействия между собой прикладного программного обеспечения. Это может быть некоторый промежуточный стандарт данных или правила обработки информации. Условно к этому уровню можно отнеси механизм портов протоколов TCP и UDP и Berkeley Sockets. Однако обычно, рамках архитектуры TCP/IP такого подразделения не делают.

Уровень обмена данными с прикладными программами (Presentation Layer) необходим для преобразования данных из промежуточного формата сессии в формат данных приложения. В Internet это преобразование возложено на прикладные программы.

Уровень прикладных программ или приложений определяет протоколы обмена данными этих прикладных программ. В Internet к этому уровню могут быть отнесены такие протоколы, как: FTP, TELNET, HTTP, GOPHER и т.п.

Вообще говоря, стек протоколов TCP отличается от только что рассмотренного стека модели OSI. Обычно его можно представить в виде схемы, представленной на рисунке 2.

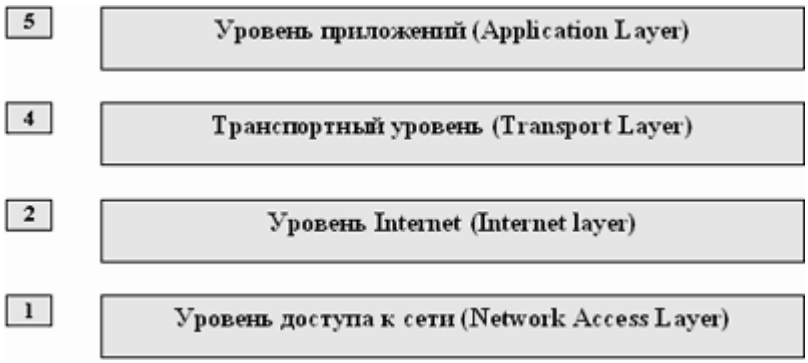


Рис. 2. Структура стека протоколов TCP/IP

В этой схеме на уровне доступа к сети располагаются все протоколы доступа к физическим устройствам. Выше располагаются протоколы межсетевого обмена IP, ARP, ICMP. Еще выше основные транспортные протоколы TCP и UDP, которые кроме сбора пакетов в сообщения еще и определяют какому приложению необходимо данные отправить или от какого приложения необходимо данные принять. Над транспортным уровнем располагаются протоколы прикладного уровня, которые используются приложениями для обмена данными.

Базируясь на классификации OSI (Open System Integration) всю архитектуру протоколов семейства TCP/IP попробуем сопоставить с эталонной моделью (рисунок 3).

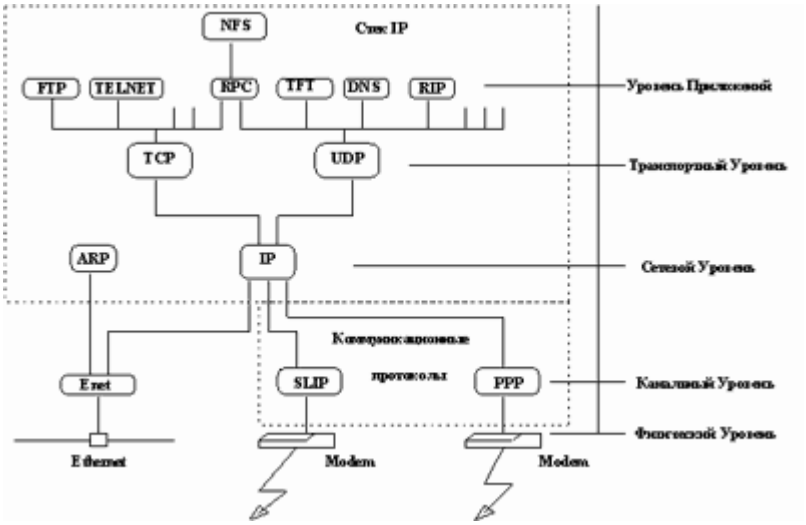


Рис. 3. Схема модулей, реализующих протоколы семейства TCP/IP в узле сети

Прямоугольниками на схеме обозначены модули, обрабатывающие пакеты, линиями - пути передачи данных. Прежде чем обсуждать эту схему, введем необходимую для этого терминологию.

Драйвер - программа, непосредственно взаимодействующая с сетевым адаптером.

Модуль - это программа, взаимодействующая с драйвером, с сетевыми прикладными программами или с другими модулями.

Схема приведена для случая подключения узла сети через локальную сеть Ethernet, поэтому названия блоков данных будут отражать эту специфику.

Сетевой интерфейс - физическое устройство, подключающее компьютер к сети. В нашем случае - карта Ethernet.

Кадр - это блок данных, который принимает/отправляет сетевой интерфейс.

IP-пакет - это блок данных, которым обменивается модуль IP с сетевым интерфейсом.

UDP-датаграмма - блок данных, которым обменивается модуль IP с модулем UDP.

TCP-сегмент - блок данных, которым обменивается модуль IP с модулем TCP.

Прикладное сообщение - блок данных, которым обмениваются программы сетевых приложений с протоколами транспортного уровня.

Инкапсуляция - способ упаковки данных в формате одного протокола в формат другого протокола. Например, упаковка IP-пакета в кадр Ethernet или TCP-сегмента в IP-пакет. Согласно словарю иностранных слов термин "инкапсуляция" означает "образование капсулы вокруг чужих для организма веществ (инородных тел, паразитов и т.д.)". В рамках межсетевого обмена понятие инкапсуляции имеет несколько более расширенный смысл. Если в случае инкапсуляции IP в Ethernet речь идет действительно о помещении пакета IP в качестве данных Ethernet-фрейма, или, в случае инкапсуляции TCP в IP, помещение TCP-сегмента в качестве данных в IP-пакет, то при передаче данных по коммутируемым каналам происходит дальнейшая "нарезка" пакетов теперь уже на пакеты SLIP или фреймы PPP.

Рис. 4. Инкапсуляция протоколов верхнего уровня в протоколы TCP/IP

Вся схема (рисунок 4) называется стеком протоколов TCP/IP или просто стеком TCP/IP. Чтобы не возвращаться к названиям протоколов расшифруем аббревиатуры TCP, UDP, ARP, SLIP, PPP, FTP, TELNET, RPC, TFTP, DNS, RIP, NFS:

TCP - Transmission Control Protocol - базовый транспортный протокол, давший название всему семейству протоколов TCP/IP.

UDP - User Datagram Protocol - второй транспортный протокол семейства TCP/IP. Различия между TCP и UDP будут обсуждены позже.

ARP - Address Resolution Protocol - протокол используется для определения соответствия IP-адресов и Ethernet-адресов.

SLIP - Serial Line Internet Protocol (Протокол передачи данных по телефонным линиям).

PPP - Point to Point Protocol (Протокол обмена данными "точка-точка").

FTP - File Transfer Protocol (Протокол обмена файлами).

TELNET - протокол эмуляции виртуального терминала.

RPC - Remote Process Control (Протокол управления удаленными процессами).

TFTP - Trivial File Transfer Protocol (Тривиальный протокол передачи файлов).

DNS - Domain Name System (Система доменных имен).

RIP - Routing Information Protocol (Протокол маршрутизации).

`NFS - Network File System (Распределенная файловая система и система сетевой печати).

При работе с такими программами прикладного уровня, как FTP или telnet, образуется стек протоколов с использованием модуля TCP, представленный на рисунке 5.

Рис. 5. Стек протоколов при использовании модуля TCP

При работе с прикладными программами, использующими транспортный протокол UDP, например, программные средства Network File System (NFS),

используется другой стек, где вместо модуля TCP будет использоваться модуль UDP (рисунок 6).

Рис. 6. Стек протоколов при работе через транспортный протокол UDP

При обслуживании блочных потоков данных модули TCP, UDP и драйвер ENET работают как мультиплексоры, т.е. перенаправляют данные с одного входа на несколько выходов и наоборот, с многих входов на один выход. Так, драйвер ENET может направить кадр либо модулю IP, либо модулю ARP, в зависимости от значения поля "тип" в заголовке кадра. Модуль IP может направить IP-пакет либо модулю TCP, либо модулю UDP, что определяется полем "протокол" в заголовке пакета.

Получатель UDP-датаграммы или TCP-сообщения определяется на основании значения поля "порт" в заголовке датаграммы или сообщения.

Все указанные выше значения прописываются в заголовке сообщения модулями на отправляющем компьютере. Так как схема протоколов - это дерево, то к его корню ведет только один путь, при прохождении которого каждый модуль добавляет свои данные в заголовок блока. Машина, принявшая пакет, осуществляет демультиплексирование в соответствии с этими отметками.

Технология Internet поддерживает разные физические среды, из которых самой распространенной является Ethernet. В последнее время большой интерес вызывает подключение отдельных машин к сети через TCP-стек по коммутируемым (телефонным) каналам. С появлением новых магистральных технологий типа ATM или FrameRelay активно ведутся исследования по инкапсуляции TCP/IP в эти протоколы. На сегодняшний день многие проблемы решены и существует оборудование для организации TCP/IP сетей через эти системы.

WEB-сервисы

WEB-сервисы представляют собой специального типа WEB-приложения, не имеющие пользовательского интерфейса. Однако благодаря наличию WEB-методов могут в реальном времени предоставлять информацию о… да о чем угодно ! Будь то прогноз погоды или курс валют, информация о наличии свободных мест на утренний сеанс в Вашем любимом кинотеатре. Одним словом - WEB-сервис предоставляет услуги другим приложениям, причем последние могут быть любого типа, как WEB-

приложениями так и обычными приложениями с графическим интерфейсом. WEB-сервис имеет следующие отличительные особенности:

Выполняется на стороне сервера

Предоставляет набор методов, доступных внешним клиентам.

Исполняет WEB- методы и возвращает результаты клиентам

WEB-сервис и его клиенты могут быть написаны на разных языках и/или разных платформах.

На этом позволим себе временно отстраниться от теории и перейти к практике

Простейший WEB-сервис

Давайте запустим Delphi 8 и создадим WEB-сервис, который назовем SampleWebService

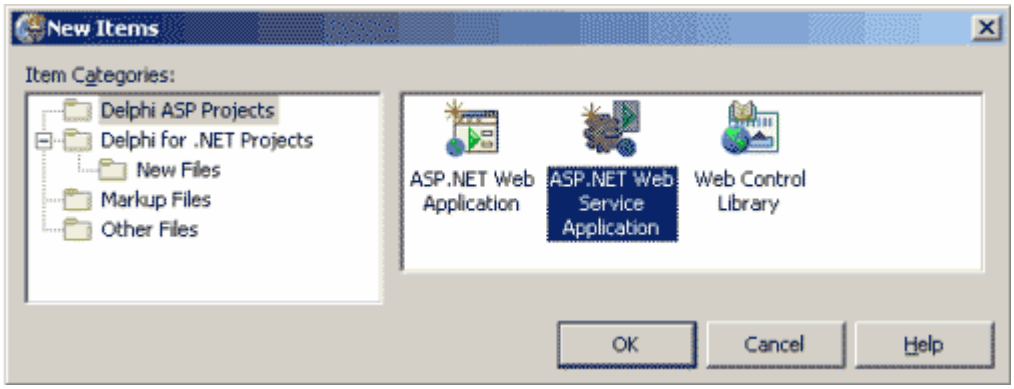


Рис.7 Выбор типа создаваемого приложения

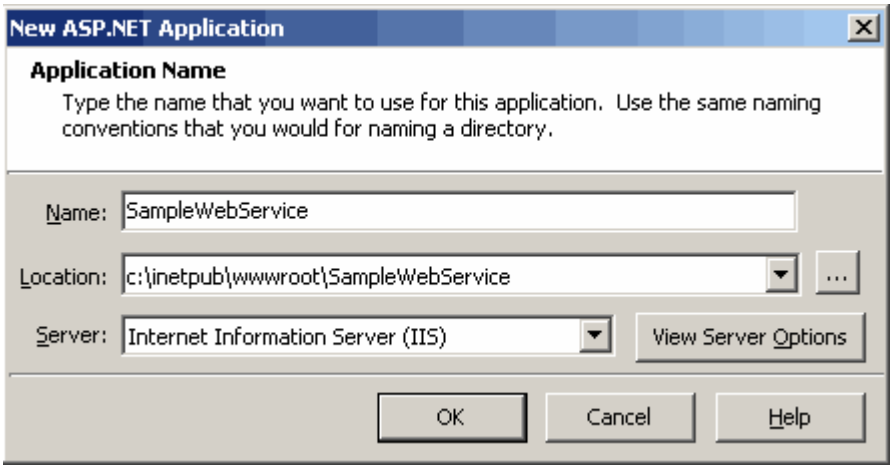


Рис.8 Диалог создания проекта.

Delphi 8 создаст для нас простейший WEB-сервис. Состав файлов в проекте WEB-сервиса требует отдельного описания, которое будет дано немного позже. Сейчас же рассмотрим файл WebService1.pas, который содержит описание класса TWebService1

```
TWebService1 = class(System.Web.Services.WebService)
  {$REGION 'Designer Managed Code'}
  strict private
    /// <summary>
```

```
  /// Required designer variable.
  /// </summary>
  components: IContainer;
  /// <summary>
  /// Required method for Designer support - do not
  /// modify the contents of this method with
  /// the code editor.
  /// </summary>
  procedure InitializeComponent;
{$ENDREGION}
strict protected
  /// <summary>
  /// Clean up any resources being used.
  /// </summary>
  procedure Dispose(disposing: boolean); override;
private
  { Private Declarations }
public
  constructor Create;
  (*
  // Sample Web Service Method
  [WebMethod]
  function HelloWorld: string;
  *)
end;
```

Обратите внимание на закомментированный метод WEB-метод HelloWorld, (WEB-метод он потому, что ему назначен атрибут [WebMethod]). Давайте попробуем раскоментировать его и его реализацию. Вот и все. Наш первый WEB-сервис готов. Как его протестировать? Очень просто, нажмите F9.

Результат не заставить себя долго ждать, вы увидите страницу подобную приведенной на рис. 9.

Рис 9. Автоматически сгенерированная страница-описание WEB-сервис

Как протестировать WEB-метод Вы наверное уже догадались? Если нет, то кликните по ссылке HelloWorld.

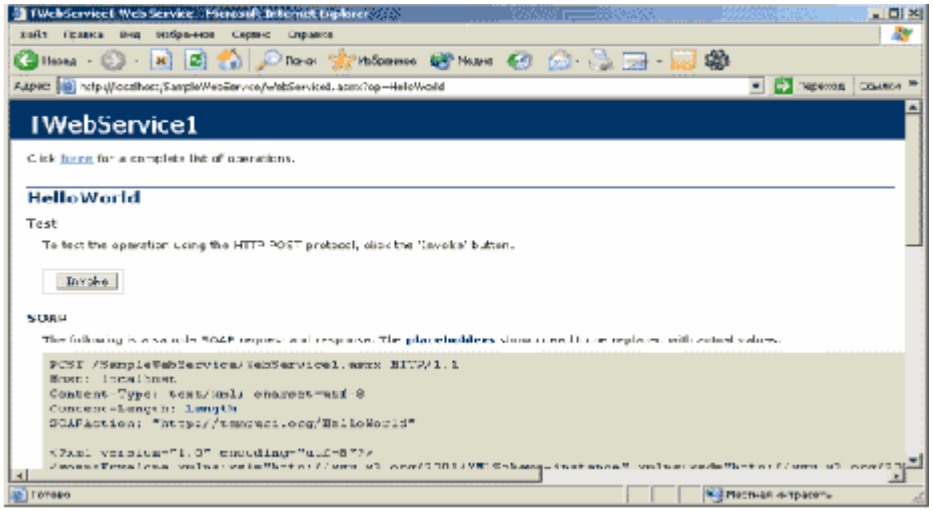


рис 10. Тестирование WEB-метода

После нажатия на кнопку "Invoke" наш WEB-сервис стартует и вернет потрясающий результат в виде XML:

```
<?xml version="1.0" encoding="utf-8" ?>
  <string xmlns="http://tempuri.org/">Hello World
```

Ну что ж, первой цели мы достигли: научились создавать простейший WEB-сервис, предоставляющий WEB-метод и все это успешно протестировано.

WEB-методы

Атрибут WebMethod

Как и было заявлено выше, обычный метод класса отличается от метода, публикуемого WEB-сервисом только наличием атрибута WebMethod. Данный атрибут имеет составной характер, т.е может содержать следующие податрибуты (Рассмотрим лишь некоторые из них):

CacheDuration - Кэширование результатов работы метода на заданное количество секунд. (например, метод с такими атрибутами будет хранить результат своей работы в течении 15 секунд : [WebMethod(CacheDuration="15")]).

Description - Добавляет текстовое описание WEB-метода.

MessageName - Имя WEB-метода. Полезно, например, когда нужно опубликовать перегруженный метод класса.(наличие двух одноименных WEB-методов запрещено)

В качестве примера давайте добавим к нашему классу еще два метода и добавим описание к существующему методу HelloWorld:

```
TWebService1 = class(System.Web.Services.WebService)
  // Экономия  места
```

```
  public
    constructor Create;
    // Sample Web Service Method

    [WebMethod
    (MessageName = 'HelloWorld' , Description =
                'Простой метод')]
    function HelloWorld:String;

   [WebMethod (MessageName = 'IntegerSubstract')]
     function Substract(a,b:Integer):Integer;overload;

    [WebMethod (MessageName = 'FloatSubstract')]
    function Substract(a,b:Single):Single;overload;
```

Реализация методов тривиальна:

```
function TWebService1.Substract(a,b:Integer):Integer;
begin
 Result := a - b;
end;

function TWebService1.Substract(a,b:Single):Single;
begin
 Result:= a - b;
end;
```

Запустите добавленные WEB-методы. Обратите внимание, что мы использовали механизм перегрузки функций, но при этом не пострадали от ограничений связанных с именованием WEB-методов: WEB-сервис предоставляет их как методы с различными именами.

Сложные типы данных в WEB-методах

Все то, что было показано до этого, выглядело неплохо. Однако на практике не очень часто приходится оперировать простыми типами данных, как это было показано в предыдущих примерах. Очень часто возникает необходимость вернуть сложный тип данных (например, объект).

На самом деле решение проблемы не представляет особых сложностей. Давайте попробуем ее решить. Итак, пусть нам необходимо создать сервис, возвращающий курс доллара за указанный промежуток времени.

Итак, курс доллара будет представлен следующим классом:

```
TDollarRate = class
  public
    Cost:Integer;
    Date:TDateTime;
```

```
  constructor Create;
 end;

constructor TDollarRate.Create;
begin
 inherited Create;
 Cost:=20 + Random(5);
 Date:=DateToStr(DateTime.Now);
end;
```

Перед добавлением WEB-метода объявим тип TDollarRates = Array of TDollarRate, в секцию uses добавим Borland.Vcl.SysUtils. Метод имеет вид:

```
[WebMethod]
  function GetRatesForDays (ADays:Integer):TDollarRates;

  function TWebService1.GetRatesForDays
        (ADays:Integer):TDollarRates;
var
  i:Integer;
begin
  SetLength(Result,ADays);
  for i:=ADays-1 downto 0  do
  Result[i]:=TDollarRate.Create;
end;
```

Попробуем протестировать метод (рис 11).

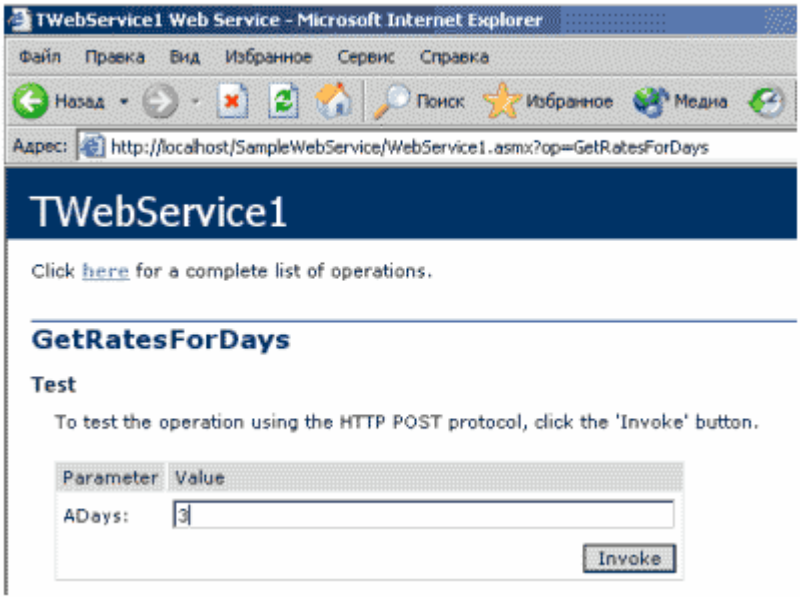


Рис. 11 Тестирование метода, возвращающего массив объектов Результат превзошел все ожидания:

```
<?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfTDollarRate
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://tempuri.org/">
- <TDollarRate>
  <Cost>23
  <Date>28.04.2004
  </TDollarRate>
- <TDollarRate>
  <Cost>23
  <Date>28.04.2004
  </TDollarRate>
- <TDollarRate>
  <Cost>20
  <Date>28.04.2004
  </TDollarRate>
  </ArrayOfTDollarRate>
```

В процессе разработки этого примера мы были неприятно удивлены одной деталью (версия Delphi 8 7.1.1146.610): мы попытались объявить новый конструктор с параметрами:

```
TDollarRate = class
  public
    Cost:Integer;
    Date:TDateTime;
   constructor Create(Adays:Integer);
  end;
constructor TDollarRate.Create(Adays:Integer);
var sDate:TDateTime;
begin
 inherited Create;
 {Код}
end;
```

и получили следующую ошибку при старте WEB-сервиса:

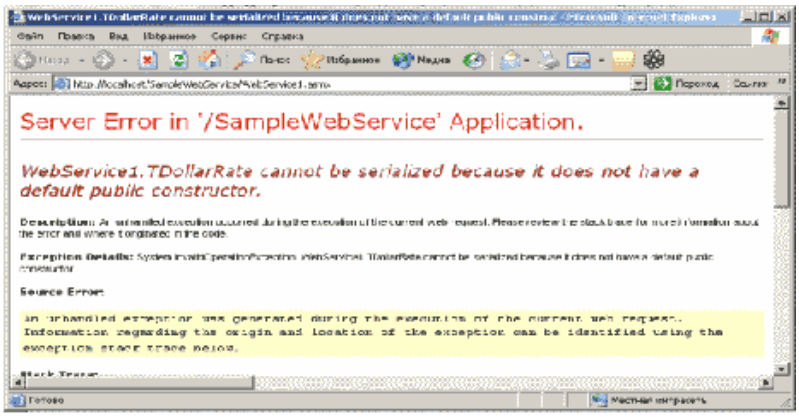


рис 12. Как же переопределить конструктор ?

Как сделать новый конструктор Default public в Delphi 8 не совсем понятно, однако выручило переименование конструктора следующим образом:

```
TDollarRate = class
 public
  Cost:Integer;
  Date:String;
 constructor TDollarRate(Adays:Integer);
 end;
```

Результат работы стал таким:

```
"?xml version="1.0" encoding="utf-8" ?>
- <ArrayOfTDollarRate
   xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns="http://tempuri.org/">
- <TDollarRate>
  <Cost>21
  <Date>26.04.2004
  </TDollarRate>
- <TDollarRate>
  <Cost>24
  <Date>26.04.2004
  </TDollarRate>
  </ArrayOfTDollarRate>
```

На этом описание WEB-методов завершается. Перед тем, как рассказать о том, каким образом клиентское приложение может взаимодействовать с нашим WEB-сервисом, а также каким образом оно будет "понимать" не только простые, но и "сложные" типы данных рассмотрим подробнее, из каких частей состоит WEB-сервис.

Архитектура WEB-сервиса

По большому счету WEB-сервис представляется всего одним файлом, с расширением Asmx, который должен как минимум иметь примерно такой заголовок:

```
<%@ WebService Language="c#"
   Class="WebService1.TWebService1" %>
```

Далее может идти код, собственно реализующий функциональность WEB-сервиса. Этот код должен быть написан на одном из языков .NET платформы (например C#).

К великому сожалению, создать WEB-сервис на Object-Pascal таким образом пока нельзя. Однако разработчики платформы .NET предусмотрели возможность перенести код WEB-сервиса в отдельно компилируемую DLL(фоновый код). Частично для того, чтобы была возможность разрабатывать WEB-приложения на

языках, непосредственно не поддерживающих ASP.NET, частично для того, чтобы диагностировать ошибки компиляции до развертывания самого сервиса.

Как вы уже догадались, Delphi 8 создает проект, компилируемый в DLL(которая, в свою очередь, помещается в корневой каталог приложения) и состоящий из таких частей:

Автоматически сгенерированный файл <Имя сервиса>.asmx, состоящий из заголовка примерно такого вида:

```
<%@ WebService Language="c#" Debug="true"
        Codebehind="WebService1.pas"
            Class="WebService1.TWebService1" %>
```

<Имя сервиса>.pas с которым мы успешно работали :-)

Global.asax, и его Pascal-реализация. Для чего он нужен, можно почитать в MSDN.

Как вы наверняка уже догадались для тестирования сервиса достаточно в браузере набрать строку

http://localhost/<путь к сервису>/<имя сервиса>.asmx

Для вызова метода

http://localhost/<путь к сервису>/

<имя сервиса>.asmx /? Op= <имя операции>.

WSDL - язык описания WEB-сервисов.

Мы практически готовы к тому, чтобы перейти к созданию клиента для нашего WEB-сервиса. Нам осталось только узнать как сторонние разработчики (пользователи нашего сервиса) могут узнать какие методы поддерживает WEB-сервис, сигнатуры этим методов, URL сервиса, типы используемых данных. Вся эта информация описывается при помощи языка WSDL. Тем не менее, вам не придется его изучать, так как этот язык больше для компьютеров, не для людей. Как же получить описание нашего WEB-сервиса на языке WSDL? Да очень просто, достаточно ввести в браузере

http://localhost/<путь к сервису>/<имя сервиса>.asmx?wsdl

Ниже приведено описание TDollarRates и TDollarRate нашего примера:

```
- <s:complexType name="ArrayOfTDollarRate">
- <s:sequence>
  <s:element minOccurs="0" maxOccurs="unbounded"
          name="TDollarRate"
          nillable="true" type="s0:TDollarRate" />
  </s:sequence>
  </s:complexType>
- <s:complexType name="TDollarRate">
- <s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="Cost"
          type="s:int" />
  <s:element minOccurs="0" maxOccurs="1" name="Date"
          type="s:string" />
  </s:sequence>
  </s:complexType>
```

Создание клиента для WEB-сервиса.

После стольких усилий по изучению WEB-сервисов пришло время научится их использовать. Как и всегда ничего сложного в этом нет. В качестве примера создадим VCL Forms приложение. Его главная и единственная форма должна выглядеть примерно так:

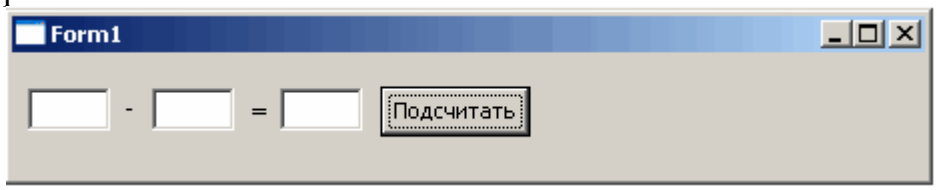


Рис. 13. Форма Веб-Калькулятора

Осталось только "оживить" нашу форму. Для этого выберите пункт меню Project/Web Reference.

В диалоге, который откроется, укажите URL к WSDL описанию нашего сервиса В нашем случае это -

http://localhost/SampleWebService/WebService1.asmx?WSDL

Нажмите кнопку "GO" а потом "AddReference".

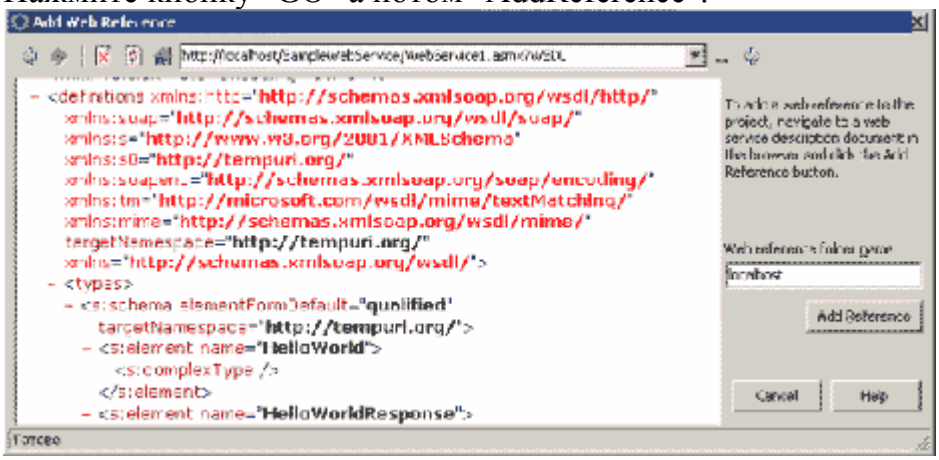


Рис. 8. Добавление ссылки на WEB-сервис.

InternetExpress - это входящее в состав Borland Delphi 5 интересное средство обработки и публикации данных в интернет, основанное на технологии MIDAS. В Delphi имеется набор компонентов, позволяющих реализовать полный цикл клиент-серверной обработки данных на базе интернет с применением как средств создания приложений на основе ISAPI/NSAPI, ASP и CGI, так и новых технологий, к примеру, стандарта XML.

В InternetExpress используются средства поддержки XML из MIDAS 3. Поскольку в настоящее время не все интернет-браузеры поддерживают представление данных по стандарту XML, в InternetExpress реализована специальная технология поддержки XML на основе JavaScript и DHTML, позволяющая использовать InternetExpress браузерами, не поддерживающими XML. Кроме того, если приложение InternetExpress работает с IE 5, то порождаемый им XML-пакет будет специальным образом оптимизироваться. В браузерах без такой поддержки пакеты данных XML разбираются с использованием специального модуля JavaScript (xmldom.js), который реализует спецификацию DOM (Document Object Model).

Объектная модель XML-документов представляет его внутреннюю структуру в виде совокупности определенных объектов. Для удобства эти объекты организуются в древообразную структуру данных: каждый элемент документа может быть отнесен к отдельной ветви, а все его содержимое - в виде набора вложенных элементов, комментариев, секций CDATA и т. д. представляется в этой структуре поддеревьями. Поскольку в любом правильно составленном XML-документе обязательно есть главный элемент, то все его содержимое можно представит в виде поддеревьев этого основного элемента, называемого в данном случае корнем дерева документа.

Объектное представление структуры документа не является для разработчиков чем-то новым. Объектно-ориентированный подход давно используется в сценариях для доступа к содержимому HTML-страницы. Доступные для JavaScript или VBScript элементы веб-страницы и раньше можно было создавать, просматривать и редактировать при помощи соответствующих объектов. Но их список и набор методов постоянно изменяется и зависит от типа браузера и версии языка. Для того, чтобы обеспечить интерфейс доступа к содержимому структурированного документа, не зависящий от языка программирования и типа документа, консорциумом W3 была разработана и официально утверждена спецификация объектной модели DOM Level 1.

DOM - это спецификация универсального платформо- и программно-независимого доступа к содержимому документов. Она является просто своеобразным API для их обработчиков. DOM служит стандартным способом построения объектной модели любого документа HTML или XML, при помощи которой можно производить поиск нужных фрагментов, создавать, удалять и модифицировать отдельные элементы.

Для описания интерфейсов доступа к содержимому XML-документов в спецификации DOM применяется платформонезависимый язык IDL (Interface Definition Language). Для использования этих интерфейсов их необходимо “перевести” на какой-то конкретный язык программирования. Однако этим занимаются создатели самих анализаторов. Нам можно ничего не знать о способе реализации интерфейсов -- с точки зрения разработчиков прикладных программ DOM выглядит как набор объектов с определенными методами и свойствами.

InternetExpress

На вкладке панели компонентов InternetExpress расположены две пиктограммы, соответствующие двум компонентам базового набора: TXMLBroker и TMIDASPageProducer.

Первая из них “отвечает” за формирование XML-пакета. Также в ее функции входят реакция на изменение данных и оповещение о действиях, выполняемых клиентом. TMIDASPageProducer “отвечает” за формирование сборного DHTML-документа. Последний, собственно, и является клиентским приложением, поскольку содержит все те визуальные элементы, соответствующие структуре пакета данных XML. В этот документ передаются XML-пакеты, формируемые компонентом XMLBroker.

В тот момент, когда от клиентского приложения на сервер приложений приходит сообщение о необходимости изменить информацию, TMIDASPageProducer

опрашивает все элементы управления HTML, формирует пакет с данными, подлежащими обновлению, и передает их на сервер приложений. Таким образом, обработка данных на клиенте происходит с использованием средств HTML, а передача структурированных данных к клиенту и изменений от него -- при помощи пакетов данных XML.

Эти компоненты помещаются в веб-модуль (WebModule) серверного приложения, для создания которого может быть использован специальный мастер (для этого нужно выбрать команду File -> New, а затем щелкнуть на пиктограмме Web Server Application).

WebModule является наследником TDataModule. По сравнению с “родителем” он обладает некоторыми дополнительными возможностями, которые позволяют обмениваться данными с веб-клиентами. Кроме базового набора InternetExpress, есть еще несколько компонентов, таких как TReconcilePageProducer, которые устанавливаются из дополнительных пакетов, входящих в комплект поставки Delphi. Естественно, существует возможность наследования базовых классов и создания на их основе собственных компонентов с расширенными возможностями.

Компонент TMIDASPageProducer отвечает за сборку HTML-документа, отображающего “живой” набор данных, получаемый от сервера приложений, или же от “типового” HTML-документа, вообще не обрабатывающего данные. Компонент может быть использован для создания веб-приложений на основе MIDAS, которые будут отображать информацию, получаемую из БД через сервер приложений, и передавать ее HTML-клиентам в пакетах XML-данных. При создании веб-модуля в соответствующих элементах TWebActionItem должна быть поставлена ссылка на один из таких компонентов (свойство Producer).

TMIDASPageProducer создает HTML-документ на основе шаблона. В отличие от других компонентов типа Producer, он имеет стандартный (default) шаблон, в котором содержатся несколько описаний верхнего уровня, на основе которых в других компонентах порождаются HTML-документы. Помимо шаблонов, содержание конечного документа может определяться данными, порождаемыми другими компонентами веб-модуля, другим компонентом TMIDASPageProducer через свойство TMIDASPageProducer.Content и т. д.

Связывание HTML-элементов с пакетами данных XML и обработчиками событий HTTP в TMIDASPageProducer осуществляется исключительно по именам HTML-объектов и соответствующих событий. Это позволяет редактировать сгенерированный HTML-шаблон в любом редакторе (специализированным для работы с HTML или нет), придавая ему необходимый внешний вид и дополняя логику обработки данных вставками JavaScript. Даже если свойства объектов, порожденных встроенным редактором TMIDASPageProducer, будут изменены другими средствами, эти изменения не будут потеряны, поскольку будут включены в шаблон.

Расширение функциональности обработчика шаблонов (свойство TMIDASPageProducer.HTMLdoc) возможно за счет реализации обработчика события TMIDASPageProducer.OnHTMLtag или перекрытия метода TMIDASPageProducer.DoTagEvent. Реализовав собственную версию обработчика этого события, программист получает возможность использовать в теле шаблона документа собственные тэги, заменяя их на этапе генерации HTML-документа

соответствующими значениями. Пример такого подхода показан в демонстрационном приложении InetXCenter из состава Delphi 5 (модуль InetXCenterProd.pas).

Конечно, возможности InternetExpress можно расширять практически неограниченно, разрабатывая специальные компоненты-наследники класса TMIDASPageProducer и компонентов, используемых для формирования содержимого документа (TDataForm, TQueryForm и др.). Создавая на их основе специализированные компоненты, можно максимально упростить создание конечного решения на основе InternetExpress, реализуя специфические возможности, необходимые тому или иному интернет-приложению. Например, в демонстрационном приложении InetXCenter создание наследника компонента TMIDASPageProducer позволило реализовать такие возможности, как генерация полей заголовка HTML-документа , комментарии и описания, автоматически подставляемые в конечный HTML-документ, и другие расширения базового компонента.

Формирование структуры HTML-документа

Поскольку TMIDASPageProducer (TCustomMIDASPageProducer) является генератором содержания HTML-документа, в его описание входит интерфейс IWebContent, который, собственно, это содержание и предоставляет. Заголовок соответствующего класса выглядит следующим образом:

TCustomMIDASPageProducer = class(TPageItemProducer, IWebContent, IWebComponentEditor, IScriptEditor)

Помимо IWebContent, в описании класса участвуют еще два интерфейса: IWebComponentEditor и IScriptEditor, которые являются средствами связи с design-time редактором для компонентов типа TWebComponent и HTML-кода. НижДалее приведено краткое описание ключевых свойств TMidasPageProducer.

HTMLDoc. Базовый шаблон, содержащий включения (includes) описателей содержания.

HTMLFile. То же, что и HTMLDoc, но с привязкой к файлу.

IncludePathURL. Путь к библиотекам JavaScript (в формате URL). Может быть полным (http://someserver/iexpress/) или относительным (/iexpress/).

Styles. Описание стандартных стилей для генерации HTML-документа. Аналог файла стилей, используемого при создании канонических веб-страниц.

StylesFile. То же, что и Styles, но с привязкой к файлу стилей.

WebPageItems. Список специальных компонентов, определяющих ключевые элементы документа. Основные типы PageItem: DataForm, QueryForm и LayoutGroup. Каждый из базовых компонентов TWebPageItem может иметь вложенные компоненты. Например, для DataForm это могут быть DataGrid, DataNavigator и т. д.

Компоненты TDataForm, TQueryForm и т. д. определяют структуру и основные параметры отображения HTML-документа, а стили определяются свойствами Styles и HTMLDoc.

Следует отметить, что благодаря тому, что состав HTML-документа определяется стандартными компонентами, поставляемыми в исходных текстах, функциональные возможности InternetExpress становятся практически неограниченными – за счет создания специализированных наборов компонентов для построения интернет-приложений. Примеры подобного подхода есть в демонстрационном приложении InetXCenter из Delphi 5.

Невизуальный компонент TWebActionItem позволяет задать реакцию интернет-приложения на те или иные события, транслируемые протоколом HTTP от веб-клиента. Предоставляя специальные свойства для задания ссылок на компоненты TMIDASPageProducer и TPageProducer, а также URL, TWebActionItem позволяет описать алгоритм перемещения между HTML-документами, составляющими интернет-приложение, реагировать на передачу параметров и значений полей HTML-документа определенным образом и т. д. Создавая обработчик события TWebActionItem.OnAction, программист получает возможность возвращать необходимые данные в полях запросов, устанавливать идентификационные маркеры (cookies) для веб-клиентов, контролировать генерацию содержания HTML-документов и выполнять ряд других операций практически на самом нижнем уровне функционирования интернет-приложения. Далее описаны основные свойства компонента TWebActionItem.

Default. Использование данного компонента как обработчика событий соответствующих типов (свойство MethodType) в тех случаях, когда явно не задан иной обработчик. Из всех компонентов TWebActionItem, присутствующих в контейнере TWebModule, только у одного свойство Default может иметь значение True.

DisplayName. Отображение текущего компонента в списке компонента TCustomWebDispatcher. Должно быть уникальным для своего контекста.

Enabled. Как и во многих других компонентах, означает разрешение или запрет выполнения связанных с компонентом действий. Если его значение равно False, то соответствующий компонент типа PageProducer не будет генерировать содержимое HTML-документа.

MethodType. Определяет HTTP-метод, при вызове которого со стороны веб-клиента будет использован данный компонент. По умолчанию имеет значение mtAny, то есть все доступные методы, но может принимать значения отдельных типов, например mtGet (запрос на получение веб-клиентом содержимого документа);

PathInfo. Задает путь к получателю всех сообщений, объекту типа TWebActionItem, в формате URI (Unified Resource Identifier). Позволяет перенаправить очередь сообщений другому компоненту PageProducer или HTML-документу;

Producer. Ссылка на компонент PageProducer. Если компонент явно не задан, то для реакции на сообщение обязательно нужно создать обработчик OnAction. Если ссылка на PageProducer актуальна (не nil), сообщение обрабатывается или PageProducer, или обработчиком события OnAction (если он есть).

Примеры использования свойств TWebActionItem есть в демонстрационном приложении InetXCenter (модуль InextXCenterModule.pas).

Поговорим теперь о невизуальных компонентах категории PageItems, предназначенных для формирования структуры HTML-документа. Как и компоненты VCL, они делятся на средства отображения типовых элементов HTML-документа и элементов для обработки данных, получаемых от сервера приложений. У каждого из этих компонентов могут быть наследники, расширяющие их свойства или реализующие те элементы HTML, эквивалента которым нет в текущей версии InternetExpress. Компоненты PageItems находятся в модуле miditems.pas. При

построении HTML-документа они объединяются в иерархические структуры. Например, компонент TDataNavigator содержит компоненты типа TDataSetButton.

При создании HTML-документа компонентом TMIDASPageProducer эти компоненты генерируют фрагменты HTML-кода, описывающего соответствующие HTML-элементы. Компонент TMIDASPageProducer объединяет их в единый поток и подставляет вместо соответствующих тэгов в шаблоне документа. К элементам HTML “привязываются” обработчики событий, написанные на JavaScript и являющиеся аналогами обработчиков событий для визуальных компонентов Delphi, таких как OnClick. Отдельные компоненты PageItems позволяют напрямую задать получателя сообщений (target) в формате URI (свойство Action), что дает возможность переходить от одного HTML-документа к другому и передавать между этими документами параметры в формате протокола HTTP.

Благодаря использованию в TMIDASPageProducer шаблонов для генерации HTML-документов появляется возможность создавать визуальные и невизуальные элементы HTML-документа прямым редактированием. Используя обработчики HTTP-событий, можно связывать такие элементы с генерируемыми шаблоном через компоненты TWebActionItem или при помощи создаваемых опять-таки прямым редактированием обработчиков на JavaScript внутри HTML-документа.

Компонент TXMLBroker передает пакеты данных в формате XML от сервера приложений к HTML-клиенту, получает от HTML-клиента измененные данные, расшифровывает разностные пакеты данных XML и передает сведения об изменении данных на сервер приложений. Этот компонент находится в модуле xmlbrokr.pas.

Компонент TXMLBroker можно использовать в приложении, которое одновременно является и MIDAS-клиентом, и серверным веб-приложением. Серверы такого класса, как правило, имеют две основные функции:

получать от сервера приложений через интерфейс IAppServer пакеты XML-данных;

обрабатывать поступающие от браузеров HTTP-сообщения, содержащие пакеты XML-данных с изменениями исходного набора и передавать их серверу приложений.

Для того чтобы сделать информацию, содержащуюся в базе данных, доступной в формате XML, достаточно добавить компонент TXMLBroker в контейнер WebModule совместно с компонентом TMIDASPageProducer, который будет использовать XML-пакеты данных для создания HTML-страниц.

TXMLBroker автоматически регистрируется в веб-модуле (или веб-диспетчере) как автодиспетчеризуемый объект (auto-dispatching object). Это означает, что веб-модуль или веб-диспетчер будут перенаправлять все входящие HTTP-сообщения прямо к нему. Все входящие сообщения считаются данными для обновления, порождаемыми браузером в ответ на получение HTML-потока, порождаемого компонентом TApplyUpdatesButton. TXMLBroker автоматически передает пакет с XML-данными, содержащий сведения о различиях в данных, на сервер приложений и возвращает все ошибки, возникшие при обновлении данных, тому компоненту управления содержимым документа (TMIDASPageProducer), который имеет возможность генерации соответствующего ответного сообщения. Среди основных свойств компонента нужно выделить следующие.

AppServer. Интерфейс IAppServer для связи с провайдерами данных.

MaxErrors. Максимальное число ошибок, по достижении которого провайдер должен прекратить обновление данных.

MaxRecords. Управляет формированием пакетов данных XML. Значение -1 разрешает передачу всех записей из набора данных в XML-пакет; значение 0 разрешает передачу только метаданных; положительное значение определяет число записей (строк), которые могут быть переданы в XML-пакет.

Params. Список параметров, передаваемых серверу приложений. Используется, в частности, для передачи параметров хранимых процедур и SQL-запросов.

ProviderName. Имя провайдера данных.

ReconcileProducer. Ссылка на компонент TReconcilePageProducer, который будет использоваться для разрешения конфликтов данных при обновлении.

WebDispatch. Список типов сообщений протокола HTTP, на которые будет реагировать компонент. Как правило, эти сообщения порождаются при нажатии на HTML-странице кнопки типа TApplyUpdatesButton.

Серверная часть приложения InternetExpress

Серверная часть интернет-приложения, созданного на основе InternetExpress, состоит из исполняемого модуля, написанного в данном случае на Delphi 5 и включающего в себя WebModule, а также файлов-библиотек JavaScript, которые, если браузер не поддерживает XML, передаются клиенту. НижДалее перечислены эти библиотеки.

xmldom.js. XML-парзер, соответствующий стандарту DOM (Document Object Model), написанный на JavaScript. Позволяет браузерам без встроенной поддержки XML использовать пакеты данных этого стандарта. Для IE5 этот файл не передается, а XML-пакет оптимизируется специальными образом.

xmldb.js. Библиотека классов доступа к данным, обслуживающая пакеты данных XML.

xmldisp.js. Библиотека с описаниями связей между классами доступа к данным в xmldb.js и элементами HTML.

xmlerrdisp.js. Библиотека классов для обработки конфликтных ситуаций, возникающих при изменении данных. Использует пакет разности данных (XML delta packet) и пакет ошибок (XML error packet);

xmlshow.js. Библиотека функций для отображения окон с данными XML.

Для того чтобы передать эти библиотеки клиенту и использовать их там, достаточо включить в HTML-документ ссылки на них

Ссылка на xmldom.js требуется только в том случае, если браузер не имеет встроенной поддержки XML.

Клиентская часть приложения InternetExpress

Клиентская часть приложения на основе InternetExpress представляет собой собственно HTML-документ, порожденный одним или несколькими компонентами TMIDASPageProducer (или их наследниками), интерпретируемый тем или иным браузером. Как уже говорилось, этот документ может содержать элементы отображения и управления, соответствующие структуре пакета данных XML. К ним также могут добавляться элементы управления, формирующие HTML-аналог DBnavigator из состава Delphi VCL, если соответствующие параметры были заданы при настройке PageProducer, а также другие элементы управления HTML, как

связанные с обработкой данных, так и образующие независимые части интерфейса, например группу для ввода имени пользователя и пароля.

Приложение InternetExpress работает следующим образом. Браузер обращается по ссылке (URL) к серверному приложению InternetExpress. Оно возвращает HTML-документ, который можно рассматривать как некую отправную точку в процессе обработки.

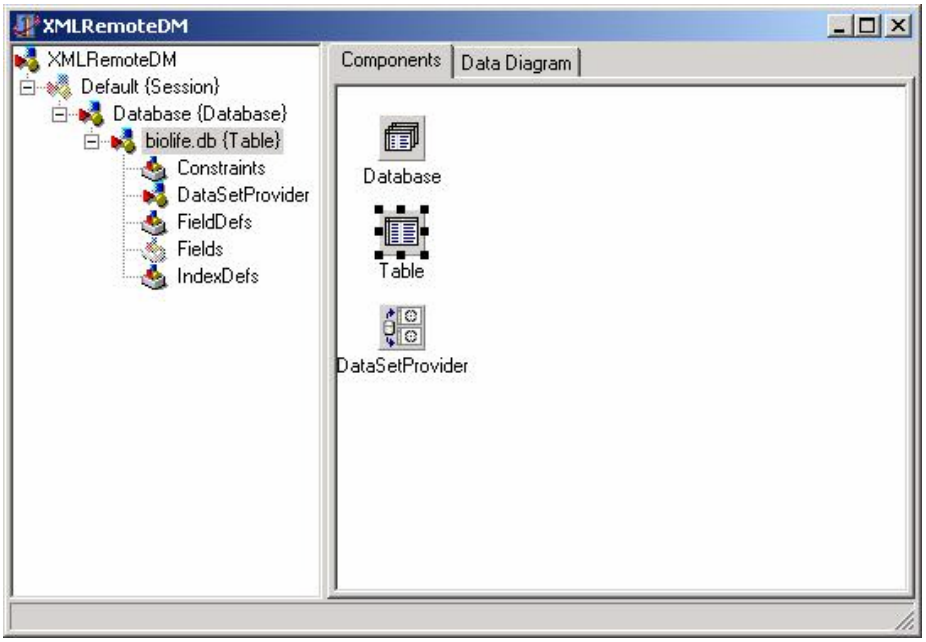
По запросу пользователя серверное приложение сначала возвращает очередной HTML-документ, содержащий (при необходимости) ссылки на библиотеки JavaScript, отвечающие за обработку XML-пакетов. Затем уже переданный пользователю документ посылает запрос серверной части приложения, которая возвращает клиенту данные в виде XML-пакетов, интерпретируемых соответствующими библиотеками JavaScript.

После того как пользователь просмотрит набор данных и, при необходимости, внесет в них изменения, он может передать изменения серверной части приложения. Этот процесс запускается событием, которое, как правило, связано с элементом управления -- кнопкой (например, Submit) и передается серверной части приложения InternetExpress, а именно компоненту TMIDASPageProducer. Все сведения об изменении данных передаются серверной части приложения в виде разностных пакетов XML (XML delta packets).

Серверная часть получает информацию об изменении данных и использует сервер приложений для внесения этих изменений в БД. В случае возникновения конфликта (reconcile error) имеется возможность сформировать HTML-вариант Reconcile Dialog из состава Delphi.

Создание веб-приложения на основе InternetExpress

Для создания веб-приложения необходим скомпилированный и зарегистрированный сервер данных. В данном примере используются данные из таблицы biolife.db, входящей в состав демонстрационной базы данных из комплекта Delphi 5. Данные публикуются через контейнер Remote Data Module.



Контейнер Remote Data Module

После создания и регистрации сервера данных необходимо создать клиента для этого сервера, который, в свою очередь, будет сервером для HTML-клиента. Для создания расширений веб-сервера в Delphi 5 есть специальный “мастер”. Он может быть вызван через меню File -> New -> Web Server Application.

В данном случае мы создаем CGI-приложение, выводящее порождаемый поток данных в устройство стандартного вывода (stdout). Поток данных этого приложения будет без изменений передаваться вызывающему документу через транспортный протокол.

Мастер автоматически создаст контейнер типа TWebModule, в который необходимо поместить компоненты TMIDASPageProducer и TXMLBroker. Сюда же мы поместим и компонент TDCOMConnection, который будем использовать для подключения к удаленному серверу данных, а также компонент TClientDataSet для доступа к удаленному модулю данных.

Определив необходимые для соединения с удаленным сервером свойства, переходим к созданию содержимого HTML-документа. Для этого необходимо назначить для объекта класса TXMLBroker свойства RemoteServer и ProviderName, а также создать хотя бы один компонент TWebActionItem, вызвав соответствующий редактор с помощью контекстного меню компонентов TXMLBroker и TMIDASPageProducer.

В примере используется только один такой компонент, свойству которого Default присвоено значение True, за счет чего все сообщения HTTP будут поступать на этот компонент, а от него -- на TXMLBroker и TMIDASPageProducer.

Далее необходимо вызвать редактор веб-страниц. Это можно сделать с помощью команды Web Page Editor контекстного меню компонента TMIDASPageProducer. Для работы этого элемента необходим Microsoft Internet Explorer 4.0 и выше.

После добавления необходимых элементов получаем готовое к применению приложение веб-сервера. При установке параметров отображения HTML-документа можно воспользоваться свойствами компонента DataGrid и других элементов HTML-документа для придания ему необходимого внешнего вида, а также вручную доработать HTML-код в соответствующем встроенном редакторе.

После компиляции исполняемый модуль (в нашем примере - XMLServerApp.exe) необходимо поместить в каталог веб-сервера, для которого выделены права на запуск приложений. В этот же каталог необходимо поместить библиотеки JavaScript. Для проверки правильности размещения библиотек можно воспользоваться специальным HTML-файлом scripttest.HTML, который находится в каталоге Demos\Midas\InternetExpress\TroubleShoot на компакт-диске Delphi 5 или в каталоге установки на жестком диске рабочей станции. Этот HTML-файл проверяет правильность размещения библиотек и настройки веб-сервера и в случае наличия тех или иных ошибок выдает некоторые рекомендации по разрешению проблем.

По окончании настройки можно обратиться к нашему приложению напрямую через протокол HTTP, поскольку оно порождает полноценный HTML-документ, не требующий дополнительной “обвязки”.

Рассматриваемое в рамках данной статьи демонстрационное приложение отнюдь не претендует на полноту и законченность. Для более полного ознакомления с возможностями InternetExpress я рекомендую обратиться к демонстрационным примерам из поставки Delphi 5 Enterprise, находящимеся в каталоге Runimage\Delphi50\Demos\Midas\InternetExpress на компакт-диске или в Demos\Midas\InternetExpress, расположенном в том каталоге на жестком диске, где находится Delphi 5. Внимательно прочитайте сопроводительные файлы к этим примерам, поскольку некоторые из них требуют специфических настроек Delphi и/или веб-сервера.

ActiveX

Технология ActiveX, рассматриваемая в данной статье, базируется на технологии Microsoft COM (Component Object Model - модель компонентных объектов), позволяющей создавать и использовать программные компоненты, предоставляющие различные сервисы другим приложениям, компонентам и операционной системе. COM представляет собой одну из реализаций концепции распределенных вычислений, базирующейся в общем случае на предоставлении возможности приложениям использовать для расширения своей функциональности готовые компоненты и объекты (иногда они называются сервисами). Технология COM позволяет использовать объектно-ориентированный подход не в рамках одного приложения, а в рамках операционной системы, но, в отличие от стандартных классов, определенных в исходном тексте и реализуемых как объекты в адресном пространстве одного процесса, эти компоненты могут в общем случае располагаться в адресных пространствах разных процессов и даже на разных компьютерах.

В настоящее время существуют три типа спецификаций COM, определенных Microsoft и включающих большое количество интерфейсов и функций:

OLE-документы - составные документы, содержащие внедренные или связанные объекты. Эта спецификация описывает правила создания контейнеров для таких документов с "активацией по месту". Отметим, что компонент OLEContainer Delphi и C++Builder создан с учетом этой спецификации (этой теме будет посвящена одна из следующих статей данного цикла).

OLE Automation. Эта спецификация описывает, как создать сервер и контроллер, управляющий его поведением с помощью скриптов или макросов. Эта спецификация также поддерживается Delphi и C++Builder (об этом также пойдет речь в ближайших статьях данного цикла).

Управляющие элементы ActiveX, использующие специальный вариант протокола Automation (о них-то и пойдет речь в данной статье).

Использование COM, и, в частности, технологии ActiveX, позволяет обеспечить создание приложений, собираемых из готовых компонентов - элементов управления ActiveX, отличающееся от привычной пользователям C++Builder или Delphi разработки приложений с помощью VCL-компонентов тем, что такая "сборка" не зависит от того, на каком языке написаны как готовые компоненты, так и использующее их приложение - лишь бы средство разработки поддерживало

возможность использования таких компонентов в разрабатываемом приложении (такое приложение обычно называется контейнером).

Элементы управления ActiveX представляют собой библиотеки, содержащие исполняемый код. Как было сказано выше, эти библиотеки могут быть использованы в различных приложениях как встроенные элементы управления, поэтому они обладают свойствами, событиями и методами, доступными посредством автоматизации. Современные средства разработки, как правило, позволяют включать такие элементы в создаваемые с их помощью приложения. Помимо этого, элементы управления ActiveX нередко используются в качестве расширений web-броузеров с целью придания им дополнительной функциональности, например, для отображения документов, отличных от поддерживаемых данным броузером.

Как любой COM-сервер, элемент управления ActiveX обладает уникальным идентификатором GUID и должен быть зарегистрирован в реестре. На основании этой записи может быть осуществлен поиск местоположения файла с расширением *.ocx, содержащего его реализацию.

Таким образом, создав элемент управления ActiveX, обладающий интересующей Вас функциональностью, Вы можете в дальнейшем позволить его пользователям встраивать этот элемент в свои приложения (например, написанные на Visual Basic, PowerBuilder, Delphi, C++Builder и др.), отображать его в web-броузере в составе выгруженной с Вашего web-сервера HTML-страницы, включать его в состав документов MS Office, при этом Вы не обязаны предоставлять исходный текст этого компонента.

Когда следует создавать управляющие элементы ActiveX? Естественно, в тех случаях, когда функциональность, содержащаяся в таком элементе, уникальна. Нет смысла создавать ActiveX, реализующий функциональность кнопки или текстового редактора - таких элементов управления, в том числе выполненных в виде ActiveX, на рынке готовых компонентов более чем достаточно. Нет смысла также создавать ActiveX, если он будет использоваться только в C++Builder - в этом случае проще создать VCL-компонент, который будет работать в использующем его приложении значительно быстрее, чем аналогичный ActiveX. Но создание элемента управления, реализующего, к примеру, часть автоматизированного рабочего места какой-либо категории сотрудников Вашего предприятия может в ряде случаев оказаться весьма полезным, особенно в следующих двух случаях. Первый случай - использование на предприятии различных средств разработки, например, C++Builder и Visual Basic; в этом случае разработчик, использующий Visual Basic, может встраивать в свои приложения ActiveX, созданный другим разработчиком и реализующий какую-либо функциональность, используемую несколькими различными автоматизированными рабочими местами. Второй случай - широкое использование Internet или intranet при автоматизации предприятия. В этом случае ActiveX, реализующий подобную функциональность, может быть встроен в HTML-страницу и отображен в web-броузере. Такой подход существенно облегчает решение проблемы обновления версий автоматизированных рабочих мест, так как вместо установки новых версий на рабочих станциях достаточно заменить один экземпляр ActiveX, хранящийся на web-сервере. Наиболее ярким примером такого подхода может быть выполненный в виде

ActiveX "тонкий" клиент, получающий данные от удаленного сервера приложений, являющегося, в свою очередь, клиентом серверной СУБД.

Такой набор преимуществ сделал эту технологию за последние два года весьма популярной, и именно поэтому многие современные средства разработки, такие, как Delphi или C++Builder, позволяют создавать элементы управления ActiveX. Эти средства обычно имеют встроенные механизмы поддержки спецификации ActiveX с помощью автоматической генерации соответствующего кода (хотя, конечно, не возбраняется писать подобный код вручную).

Спецификация ActiveX представляет собой набор правил (а именно описание стандартных интерфейсов), с помощью которых следует создавать такие элементы управления. Отметим, что текущая версия этой спецификации учитывает возможность использования в качестве контейнера web-броузеров и необходимость загрузки элементов ActiveX с удаленных web-серверов с их автоматической регистрацией.

Программирование сокетов

Для обеспечения сетевых коммуникаций используются сокеты. Сокет это конечная точка сетевых коммуникаций. Каждый использующийся сокет имеет тип и ассоциированный с ним процесс. Сокеты существуют внутри коммуникационных доменов. Домены это абстракции, которые подразумевают конкретную структуру адресации и множество протоколов, которое определяет различные типы сокетов внутри домена. Примерами коммуникационных доменов могут быть: UNIX домен, Internet домен, и т.д.

В Internet домене сокет - это комбинация IP адреса и номера порта, которая однозначно определяет отдельный сетевой процесс во всей глобальной сети Internet. Два сокета, один для хоста-получателя, другой для хоста-отправителя, определяют соединение для протоколов, ориентированных на установление связи, таких, как TCP.

Создание сокета

Для создания сокета используется системный вызов socket.

s = socket(domain, type, protocol);

Этот вызов основывается на информации о коммуникационном домене и типе сокета. Для использования особенностей Internet, значения параметров должны быть следующими:

communication domain -

AF_INET (Internet протоколы).

type of the socket -

SOCK_STREAM; Этот тип обеспечивает последовательный, надежный, ориентированный на установление двусторонней связи поток байтов.

Выше был упомянут сокет с типом stream. Краткое описание других типов сокетов приведено ниже:

Datagram socket -

поддерживает двусторонний поток данных. Не гарантируется, что этот поток будет последовательным, надежным, и что данные не будут дублироваться. Важной

характеристикой данного сокета является то, что границы записи данных предопределены.

Raw socket -

обеспечивает возможность пользовательского доступа к низлежащим коммуникационным протоколам, поддерживающим сокет-абстракции. Такие сокеты обычно являются датаграм- ориентированными.

Функция socket создает конечную точку для коммуникаций и возвращает файловый дескриптор, ссылающийся на сокет, или -1 в случае ошибки. Данный дескриптор используется в дальнейшем для установления связи.

Для создания сокета типа stream с протоколом TCP, обеспечивающим коммуникационную поддержку, вызов функции socket должен быть следующим:

```
s = socket(AF_INET, SOCK_STREAM, 0);
```

Привязка к локальным именам

Сокет создается без имени. Пока с сокетом не будет связано имя, удаленные процессы не имеют возможности ссылаться на него и, следовательно, на данном сокете не может быть получено никаких сообщений. Коммуникационные процессы используют для данных целей ассоциации. В Internet домене ассоциация складывается из локального и удаленного адреса и из локального и удаленного порта. В большинстве доменов ассоциация должна быть уникальной.

В Internet домене связывание сокета и имени может быть весьма сложным, но, к счастью, обычно нет необходимости специально привязывать адрес и номер порта к сокету, так как функции connect и send автоматически свяжут данный сокет с подходящим адресом, если это не было сделано до их вызова.

Для связывания сокета с адресом и номером порта используют системный вызов bind:

```
bind(s, name, namelen);
```

Привязываемое имя (name) это строка байт переменной длины, которая интерпретируется поддерживаемым протоколом. Интерпретация может различаться в различных коммуникационных доменах.

Установление связи

Со стороны клиента связь устанавливается с помощью стандартной функции connect:

```
error = connect(s, serveraddr, serveraddrlen);
```

которая инициирует установление связи на сокете, используя дескриптор сокета s и информацию из структуры serveraddr, имеющей тип sockaddr_in, которая содержит адрес сервера и номер порта на который надо установить связь. Если сокет не был связан с адресом, connect автоматически вызовет системную функцию bind.

Connect возвращает 0, если вызов прошел успешно. Возвращенная величина -1 указывает на то, что в процессе установления связи произошла некая ошибка. В случае успешного вызова функции процесс может работать с дескриптором сокета, используя функции read и write, и закрывать канал используя функцию close.

Со стороны сервера процесс установления связи сложнее. Когда сервер желает предложить один из своих сервисов, он связывает сокет с общеизвестным адресом, ассоциирующимся с данным сервисом, и пассивно слушает этот сокет. Для этих целей используется системный вызов listen:

error = listen(s, qlength);

где s это дескриптор сокета, а qlength это максимальное количество запросов на установление связи, которые могут стоять в очереди, ожидая обработки сервером; это количество может быть ограничено особенностями системы.

Когда сервер получает запрос от клиента и принимает решение об установлении связи, он создает новый сокет и связывает его с ассоциацией, эквивалентной 'слушающему сокету'. Для Internet домена это означает тот же самый номер порта. Для этой цели используется системный вызов accept:

newsock = accept(s, clientaddr, clientaddrlen);

Сокет, ассоциированный клиентом, и сокет, который был возвращен функцией accept, используются для установления связи между сервером и клиентом.

Процесс установления связи показан на рисунке 1.

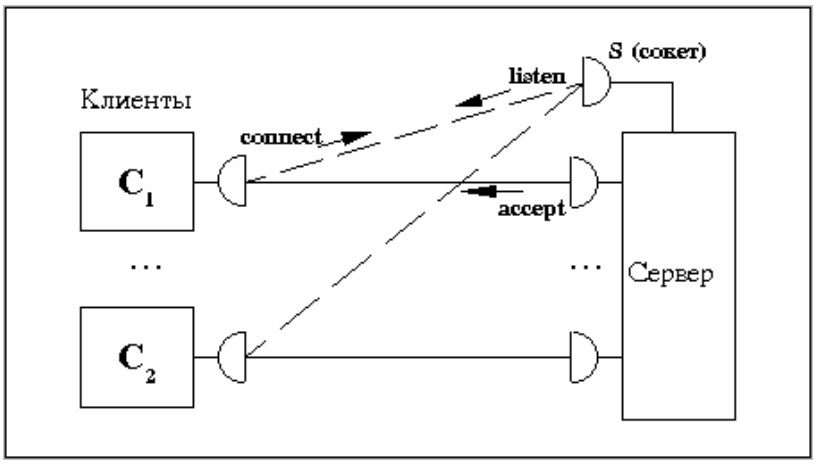


Рис. 1: Взаимодействие клиента и сервера

Передача данных

Когда связь установлена, с помощью различных функций может начаться процесс передачи данных. При наличии связи, пользователь может посылать и получать сообщения с помощью функций read и write:

write(s, buf, sizeof(buf)); read(s, buf, sizeof(buf));

Вызовы send и recv практически идентичны read и write, за исключением того, что добавляется аргумент флагов.

send(s, buf, sizeof(buf), flags); recv(s, buf, sizeof(buf), flags);

Могут быть указаны один или более флагов с помощью ненулевых значений, таких, как следующие:

MSG_OOB - Посылать/получать данные, характерные для сокетов типа stream.

MSG_PEEK - Просматривать данные без чтения. когда указывается в recv, любые присутствующие данные возвращаются пользователю, но сами данные остаются как "непрочитанные". Следующий read или recv вызванный на данном сокете вернет прочитанные в прошлый раз данные.

MSG_DONTROUTE - посылать данные без маршрутизации пакетов. (Используется только процессами, управляющими таблицами маршрутизации.)

Закрывание сокетов

Когда взаимодействующие модули решают прекратить передачу данных и закрыть сеанс связи, они обмениваются трехсторонним рукопожатием с сегментами, содержащими установленный бит "От отправителя больше нет данных" (этот бит еще называется FIN бит).

Если сокет больше не используется, процесс может закрыть его с помощью функции close, вызвав ее с соответствующим дескриптором сокета:

close(s);

Если данные были ассоциированы с сокетом, обещающим доставку (сокет типа stream), система будет пытаться осуществить передачу этих данных. Тем не менее, по истечении довольно таки длительного промежутка времени, если данные все еще не доставлены, они будут отброшены. Если пользовательский процесс желает прекратить любую передачу данных, он может сделать это с помощью вызова shutdown на данном сокете для его закрытия. Вызов shutdown вызывает "моментальное" отбрасывание всех стоящих в очереди данных. Формат вызова следующий:

shutdown(s, how);

где how имеет одно из следующих значений:

0 - если пользователь больше не желает читать данные

1 - если данные больше не будут посылаться

2 - если данные не будут ни посылаться ни получаться

На этой странице собраны компоненты, предназначенные для быстрой разработки приложений Интернет и интранет, использующих ту или иную службу, или сервис Интернет, или протокол (HTTP, UDP) и работающих по протоколу TCP/IP. Все эти компоненты перечислены в таблице 1.

Компоненты TNMURL и TNMUUProcessor выполняют утилитарные задачи: первый преобразует строку в формате URL в "обычную" строку символов и обратно; второй - кодирует и декодирует данные по стандарту MIME и UUEncode. Эти компоненты выполняют вспомогательную роль в приложениях.

В конце панели FastNet представлены два компонента: TPowersock и TNMGeneralServer. Первый компонент инкапсулирует API сокетов Windows и является базовым классом для большинства компонентов страницы FastNet за исключением компонентов TNMUDP, TNMURL и TNMUUProcessor. Его можно использовать для создания элементов управления, использующих другие известные протоколы, или для создания своих собственных протоколов.

Компонент TNMGeneralServer используется для создания серверов TCP/IP общего назначения в качестве базового класса для многопотоковых серверов Интернет или интранет.

Для использования этих компонентов необходим 32-х битный стек протоколов TCP/IP (библиотека WSOCK32.DLL), которая включена в операционные системы Windows 95, 98, 2000, Windows NT и Windows XP.

Компонент TStatusBar - строка состояния, часто используется в различных программах. Если во время разработки программы присваивайте свойству SimplePanel этого компонента значение True. В этом случае в строке состояния можно будет выводить только одну строку, которая должна помещаться в свойство SimpleText. Если свойству SimplePanel присвоить значение False, то в строке состояния можно завести несколько панелей сообщений и в каждую выводить свой текст. Для этой цели используется свойство Panels. При его выборе загружается редактор свойств коллекции объектов, в котором можно создать нужное количество панелей и настроить их внешний вид.

Поскольку компонент TPowersock является базовым для большинства компонентов панели FastNet, то многие свойства, методы и события этих компонентов совпадают.

Из общих свойств всех компонентов можно отметить свойства HostPort и TimeOut. В свойстве Host указывается имя или IP-адрес удаленного компьютера, с которым устанавливается связь. В свойстве Port указывается номер порта протокола TCP/IP на удаленной машине, по которому с ней нужно связываться. В свойстве TimeOut указывается временной интервал в миллисекундах, в течение которого программа будет ожидать ответа от сокета удаленной машины. Если по истечении этого времени ответ не поступит, будет возбуждаться исключительная ситуация. Если этому свойству задать значение 0, то ответ будет ожидаться бесконечно долго, пока не будет получен. Но если он все-таки не придет, то приложение зависнет. Поэтому

лучше указывать ненулевое число, а по том подбирать его значение с учетом пропускной способности сети.

Среди общих методов нужно упомянуть методы Connect и Disconnect. Первый из них применяется для установления связи с удаленной машиной. Второй - для разрыва установленного соединения.

Среди общих свойств компонентов можно упомянуть свойства OnConnect, OnConnectionFailed, OnConnectionRequired, OnDisconnect, OnError, OnHostResolved, OnlnvalidHost и OnStatus.

События OnConnect и OnDisconnect происходят в результате успешного выполнения методов Connect и Disconnect. Если процесс установления связи заканчивается ошибкой, происходит событие OnConnectionFailed. Событие OnConnectionRequired происходит после попытки вызова одного из методов, требующих для своей работы подключения к удаленному компьютеру. В обработчике этого события можно выполнить подключение и вернуть через параметр Handled значение True. В этом случае будет повторена попытка вызова того метода, который сгенерировал это со бытие. Если окажется, что соединение не установлено, или если в параметре Handled находится значение False, то метод завершится аварийно и будет возбуждена исключительная ситуация. Событие OnError наступает когда происходит ошибка в работе сокетов Windows. Через параметр ErrNo обработчика этого события передается числовой код ошибки, а через второй параметр ErrMsg - строка сообщения об ошибке.

Событие OnHostResolved происходит в случае успешного преобразования имени компьютера в IP-адрес. Событие OnlnvalidHost наступает, если в свойстве Host задано неверное имя или IP-адрес удаленного компьютера. Событие OnStatus генерируется, когда меняется статус компонента.

Таблица 1. Элементы управления FastNet для Интернет Элемент Описание

TNMDayTime Получает дату и время от серверов Интернет даты и времени в соответствии со стандартом RFC 867.

TNMMsg Посылает простые текстовые сообщения в кодах ASCII по Интернет или интранет с использованием протокола TCP/IP. При этом на хост-компьютере, на который посылается сообщение, должен быть запущен сервер, использующий компонент на основе TNMGeneralServer, например, компонент TNMMsg.

TNMMsgServ Компонент TNMMsgServ необходим для получения сообщений, посланных компонентом TNMMsg. Заметим, что для решения задач правильнее будет разработать свой собственный протокол обмена сообщениями, чем непосредственно использовать компоненты TNMMsg и TNMMsgServ.

TNMEcho Используется для отправки и получения сообщений через эхо-сервера Интернет по стандарту RFC 862.

TNMFTP Осуществляет передачу файлов в Интернет и интранет между серверами FTP по протоколу FTP (File Transfer Protocol).

TNMHTTP Выполняет передачу файлов по протоколу HTTP (Hypertext Transfer Protocol). Компонент отвечает стандарту HTTP 1.1, который описывается в спецификации RFC 2068.

TNMNNTP Применяется для получения и публикации статей через Интернет и интранет по протоколу NNTP (Network News Transfer Protocol). Описан в RFC 977.

TNMStrm Используется для пересылки потоков данных компоненту TNMStrmServ в Интернет или интранет.

TNMStrmServ Принимает и пересылает потоки, посланные компонентом TNMStrm.

TNMPOP3 Принимает почтовые сообщения от почтовых серверов по протоколу POP3 (Post Office Protocol). Описан в RFC 822.

TNMSMTP Отправляет почту через серверы SMTP (Simple Mail Transfer Protocol) и выполняет другие задачи, описанные в стандарте RFC 821.

TNMTime Получает время от серверов времени Интернет по стандарту RFC 868.

TNMUDP Передает данные по сети с использованием протокола UDP (User Datagram Protocol).

TNMURL Преобразует строку URL в строку символов и обратно.

TNMUUProcessor Кодирует и декодирует файлы по стандарту MIME и UUEncode.

TPowersock Инкапсулирует API сокетов Windows. Компонент является базовым классом для многих компонентов страницы FastNet. Его можно использовать его для создания элементов управления, использующих другие известные протоколы или для создания своих собственных протоколов.

TNMGeneralServer Используется для серверов TCP/IP общего назначения. Используется в качестве базового класса для создания многопотоковых Интернет-серверов как поддерживающих стандарты RFC, так и других.

TNMFinger Получает информацию о пользователе с finger-сервера Интернет, используя протокол Finger, описанный в стандарте RFC 1288.

Компонент **TNMDayTime** предназначается для получения даты и времени от серверов даты и времени в Интернет и интранет в соответствии col стандартом RFC 867, называемым "Протокол времени суток" (Daytime Protocol, 1983). Если у вас в "подчинении" один домашний компьютер,то время на нем можно периодически подводить по своим часам - здесь точность не требуется. Но если вы администратор малой или большой сети или у вас есть многопользовательская программа, то для синхронизации времени на машинах нужно использовать некий механизм. Например, эту возможность, предоставляемую Интернет.

В Интернет используются четыре временных протокола. Протокол, используемый в TNMDayTime, достаточно простой и дает время с погрешностью в 1 секунду. Этому компоненту требуется наличие 32-битного стека протоколов TCP/IP (библиотеки WSOCK32.DLL), поставляемой с операционными системами Windows 95/98/NT. За протоколом даты/времени закреплен порт 13.

Перед тем, как использовать этот компонент, вы должны знать имя или IP-адрес соответствующего сервера Интернет и присвоить это значение свойству Host, в которое помещается имя сервера или разделенный точками его IP-адрес. После того, как вы сделали это и установили соединение, вам остается считать полученное значение из свойства DayTimeStr.

У этого компонента нет собственных методов, все наследуются. Свойства также наследуются, кроме одного - DayTimeStr.

Компонент **TNMTime** получает данные от серверов времени в соответствии с протоколом RFC 868.

Перед тем, как запросить время у сервера времени, необходимо в свойстве Host указать имя сервера или его IP-адрес. Свойство Port изменяется только в том случае, если сервер, к которому обращается программа, "слушает" нестандартный порт.

После задания имени сервера читается содержание свойства TimeStr, в которое записывается полученное от сервера времени, значение.

У этого компонента нет собственных методов и событий, и только два своих свойства TimeStr и TimeInt, остальные наследуются от TComponent и TPowersock.

Эти компоненты обмениваются простыми текстовыми сообщениями в кодах ASCII по Интернет или интранет с использованием протокола TCP/IP. Вы можете использовать их для создания "чата" в Интернет или для программы переговоров по локальной сети. При этом на хост-компьютере, на который посылается сообщение, должен быть запущен сервер, использующий компонент TNMGeneralServer, например, компонент TNMMsgServ, который является производным от компонента TNMGeneralServer.

Для того, чтобы компонент TNMMsg смог отправить сообщение, вам нужно указать в свойстве Host удаленный компьютер, на который вы хотите его отправить. Это может быть имя или IP-адрес. Кроме этого, значение свойства Port должно соответствовать значению этого свойства у компонента на удаленном компьютере. Стандартный порт, который "слушает" серверный компонент, имеет номер 6711. В свойстве FromName укажите имя своего компьютера, чтобы получатель мог знать, от кого получено сообщение. После определения этих свойств вы можете отправить сообщение, вызвав метод PostIt. Если метод выполняется успешно, генерируется событие OnMessageSent. У этого компонента только один свой метод PostIt, остальные наследуются. Свойство FromName - единственное свое, остальные наследуются.

Компонент TNMMsgServ предназначен для получения сообщений, отправленных компонентом TNMMsg. У него нет своих свойств и методов, только унаследованные. Главное событие серверного компонента - это событие OnMSG. Возникает оно при получении сообщения, текст которого передается обработчику события через параметр sMsg. Другой параметр этого обработчика - sFrom - содержит имя отправителя.

TNMEcho

Компонент применяется для отсылки текстовых сообщений на эхо-сервер и получения этого сообщения обратно в соответствии со стандартом RFC 862. Этот компонент обычно применяется для тестирования и настройки сети, поскольку мы можем оценить время ответа сервера, которое помещается в свойство ElapsedTime. Это свойство - единственное собственное свойство компонента, остальные наследуются.

Перед отправкой сообщения должно быть установлено соединение с сервером. Для этого нужно задать имя сервера или его адрес в свойстве Host, а в свойстве Port определить соответствующий порт (обычно серверы эхо "слушают" 7-й порт). После этого нужно вызвать метод Connect. Для отправки сообщения нужно использовать метод Echo. В случае успеха вы должны получить от сервера эхо ту строку, которую вы передавали в качестве параметра методу Echo. После этого, для завершения связи вы должны вызвать метод Disconnect. Все методы, кроме метода Echo, компонент наследует.

Эти компоненты, возможно, не лучший выбор, и вам следовало бы попробовать разработать свой собственный протокол работы с сообщениями, если ваша задача того стоит.

TNMFinger

Компонент TNMFinger применяется для получения информации в Интернет о пользователе от сервера finger, используя протокол, описанный в стандарте RFC 1288.

Для того, чтобы получить информацию о пользователе, помещаемую в свойство FingerStr, вам нужно указать в свойстве Host имя сервера finger. Обычно свойство Port не нужно переопределять, поскольку большинство рассматриваемых серверов использует порт 79. И, конечно, вам нужно указать интересующее вас имя пользователя в свойстве User. Иногда эта служба используется для получения некоторой информации, например, о спорте, погоде. Обратившись по такому адресу пользователя, вы найдете в свойстве FingerStr рассылаемую информацию.

TNMFTP

Компонент TNMFTP предназначается для обмена файлами между сервером FTP и клиентской машиной по протоколу FTP. FTP является одной из старейших и заслуженных служб Интернет. Она существовала тогда, когда еще не было WWW, и предоставляла удобный по тем временам сервис для обмена файлами и организации различных архивов документов и программ. Сейчас получить доступ к серверу FTP можно непосредственно из браузера WWW, так что клиентские программы для работы с сервером FTP уже не так актуальны. Мы рассмотрим пример с этим компонентом, поскольку такая программа может потребоваться для работы с корпоративным FTP-сервером, где доступ разграничен и требуется пройти идентификацию для доступа к файловому архиву.

Перед использованием компонента TNMFTP для обмена файлами с удаленным компьютером вам нужно "подключиться" к серверу FTP. Для этого вы

должны определить свойства Host и Port значениями, соответствующими нужному серверу FTP. Затем задайте в свойствах UserID и Password необходимые имя пользователя и пароль. Многие публичные серверы FTP принимают имя Anonymous в качестве имени пользователя и в качестве пароля ваш e-mail или строку, напоминающую e-mail, например, user@mycomputer.com. Главное, чтобы в этой строке присутствовал символ "коммерческое эт" (@). Таких "анонимных" серверов в Интернет довольно много, и используются они как публичные архивы программ и документов. После определения этих свойств вызывайте метод Connect для установления связи с сервером.

Этот компонент имеет следующие основные свойства:

CurrentDir - имя текущей директории или папки;

FTPDirectoryList - значение учитывается тогда, когда свойство ParseList установлено в True. Свойство FTPDirectoryList содержит разобранный листинг директории, полученный методом List;

Событие OnListItem возникает при чтении директории;

ParseList - определяет возможность разбора директории;

Password - задает пароль для подключения к серверу;

UserID - содержит имя пользователя;

Vendor - используется для определения производителя сервера.

TNMHTTP

Компонент TNMHTTP используется для передачи гипертекста через WWW или интранет по протоколу HTTP. Компонент поддерживает версию протокола HTTP 1.1.

Вы можете использовать соответствующие методы для работы с документами. Компонент имеет семь методов: Get, Head, Options, Trace, Put, Post и Delete. Метод Get применяется для получения гипертекстовых документов с серверов World Wide Web. Запрашиваемый документ указывав ется в параметре URL. Дальнейшая судьба документа зависит от значении свойства InputFileMode типа Boolean. Если свойству присвоено значение True, то заголовок документа и его тело помещаются в текстовый и HTML файлы соответственно. Имена файлов для этого случая определяются в свойствах Header и Body. Если свойство InputFileMode выставлено в False, то заголовок и тело документа размещаются непосредственно в свойствах Header и Body. Метод Head аналогичен методу Get, но с его помощью мож но загрузить только заголовок документа. Заголовок тем же образом может помещаться либо в файл, либо в свойство Head.

Методы Post и Put выполняют обратную задачу - публикацию документов в WWW. Метод Post позволяет дописать данные в существующий документ на сервере. Метод Put создает новый документ на сервере World Wide Web. Эти методы имеют два параметра. Первый из них - это параметр с именем URL, в котором указывается имя документа на Web-сервере. Второй параметр типа String по имени PostData у метода Post и PutData у метода Put используется для задания публикуемых данных. Данные передаются непосредственно через этот строковый параметр, если свойство OutputFileMode имеет значение False. Если этому свойству присвоили значение True,

то тогда во втором параметре указываются путь и имя файла, в котором размещаются данные для публикации. Нужно заметить, что для того, чтобы опубликовать документ на сервере Web, у вас должны быть соответствующие права. Как и права на удаление файла, если вы используете метод Delete, с помощью которого можно удалять документы на удаленном сервере.

Метод Abort прерывает выполнение транзакции.

Оставшиеся два метода предоставляют дополнительные возможности. Метод Trace применяется при отладке. Метод использует указанный в первом параметре ресурс для отображения отладочных данных, передаваемых через второй параметр TraceData. Данные передаются через этот параметр уже известным способом в зависимости от значения свойства OutputFileMode.

И последний метод - метод Options позволяет получить справочную информацию о запрашиваемом ресурсе и командах (методах), разрешенных на данном сервере. Выполнить все методы на большинстве серверов вам не удастся. Обычно вам разрешат использовать методы Get, Head, Options и Trace.

Компонент имеет свойства Port и Host, которые здесь излишни, поскольку эта информация задается через парметр URL методов этого компонента. Свойствам InputFileMode и OutputFileMode присваиваются по умолчанию значения False. В составном свойстве HeaderInfo имеются свойства UserID и Password, необходимые в том случае, если вам нужно публиковать данные на Web-сервере. Понятно, что в этом случае вам потребуются соответствующие права доступа, и, значит, регистрация на сервере под некоторым именем, после ввода которого придется указать и пароль. Еще вам могут понадобиться свойства BytesRecvd, BytesSent и BytesTotal, которые указывают число принятых байтов данных, отправленных данных и общее число передаваемых байтов.

TNMNNTP

Компонент TNMNNTP используется для чтения и публикации сообщений в телеконференциях Интернет (NetNews, NewsGroups) в соответствии с протоколом NNTP (Network News Transfer Protocol). Протокол описан в стандарте RFC 977.

Наряду с электронной почтой и серверами FTP телеконференции являются классическим сервисом Интернет. Ветераны Интернет продолжают пользоваться им весьма активно, но новые пользователи, ввиду засилья WWW, знают этот сервис плохо, хотя работу с новостями поддерживают не только специальные программы, но и Microsoft Outlook Express и Netscape Navigator. Телеконференции имеют иерархическую структуру. Название конференции состоит из нескольких слов, разделенных точками. Чтобы получить доступ к этому сервису, вам нужно подключиться к одному из серверов новостей. Обычно поставщики услуг Интернет наряду с электронной почтой предоставляют доступ к своему серверу новостей. Если у вашего провайдера нет своего сервера, то вам придется найти его с помощью поисковых систем Интернет. В отличие от WWW, для доступа к которой не требуется использования какого-то специфического сервера, с электронными новостями нужно работать именно так - через сервер. Подключившись к серверу, вы должны сначала загрузить полный список конференций, который он поддерживает. Сервер, к которому

вы подключитесь, возможно, будет иметь не полный список, а какой-то ограниченный набор телеконференций Интернет. После этого вам нужно подписаться на какое-то количество конференций и загрузить из них заголовки сообщений. Ваша подписка будет сохраняться на все последующие сеансы связи, пока вы не откажетесь от нее. Размещение сообщений в конференции часто называется публикацией (posting), а само сообщение называется словом статья (article).

Итак, прежде, чем начинать работу с сервером новостей, вам нужно подключиться к одному из таких серверов. Это можно сделать, задав нужное имя сервера новостей в свойстве Host и вызвав метод Connect, который подключит вас к выбранному серверу. После этого вам нужно получить список групп новостей этого сервера. Для этой цели вызывается метод GetGroupList, который помещает полученный список в свойство GroupList. После этого нужно выбрать одну из групп новостей, в которой вы будете просматривать сообщения. Для этого вызывается метод SetGroup, которому в качестве параметра передается имя выбранной вами группы новостей.

Чтение статей в выбранной группе новостей осуществляется при помощи вызова метода GetArticle. Но перед этим нужно загрузить список сообщений из группы методом GetArticleList. После этого для каждой интересующей вас статьи нужно вызывать метод GetArticle, который будет помещать в свойство Body текст статьи, а в свойство Header - заголовок выбранной статьи.

Для того, чтобы отправить статью в группу новостей, сначала нужно заполнить свойство PostBody текстом сообщения, в свойстве PostHeader разместить заголовок сообщения, и после этого вызвать метод PostArticle.

После обсуждения порядка работы с компонентом поговорим о его свойствах, методах и событиях. Основными свойствами являются свойства GroupList, SelectedGroup, Body, Header, CurrentArticle, HiMessage, LoMessage, PostBody и PostHeader. В свойстве GroupList типа TStringList размещается список групп новостей выбранного сервера. Свойство заполняется в случае успешного вызова метода GetGroupList. После успешного вызова метода SetGroup в свойство SelectedGroup помещается имя выбранной группы. Метод GetArticleList имеет два параметра: AH типа boolean и ArticleNumber - целого типа. Если первый параметр выставляется в True, то считываются все статьи группы, в противном случае второй параметр задает номер сообщения, с которого нужно начинать загрузку. При загрузке очередной статьи происходит событие OnHeaderList. В свойстве CurrentArticle помещается номер текущей статьи. Это свойство получает значение после вызова одного из методов GetArticle, GetArticleBody или GetArticleHeader. Можно сказать, что в этом свойстве помещается номер той статьи, текст и заголовок которой находятся в свойствах Body и Header. В свойствах HiMessage и LoMessage прописываются максимальный и минимальный номера сообщений, доступных в выбранной группе. В свойства PostBody и PostHeader помещаются тело и заголовок отправляемого в группу сообщения. Возможность или невозможность публикации сообщения в выбранной группе определяется значением свойства Posting типа boolean. Среди свойств компонента имеются составные свойства HeaderRecord и PostHeader, куда записываются тема сообщения, электронный адрес отправителя и электронный адрес для ответа, имя группы или групп, в которых опубликовано данное сообщение, время

и дата публикации и другие данные. В первом свойстве размещаются данные полученного сообщения, а во втором - отправляемого. В технологии электронных новостей, как и в электронной почте, применяется присоединение или вложение в сообщение одного или нескольких файлов. Свойства AttachFilePath, Attachments, PostAttachments и ParseAttachments служат для работы с вложенными файлами. Значение логического свойства ParseAttachments, равное True, задает режим разбора вложенных файлов, при котором файлы будут декодироваться. После этого они будут помещаться с именами, под которыми они перечислены в свойстве Attachments, в каталоге, заданном в свойстве AttachFilePath. Если в ParseAttachments записано False, то файлы просто остаются в теле сообщения. Если вы присоединяете к отправляемому сообщению некоторое количество файлов, то их имена перечисляются в свойстве PostAttachments. В случае если требуется регистрация пользователя на сервере новостей, в свойства UserID и Password помещаются имя пользователя и пароль.

4 ПЕРЕЧЕНЬ ПРОГРАММНЫХ ПРОДУКТОВ, ИСПОЛЬЗУЕМЫХ В ПРЕПОДАВАНИИ ДИСЦИПЛИНЫ

В процессе обучения студенты используют следующие программные продукты:

- Borland Delphi
- Borland C++ Builder

Основная цель применения данных программных продуктов – дать студентам первоначальные знания в области современных средств программирования в интернет и обучить основным навыкам работы с распространенным ПО.

5 МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ПРИМЕНЕНИЮ СОВРЕМЕННЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

В процессе изучения дисциплины «Программирование в интернет» используются современные информационные технологии – в частности при самостоятельной работе рекомендуется использовать сеть Интернет.

Рекомендуемые Интернет ресурсы:

1. Поисковые системы – www.rambler.ru, www.yandex.ru, www.google.com и т.д.
2. Специализированные сайты – www.3dnews.ru, www.ferra.ru, www.citforum.ru, www.ixbt.com.

6 КАРТА ОБЕСПЕЧЕНИЯ ДИСЦИПЛИНЫ КАДРАМИ ПРОФЕССОРСКО-ПРЕПОДАВАТЕЛЬСКОГО СОСТАВА

№ п/п	Наименование дисциплины в соответствии с учебным планом	Обеспеченность преподавательским составом								
		ФИО должность по штатному расписанию	Какое образовательное учреждение профессионального образования окончил, специальность по диплому	Ученая степень и ученое звание	Стаж научно педагогической работы			Основное место работы, должность	Условия привлечения к трудовой деятельности(Штатный, совместитель; внутренний или внешний, с указанием доли ставки)	Кол-во часов
					Всего	В т.ч. педагогический				
						Всего	В том числе по преподаваемой дисциплине			
	Программирование в интернет	Степанов А.Е., Ассистент кафедры ИУС	Амурский государственный университет, инженер по специальности «Автоматизированные системы обработки информации и управления»	-	1 год	1 год	5 мес	Кафедра ИУС, ассистент	Совместитель, 0,25 ст	