FastReport® 3 Enterprise Edition

руководство программиста

версия от 18.03.2005

Оглавление

Введение ... 3

1. FastReport 3 Enterprise - построение отчетов по технологии Client/Server ... 4

2. Компоненты FastReport Enterprise Edition ... 6

2.1. TfrxReportServer ... 6
2.2. TfrxServerConnection ... 9
2.3. TfrxReportClient ... 10
2.4. TfrxHTTPClient ... 10

3. Сервер ... 13

3.1. Внутренняя архитектура и принцип работы ... 13
3.2. Поддерживаемые форматы результатов выполнения отчета ... 17
3.3. Синтаксис запросов ... 19
3.4. Передача параметров в отчет ... 20
3.5. Внутренние переменные сервера ... 20
3.6. Использование произвольных HTML документов ... 21
3.7. Журналы доступа, ошибок и служебной информации ... 21
3.8. Аутентификация ... 22
3.9. Ограничение доступа к серверу по IP адресу ... 23
3.10. Варианты подключений к базам данных ... 24
3.11. Использование кеша отчетов ... 24
3.12. Увеличение производительности сервера ... 25
3.13. Использование сервера FastReport совместно с HTTP серверами Apache, IIS и другими ... 26

4. Разработка отчетов ... 28

4.1. Ограничения при разработке отчетов для FastReport Client/Server ... 28
4.2. Рекомендации по разработке отчетов, результат которых будет представлен в табличных форматах ... 28

5. Клиент ... 29

5.1. Клиент на основе FastReport 3 ... 29
5.2. Другие возможные клиенты ... 29

6. Перевод существующих приложений на платформу Client/Server ... 30

7. Примеры ... 31

7.1. Пример простейшего Client/Server приложения ... 31
7.1.1. Серверная часть ... 31
7.1.2. Клиентская часть ... 33
7.1.3. Клиентская часть с несколькими потоками ... 34

8. Важные замечания по безопасности ... 37

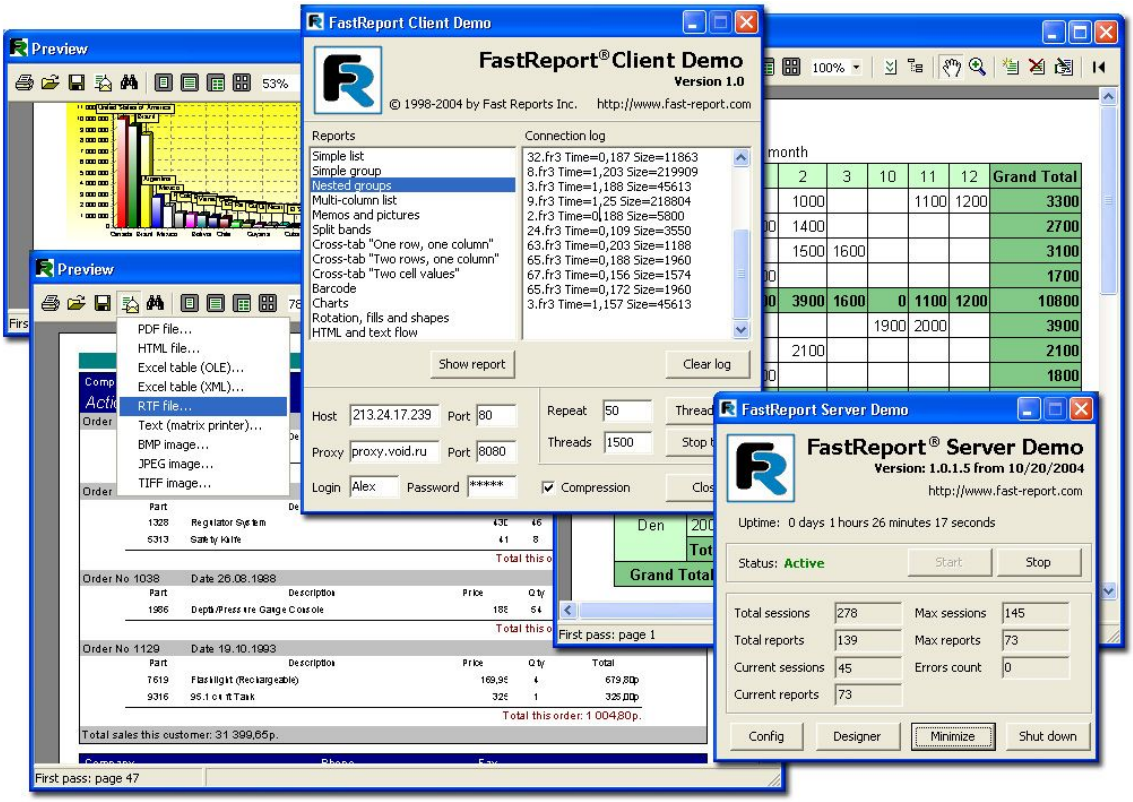
9. Дополнительные источники информации ... 38

10. Связь с разработчиками ... 39

Введение

Данное руководство программиста содержит информацию о расширении библиотеки FastReport 3, позволяющем выполнять построение отчетов по технологии клиент-сервер с применением уже существующих компонент FastReport 3 и дополнительных, служащих взаимодействию клиента и сервера.

FastReport 3 – генератор отчетов с высокой производительностью и уникальными возможностями. Подробное описание генератора отчетов содержится в книгах “FastReport 3 – Руководство разработчика” [7], “FastReport 3 – Руководство программиста” [8], “FastReport 3 – Руководство пользователя” [9].



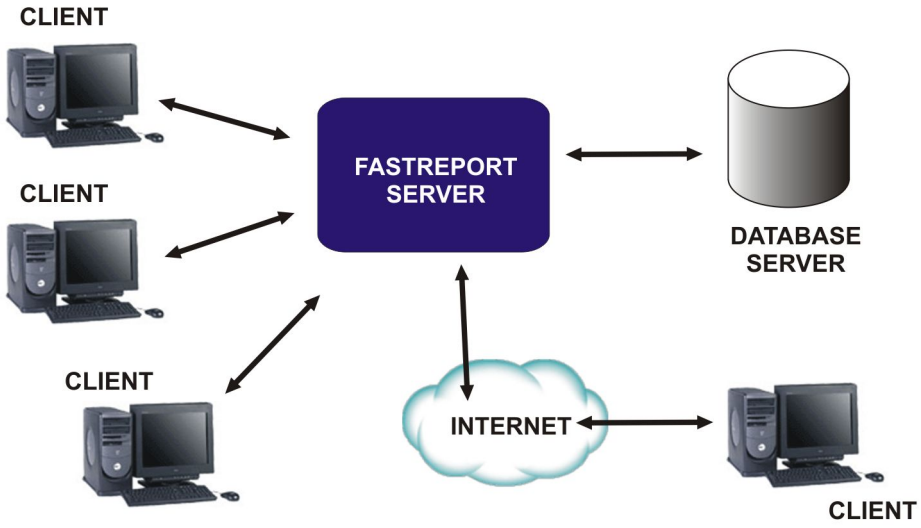
В руководстве отражена структура клиентских и серверных компонент, их свойства и методы, внутренняя архитектура сервера отчетов и принцип его работы, даны рекомендации по оптимизации и применению новых возможностей в уже существующих и вновь разрабатываемых приложениях.

Опытным пользователям FastReport будут интересны рекомендации по увеличению быстродействия серверных компонент, оптимизации отчетов для корректного экспорта их в различные табличные форматы, применению правил информационной безопасности для защиты своих приложений от несанкционированного доступа.

В связи с постоянной модификацией компонент FastReport 3 Enterprise в данном руководстве могут быть не отражены их некоторые возможности. Все изменения будут обязательно внесены в следующую версию данного руководства.

1. FastReport 3 Enterprise - построение отчетов по технологии Client/Server.

Технология клиент-сервер базируется на взаимодействии клиентского приложения, осуществляющего запрос, обработку или отображение полученной информации, и серверного приложения, выполняющего основную работу связанную с большим количеством вычислений либо просто из целей централизованной обработки большого массива информации по запросам клиентов.



Применение технологии клиент-сервер в приложениях имеет ряд существенных преимуществ по отношению к традиционным решениям:

- меньшие аппаратные требования к компьютерам, установленным на клиентских местах;
- уменьшение сетевого трафика за счет уменьшения объемов пересылаемой информации между сервером базы данных и клиентским приложением;
- простота администрирования уже разработанного клиент-сервер приложения;
- централизация мер по осуществлению защиты информации.

Среди недостатков применения технологии клиент сервер можно отметить следующие:

- повышенные аппаратные требования к компьютеру, работающего в качестве сервера;
- определенные сложности при разработке клиент-сервер приложения, решаемые в основном за счет четкого планирования архитектуры на ранних этапах разработки.

При разработке FastReport 3 Enterprise в полной мере были учтены все основные требования к клиент-сервер приложениям, что позволило добиться следующих возможностей:

- построение отчетов любой сложности на стороне сервера по запросу клиента без непосредственного доступа клиента к серверу баз данных;
- обслуживание нескольких клиентов сервером в различных потоках позволяет добиться высокой нагрузочной способности и минимизации времени

отклика сервера;

- применение протокола передачи данных HTTP (RFC 2068 [2]) позволяет использовать большое количество уже существующих программ, таких как web-браузеры (Internet Explorer, Netscape Navigator, Mozilla, Opera и др.), Proxy серверы, web-серверы (Internet Information Server, Apache и др.) для совместной работы без дополнительных трудоемких решений;

- примение технологий сжатия на основе алгоритма GZip (RFC 1952 [6]) уменьшает сетевой трафик и увеличивает общую производительность клиент-сервер системы;

- применение контрольных сумм на основе алгоритма MD5 в качестве MIC (Message Integrity Checksum) полей в заголовках (RFC 1321 [4], RFC 1864 [5]) увеличивает целостность переданных данных и уменьшает вероятность ее искажения;

- совместимость с ранее разработанными отчетами FastReport 3, с некоторыми ограничениями, упрощает процесс перевода уже разработанных приложений на технологию клиент-сервер;

- реализация полностью автономного сервера позволяет увеличить производительность (уменьшить время построения отчета), минимизировать затраты на использование оперативной памяти и процессора по отношению к решениям для построения отчетов на основе технологии CGI, в тот же момент такой подход никак не сказывается на взаимодействии с уже работающими HTTP серверами;

- сервер может быть использован в качестве простого HTTP сервера для хранения и отображения любых HTML документов, что позволяет обойтись без применения других HTTP серверов;

- примение технологии SSI (Server Side Include) позволяет упростить построение web-сайта под управлением сервера FastReport;

- ведение журналов доступа, ошибок и другой служебной информации позволяет оперативно диагностировать неисправности, вести статистику работы и отслеживать попытки несанкционированного доступа к серверу;

- применение аутентификации, а также списков разрешенных и запрещенных к обслуживанию IP адресов позволяют ограничить доступ к серверу;

- работа с несколькими базами данных одновременно для получения перекрестных отчетов (при применении подключений к базам данных в самих отчетах);

- использование в качестве клиента не только внутренних компонент FastReport, но и любого web-браузера;

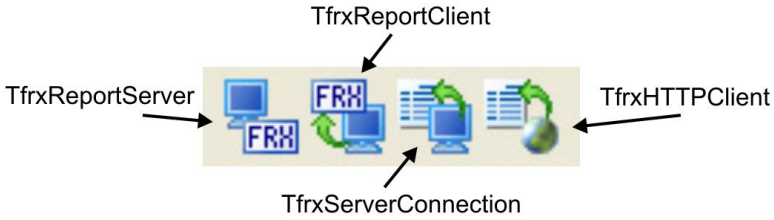
- конвертация диалоговых форм FastReport в web-формы на лету позволяет реализовать элементы концепции web-приложений без дополнительных трудоемких решений;

- поддерживаемые форматы результатов отчета: HTML, PDF, RTF, XML, XLS, JPEG, Text;

- при использовании в качестве клиента web-браузера во время просмотра результатов отчета в HTML формате доступен удобный навигатор страниц.

2. Компоненты FastReport Enterprise Edition.

После установки пакетов FastReport 3 Enterprise будет доступна закладка “FastReport 3 Client/Server” в палитре компонент среды разработки Delphi/C++Builder.



Компоненты “FastReport 3 Client/Server”:

- TfrxReportServer – серверный компонент, сервер отчетов и HTTP-сервер в едином целом;

- TfrxServerConnection – клиентский компонент, содержащий информацию для удаленного подключения к TfrxReportServer;

- TfrxReportClient - клиентский компонент, аналог TfrxReport, выполняет запрос отчета на сервере и отображение его на клиенте;

TfrxHTTPClient – клиентский компонент, предназначенный для получения произвольных файлов по протоколу HTTP.

2.1. TfrxReportServer.

TfrxReportServer компонент, выполняющий роль сервера отчетов и HTTP-сервера. Для своего функционирования не требует никаких дополнительных компонент.

Класс TfrxReportServer – наследник TComponent

Свойства:

Active: Boolean – значение указывает на активность сервера, может быть использовано для запуска сервера путем установки значения в True;

Configuration: TfrxServerConfig – конфигурация сервера (класс TfrxServerConfig будет описан ниже), изменения в конфигурации станут активными только при последующем запуске сервера;

AllowIP: TStrings – список разрешенных к обслуживанию IP адресов, формат списка: одна строка – один IP адрес, в случае, если сервер не найдет адрес клиента в этом списке, ему будет отказано в обслуживании, если список пуст – будут обслуживаться клиенты с любыми IP адресами;

DenyIP: TStrings - список запрещенных к обслуживанию IP адресов, формат списка: одна строка – один IP адрес, в случае, если сервер найдет адрес клиента в этом списке, ему будет отказано в обслуживании, если список пуст – будут обслуживаться клиенты с любыми IP адресами;

PrintPDF: Boolean – если установлено в True, по при просмотре HTML страниц в web-браузере при нажатии на кнопку печать в навигаторе страниц отчета будет создан PDF файл и передан клиенту для последующей печати, если свойство установлено в False – для печати будет вызываться штатная функция печати web-

браузера, по умолчанию свойство установлено в True;

Следующие свойства недоступны в инспекторе объектов, но к ним можно обращаться из программы:

Statistic: TfrxServerStatistic – статистика работы сервера с момента его запуска (класс TfrxServerStatistic будет описан ниже);

Totals: TStrings – статистика сервера в виде, удобном для восприятия;

Variables: TfrxServerVariables – внутренние переменные сервера (класс TfrxServerVariables будет описан ниже).

Методы:

constructor Create(AOwner: TComponent) – создание объекта;

procedure Open – запуск сервера, в момент вызова этого метода присваиваются все изменения в конфигурации сервера;

procedure Close – останов сервера;

Обработчики событий:

OnGetReport: TfrxServerGetReportEvent – может быть использован для загрузки отчетов из произвольных мест (BLOB поля баз данных, файлы из различных каталогов и пр.) тип обработчика следующего вида

TfrxServerGetReportEvent = procedure(ReportName: String; Report: TfrxReport) of object;

ReportName – имя запрашиваемого отчета, может быть использовано для идентификации конкретного отчета;

Report – экземпляр объекта TfrxReport, в который необходимо загрузить отчет.

OnGetVariables: TfrxServerGetVariablesEvent – может быть использован для ручной обработки полученных от клиента параметров и выполнения каких-либо действий, непосредственно на сервере.

TfrxServerGetVariablesEvent = procedure(const ReportName: String; Variables: TfrxVariables) of object;

ReportName – имя отчета, переданное в запросе, может быть использовано для фильтрации тех или иных параметров непосредственно в обработчике;

Variables – список полученных от клиента параметров, класс TfrxVariables подробно описан в “Руководстве программиста FastReport 3” [8].

Класс TfrxServerConfig – наследник TPersistent

Содержит конфигурацию сервера.

Свойства:

Port: Integer – номер TCP/IP порта, для обслуживания клиентов, по умолчанию равен 80;

IndexFileName: String – имя файла по умолчанию, если в HTTP запросе имя файла не указано, по умолчанию равен index.html;

SessionTimeOut: Integer – время хранения результатов отчета на сервере в секундах, по умолчанию равно 300, по истечении этого времени результаты будут

удалены с сервера, настраивается в зависимости от специфики создаваемых отчетов и методов взаимодействия клиента и сервера;

SocketTimeOut: Integer – время ожидания активности клиента после его подключения в секундах, по умолчанию равно 60, по истечении этого времени сессия клиента будет удалена;

Logging: Boolean – ведение журналов, True – включены, False – выключены, по умолчанию – True;

LogPath: String – путь к папке хранения журналов, по умолчанию – текущий каталог;

ReportPath: String – путь к папке хранения шаблонов отчетов, по умолчанию текущий каталог;

RootPath: String – путь к HTML документам и результатам отчетов;

Login: String – имя пользователя для аутентификации, если пустая строка, то аутентификация не запрашивается, по умолчанию - пустая строка;

Password: String – пароль для аутентификации, по умолчанию - пустая строка;

Compression: Boolean – сжатие передаваемых документов, при соответствующей поддержке сжатия клиентом, по умолчанию – True;

MIC: Boolean – Message Integrity Checksum, использование контрольных сумм MD5, по умолчанию – True;

NoCacheHeader: Boolean – передача команд запрета кэширования файлов клиентом в заголовке, по умолчанию – True;

OutputFormats: TfrxServerOutputFormats – разрешенные форматы для получения результатов отчета, может содержать одно или несколько значений из множества (sfHTM, sfXML, sfXLS, sfRTF, sfTXT, sfPDF, sfJPG, sfFRP), по умолчению включено все множество;

ReportCaching: Boolean - разрешить кеширование отчетов на сервере (подробности см. в разделе 3.11);

ReportCachePath: String – путь к папке с кешем отчетов;

DefaultCacheLatency: Integer – время хранения отчета в кеше по умолчанию в секундах.

Методы:

procedure LoadFromFile(const FileName: String) – загрузка конфигурации из файла, формат файла соответствует принятому формату INI файлов;

procedure SaveToFile(const FileName: String) – сохранение конфигурации в файл, формат файла соответствует принятому формату INI файлов.

Класс TfrxServerStatistic – наследник TPersistent

Свойства:

CurrentReportsCount: Integer – количество строящихся в данный момент отчетов;

CurrentSessionsCount: Integer – количество обслуживаемых в данный момент сессий;

MaxReportsCount: Integer – максимальное количество построенных одновременно отчетов;

MaxSessionsCount: Integer - максимальное количество одновременных сессий;

TotalErrors: Integer – общее количество ошибок;

TotalReportsCount: Integer – общее количество построенных отчетов;

TotalSessionsCount: Integer – общее количество сессий;

UpTimeDays: Integer (дни),

UpTimeHours: Integer (часы),

UpTimeMins: Integer (минуты),

UpTimeSecs: Integer (секнды) – время непрерывной работы сервера с момента запуска.

Класс TfrxServerVariables - наследник TCollection

Служит для хранения внутренних переменных сервера.

Использованные в данный момент имена переменных описаны в разделе 3.4. настоящего руководства.

Методы:

function GetValue(const Name: String): String – возвращает значение переменной с именем Name;

procedure AddVariable(const Name: String; const Value:String) – добавляет переменную с именем Name и значением Value.

2.2. TfrxServerConnection

TfrxServerConnection – клиентский компонент, содержащий информацию для подключения к серверу отчетов TfrxReportServer. Необходим для функционирования одного или нескольких компонент TfrxReportClient.

Класс TfrxServerConnection – наследник TComponent

Свойства:

Host: String – сетевое имя сервера или его IP адрес, по умолчанию - 127.0.0.1;

Port: Integer – порт для подключения к серверу, по умолчанию 80;

ProxyHost: String - сетевое имя промежуточного HTTP-прокси сервера или его IP адрес, по умолчанию пустая строка;

ProxyPort: Integer – порт HTTP-прокси сервера, по умолчанию – 8080;

Login: String – имя пользователя для аутентификации на сервере;

Password: String – пароль для аутентификации на сервере;

Timeout: Integer – время ожидания данных в секундах, по умолчанию – 120;

RetryCount: Integer – количество повторов, если произошла ошибка или истек таймаут ожидания, по умолчанию – 3;

RetryTimeout: Integer – время ожидания в секундах перед повтором, по умолчанию – 3;

Compression: Boolean – сжатие при приеме файлов, по умолчанию - True;

MIC: Boolean – проверка контрольной суммы, по умолчанию – True.

2.3. TfrxReportClient

TfrxReportClient – клиентский компонент, осуществляющий запрос, получение и обработку результатов отчета на машине клиента. Для функционирования необходим сконфигурированный компонент TfrxServerConnection. TfrxReportClient может служить заменой TfrxReport в ранее разработанных приложениях, базирующихся на традиционной архитектуре.

Класс TfrxReportClient – наследник TfrxReport

Свойства:

Connection: TfrxServerConnection – ссылка на описанный выше объект класса TfrxServerConnection;

ReportName: String – имя запрашиваемого отчета с сервера, вместо заполнения данного свойства можно использовать метод LoadFromFile (см. ниже).

Перечисленные ниже свойства унаследованы от **TfrxReport**, но могут быть использованы при работе с TfrxReportClient, подробное описание этих свойств можно найти в книге “FastReport 3 – руководство программиста” [8]:

Variables: TfrxVariables – содержит переменные отчета, может быть использовано для передачи параметров (переменных) на сервер при запросе отчета;

Errors: TStrings – содержит лог ошибок в случае неудачного запроса отчета.

Методы:

procedure LoadFromFile(FileName: String) – устанавливает имя запрашиваемого отчета с сервера в свойство ReportName, путь к имени файла игнорируется, наименование метода выбрано исходя из целей совместимости с ранее разработанными приложениями;

function PrepareReport: Boolean – осуществляет подключение к серверу отчетов, выполняет запрос отчета с передачей параметров и получает от сервера результат, который помещается в свойство PreviewPages, результат равен True в случае успешного выполнения, иначе – False;

procedure ShowPreparedReport – вызывает предварительный просмотр ранее полученного отчета;

procedure ShowReport – выполняет запрос отчета, и вызывает его предварительный просмотр.

2.4. TfrxHTTPClient

TfrxHTTPClient – клиентский компонент, предназначенный для получения произвольных файлов по протоколу HTTP, не требует применения других компонент.

Класс TfrxHTTPClient – наследник TComponet

Свойства:

Active: Boolean – если установлен в True, выполняется запрос;

Host: String – сетевое имя сервера или IP адрес, по умолчанию – 127.0.0.1;

Port: Integer – порт для подключения, по умолчанию 80;

ProxyHost: String – имя или IP адрес HTTP-прокси сервера;

ProxyPort: Integer – порт HTTP-прокси сервера;

RetryCount: Integer – количество повторов, если при запросе или приеме файла произошла ошибка, по умолчанию - 3;

RetryTimeOut: Integer – пауза между повторами в секундах, по умолчанию – 5;

TimeOut: Integer – время ожидания ответа секундах, по умолчанию - 30;

ClientFields: TfrxHTTPClientFields – поля заголовка запроса, класс TfrxHTTPClientFields описан ниже;

ServerFields: TfrxHTTPServerFields – поля ответа сервера, класс TfrxHTTPServerFields описан ниже;

MIC: Boolean – проверка контрольной суммы полученного файла (Message Integerity Checksum), по умолчанию True;

Следующие свойства недоступны в инспекторе объектов, но к ним можно обращаться из программы:

Header: TStrings – заголовок сформированного запроса, генерируется перед запросом автоматически на основе полей, перечисленных в свойстве ClientFields;

Answer: TStrings – заголовок принятого ответа от сервера, основные поля разбираются автоматически и записываются в свойство ServerFields;

Stream: TMemoryStream – полученный поток данных от сервера, его можно использовать в дальнейшем для записи в файл или другой обработки;

Breaked: Boolean – признак экстренного завершения запроса;

Errors: TStrings – журнал ошибок, возникших в процессе подключения.

Методы:

procedure Connect – осуществляет подключение к удаленному серверу и запрос файла, после получения файла или при возникновении ошибки происходит автоматическое отключение от сервера;

procedure Disconnect – отключение от сервера;

procedure Open – то-же что и Connect;

procedure Close – то-же что и Disconnect.

Класс TfrxHTTPClientFields – наследник TPersistent

Свойства:

AcceptEncoding: String – поддерживаемые форматы сжатия, по умолчанию содержит 'gzip';

FileName: String – имя запрашиваемого файла;

Host: String – адрес или сетевое имя клиента, если поле пустое, то генерируется автоматически;

HTTPVer: String – версия протокола HTTP, по умолчанию содержит 'HTTP/1.1';

Login: String – имя пользователя для аутентификации на сервере;

Password: String – пароль для аутентификации на сервере;

QueryType: TfrxHTTPQueryType – тип запроса, qtGet – запрос формата GET, qtPost – запрос формата POST, по умолчанию установлено в qtGet;

Referer: String – адрес сервера и документ, ссылающиеся на запрашиваемый файл, по умолчанию – пустая строка;

UserAgent: String – наименование программы клиента, по умолчанию содержит 'FastReport/3.0'.

Класс TfrxHTTPServerFields – наследник TPersistent

Свойства:

AnswerCode: Integer – код полученного ответа от сервера;

ContentEncoding: String – формат сжатия принятых данных;

ContentMD5: String – контрольная сумма принятых данных;

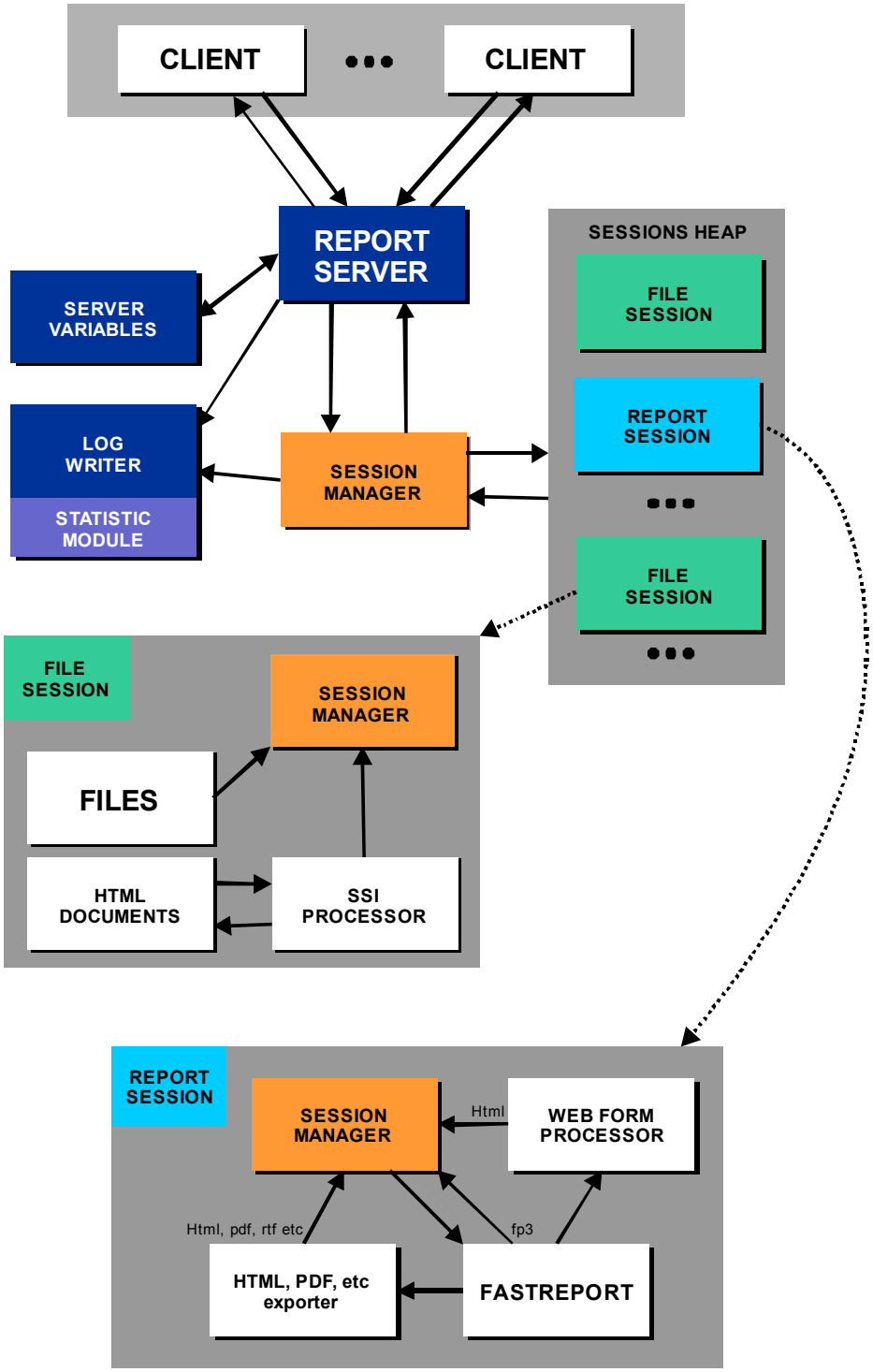
ContentLength: Integer – размер полученного файла;

Location: String – реальное расположение файла на сервере.

3. Сервер.

Серверная часть (компонент TfrxReportServer) представляет собой автономный HTTP сервер с возможностью генерации отчетов. Сервер способен обслуживать одновременно нескольких клиентов, вести журналы и собирать статистику.

3.1. Внутренняя архитектура и принцип работы.



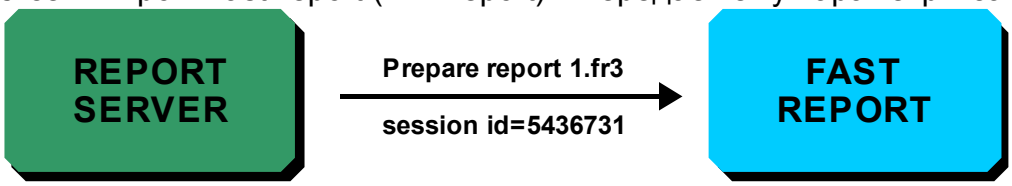
Принцип работы сервера наглядно показан на следующей схем

При поступлении запроса от клиента создается сессия с уникальным идентификатором. Строка запроса анализируется. Если запрашивается файл и он существует на сервере, клиенту возвращается положительный ответ и запрашиваемый файл. В журналах делается соответствующая запись. Если в строке запроса содержится команда построения отчета, создается специальная сессия отчета, в которой производится построение запрашиваемого отчета, результат сохраняется в папку с именем, соответствующем номеру сессии, и клиенту возвращается команда перенаправления на соответствующий файл результата. Клиент, получив команду перенаправления на файл результата, делает повторный запрос, теперь уже файла результата. Сервер выполняет запрос, передавая файл результата клиентской программе. Сессия с файлом результата сохраняется в списке сессий определенное время (время сохранения сессии), после чего удаляется.

Наглядно можно проследить действия клиента и сервера в случае запроса на построение отчета. В качестве клиента будет выступать какой-либо web-браузер:

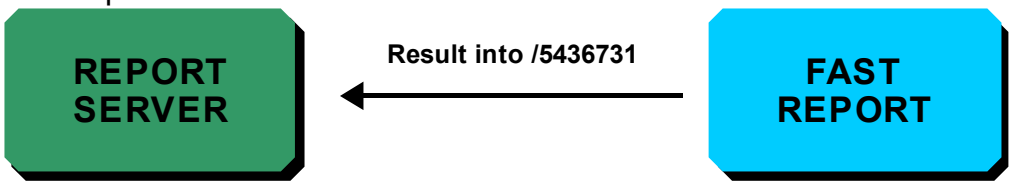
- клиент запрашивает отчет с именем 1.fr3

- сервер определяет, что поступил запрос на построение отчета, создает поток с экземпляром FastReport (TfrxReport) и передает ему параметры запроса



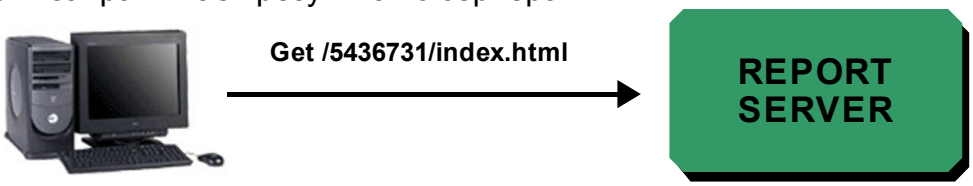
- FastReport выполняет отчет, сохраняет результаты в формате HTML (web-клиент) в папку с номером сессии

- поток сессии, ожидающий выполнения отчета, определяет завершение работы FastReport



- клиенту возвращается перенаправление на результат

- клиент запрашивает результат с сервера

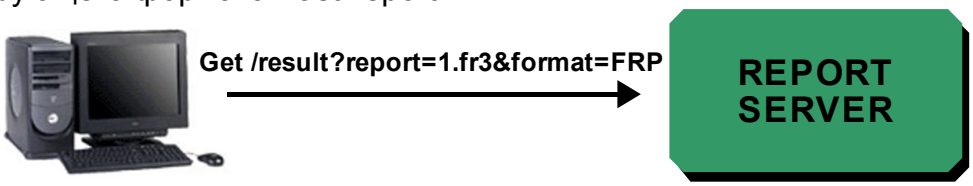


- сервер возвращает файл результата

Если клиентом выступает приложение на основе FastReport 3 (TfrxReportClient), обработка запроса будет выглядеть аналогичным образом:

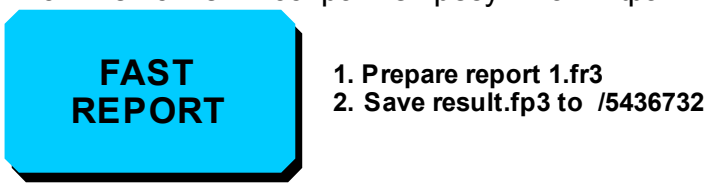
- на клиенте обрабатывается команда показа отчета с именем 1.fr3:

- клиентская компонента посылает запрос на сервер с указанием результирующего формата FastReport

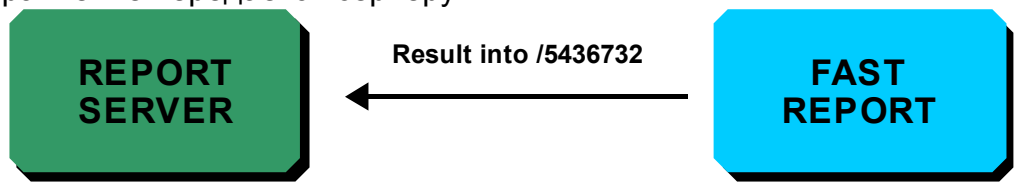


...

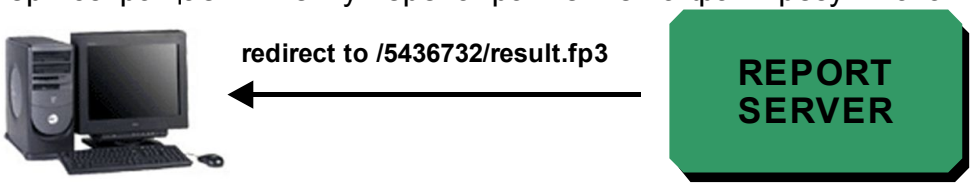
- FastReport выполняет отчет и сохраняет результат в файл



- управление передается серверу

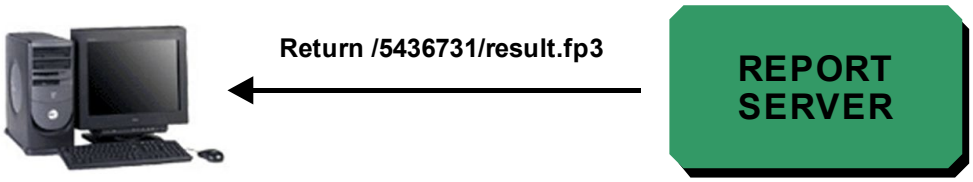


- сервер возвращает клиенту перенаправление на файл результата

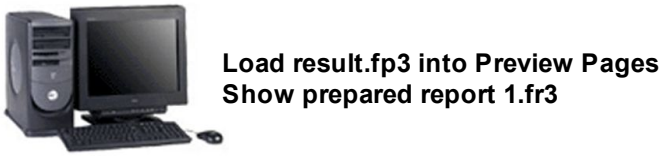


...

- клиент получает файл результата



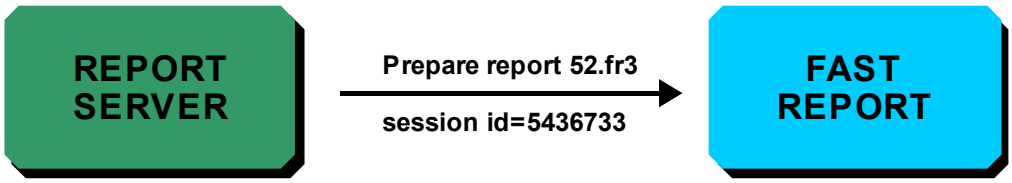
- и отображает предварительный просмотр результатов отчета



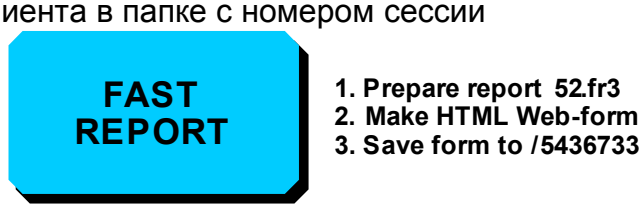
Если запрашиваемый отчет содержит формы, процесс усложняется:

- клиент запрашивает отчет с именем 52.fr3

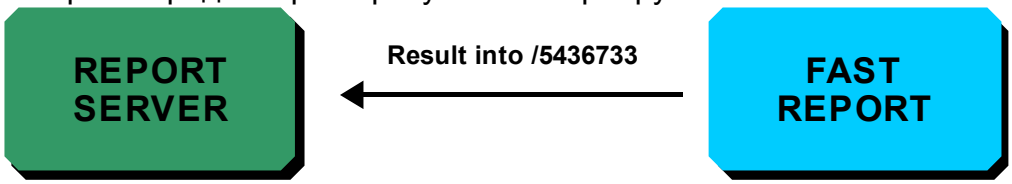
- сервер присваивает идентификатор сессии и передает команду FastReport на построение отчета



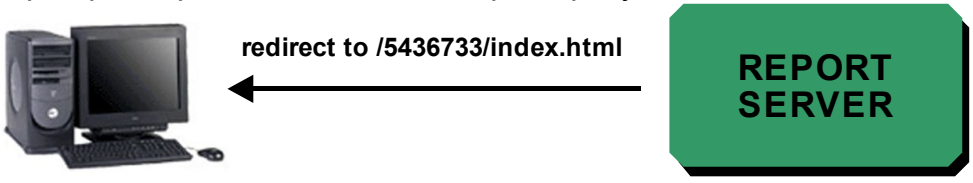
- FastReport в процессе построения создает файл с описанием формы для конкретного типа клиента в папке с номером сессии



- FastReport передает файл результата серверу

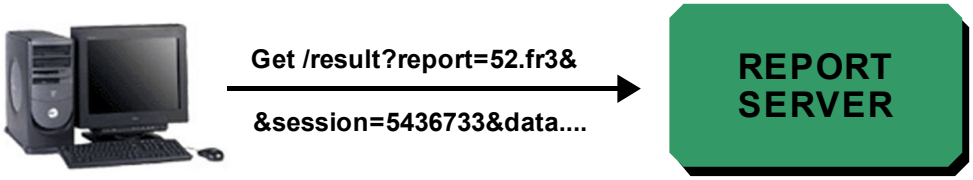


- сервер перенаправляет клиента на файл результата

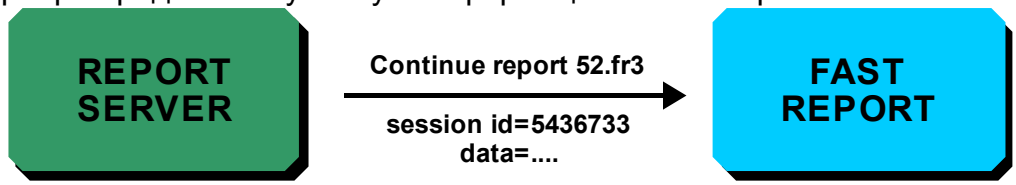


- клиент получает файл формы, отображает ее, пользователь вводит информацию, тем временем сессия построения отчета на сервере находится в состоянии ожидания

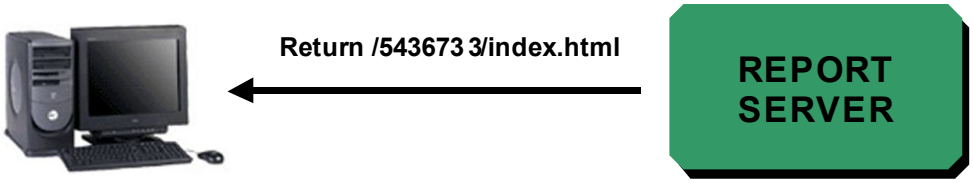
- клиент передает на сервер значения управляющих элементов формы с указанием предыдущего идентификатора сессии (он был записан в файле формы)



- сервер передает полученную информацию в FastReport



- выполнение отчета продолжается и в результате файл результата передается клиенту



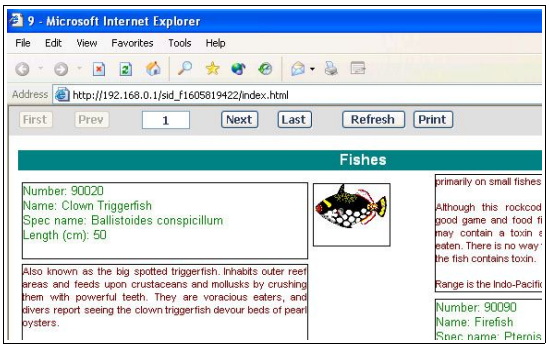
Формат строки запроса к серверу, конфигурирование, ведение журналов, аутентификация и другие, касающиеся работы сервера вопросы будут рассмотрены ниже.

3.2. Поддерживаемые форматы результатов выполнения отчета.

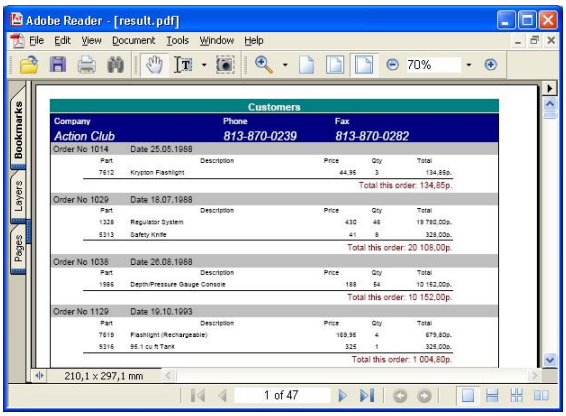
Формат FR3 является внутренним форматом FastReport 3 и представляет собой XML документ. Используется при применении связки компонент TfrxReportServer и TfrxReportClient. Наиболее приспособлен для печати документов, так как в процессе передачи результатов отчета клиенту не производится никаких промежуточных преобразований. В большинстве случаев использование данного формата дает существенную экономию времени и размеров передаваемых файлов (за исключением отчетов с большим количеством внедренных графических изображений высокого разрешения). Применение дополнительных компонент сжатия выходных файлов FastReport (таких как TfrxGZipCompressor) на стороне сервера нежелательно, так как это ведет к значительному снижению производительности сервера и не дает выигрыша в объеме пересылаемой информации, если также в этот момент включено сжатие при передаче информации на сервере и на клиенте (TfrxReportServer.Configuration.Compression := True; TfrxReportClient.Connection.Compression := True).

HTML формат - применяемый в сети Интернет для создании web-сайтов, предназначен для просмотра на экране компьютера. Качественная печать из этого формата затруднительна, поэтому использование его для вывода результатов оправдано исключительно целями предварительного просмотра или

в тех случаях, когда печать документов не играет решающей роли. Данный формат также удобен в случае использования в качестве клиентов обычных web-браузеров (подробно об этом в разделе 5.2). В целях упрощения навигации по результатам отчета, сервер генерирует результат в файле HTML вместе с навигационной панелью, с помощью которой можно листать страницы.



Формат PDF разработан фирмой Adobe для создания документов, предназначенных для последующей печати. Экспорт в этот формат из генератора отчетов отличается большей точностью по отношению ко всем остальным форматам. Для просмотра файлов формата PDF на компьютере должна быть установлена программа Adobe Acrobat Reader. Если установлено свойство TfrxReportServer.PrintPDF := True, то генерация файла в этом формате возможна при просмотре HTML страниц с результатами отчета (при нажатии на кнопку “Print”, расположенную на панели навигатора).



Сервер также поддерживает следующие форматы:

- RTF формат, документ RichText, может быть открыть практически в любом текстовом редакторе;

- XLS и XML – форматы табличного редактора Excel;

- текстовый файл, необходим для создания файлов для последующей печати на матричных принтерах;

- графический файл jpeg.

Набор разрешенных к запросу форматов конфигурируется свойством TfrxReportServer.Configuration.OutputFormats, которое может содержать одно или несколько значений из следующего множества: sfHTM – формат HTML, sfXML – формат XML, sfXLS – формат Excel, sfRTF – формат RichText, sfTXT – текстовый формат, sfPDF – формат Adobe Acrobat, sfJPG – jpeg изображение, sfFRP – формат FastReport 3 (FR3).

Если при запросе явно не указан тип возвращаемого формата, сервер

генерирует результат в формате HTML.

3.3. Синтаксис запросов.

При использовании в качестве клиента обычного web-браузера, либо при запросе файла с помощью компоненты TfrxHTTPClient, вы можете использовать параметры командной строки для получения необходимых результатов.

Наименование запрашиваемого отчета

report=name

name - имя доступного отчета на сервере

Пример: /report=1.fr3

(построить отчет 1.fr3, результат выдать в формате HTML)

Формат возвращаемого файла.

format=name

name - формат возвращаемого файла, доступные форматы: HTM (HTML), XML (таблица xml), XLS (таблица Excel), RTF (документ rich-text), TXT (текстовый файл), PDF(файл Adobe Acrobat), JPG (графический файл jpeg), FRP (внутренний формат построенных отчетов FastReport). По умолчанию установлен формат HTM (HTML). Более подробно форматы данных описаны в разделе 3.2.

Пример: /report=1.fr3&format=TXT

(построить отчет 1.fr3, результат выдать в текстовом формате)

Диапазон страниц отчета

pagerange=value

value - запрашиваемые страницы отчета (для формата FRP данная возможность недоступна). Пример перечисления: 1,3,5-12.

Пример запроса: /report=3.fr3&pagerange=20-25

(построить отчет 3.fr3, страницы с 20 по 25, результат выдать в формате HTML)

Специальные параметры для формата HTML.

multipage=param

Если param установлен в 1, то результат построения отчета будет выдан постранично по одному файлу на страницу, если param установлен в 0 - будет сгенерирована одна результирующая страница, содержащая все страницы отчета. По умолчанию параметр установлен в 1.

Пример: /report=3.fr3&multipage=0

(построить отчет 3.fr3, результат выдать в формате HTML, все страницы отчета разместить на одной HTML странице)

pagenav=param

Установите param в 1 для включения встроенного навигатора страниц. Если установить param в 0, навигатор страниц будет отключен. Для корректного отображения результатов отчета с навигатором страниц необходима поддержка javascript и фреймов в web-браузере, который Вы используете. По умолчанию

параметр установлен в 1.

Пример: /report=9.fr3&multipage=0&pagenav=0 (построить отчет 9.fr3, результат выдать в формате HTML, все страницы отчета последовательно разместить на одной HTML странице, навигатор страниц отключить).

3.4. Передача параметров в отчет.

Если в строке запроса присутствуют другие параметры (не перечисленные выше), то они будут интерпретированы сервером в качестве внутренних переменных FastReport.

Пример:

/report=myreport.fr3&param1=Hello%20World!

(при построении отчета "myreport.fr3" будет создана внутренняя переменная с именем "param1" и значением "Hello World!".

На передаваемые параметры накладываются следующие ограничения:

- все текстовые строки содержащие служебные символы или символы национальных алфавитов должны быть преобразованы в Unicode формат UTF-8 и приведены к стандарту запросов HTTP, для этого можно использовать стандартную функцию Untf8Encode и функцию HTMLCodeStr, описание которой содержится в файле frxServerUtils.pas;

- все параметры передаются как текстовые строки, независимо от их формата, это необходимо учитывать в отчете при применении принятых параметров в качестве операндов в скрипте.

Передача внутренних переменных FastReport, хранящихся в свойстве TfrxReportClient.Variables, при запросе отчета с сервера осуществляется автоматически.

3.5. Внутренние переменные сервера.

При работе сервера в свойстве TfrxReportServer.Variables содержатся автоматически создаваемые и обновляемые переменные:

SERVER_NAME – имя сервера;

SERVER_COPYRIGHT – авторское право;

SERVER_SOFTWARE – версия сервера;

SERVER_LAST_UPDATE – дата модификации программного обеспечения;

SERVER_UPTIME – время непрерывной работы сервера;

SERVER_TOTAL_SESSIONS – количество сессий;

SERVER_TOTAL_REPORTS – количество отчетов;

SERVER_TOTAL_ERRORS – количество ошибок;

SERVER_MAX_SESSIONS – максимальное количество сессий одновременно;

SERVER_MAX_REPORTS - максимальное количество отчетов одновременно.

Пример получения значения переменной SERVER_TOTAL_REPORTS:

Totals := frRepotServer1.Variables.GetValue('SERVER_TOTAL_REPORTS');

В Totals будет помещено значение переменной SERVER_TOTAL_REPORTS.

3.6. Использование произвольных HTML документов.

Сервер отчетов может быть использован как обычный HTTP сервер для отображения различных HTML документов и хранения различных файлов.

HTML документы должны быть размещены в каталоге, путь к которому указан в свойстве TfrxReportServer.Configuration.RootPath. Наименование документа по умолчанию (если при запросе клиент не указал имени файла) необходимо указать в свойстве TfrxReportServer.Configuration.IndexFileName (по умолчанию index.html). Соответственно, такой файл должен существовать в каталоге с HTML документами.

Сервер ограниченно поддерживает технологию SSI (Server Side Includes - директивы включения на стороне сервера). Сервер, обрабатывая HTML документ, исполняет специальные директивы SSI.

Описание директив SSI.

Вставка другого документа в файл.

<!--#include virtual="filename.html" -->

Осуществляет вставку файла с именем filename.html в место нахождения директивы. Путь к файлу считается от пути к каталогу с файлами HTML (RootPath).

Пример использования директивы:

```
<!--#include virtual="header.html" --> || Command line help
<!--#include virtual="top.html" -->
<font face="Tahoma" size="3"><a href="index.html"><b>Back to main page</b></a><b><br>
</b></font>
<hr>
...
```

Вставка переменных сервера.

<!--#echo var="VARIABLE"-->

Осуществляет вставку значения переменной с именем VARIABLE в место нахождения директивы.

Пример использования директивы:

```
...
<tr>   <td align="right" width="200"><b>Uptime:</b></td>
       <td width="300"><!--#echo var="SERVER_UPTIME"--></td></tr>
<tr>   <td align="right"><b>Total sessions:</b></td>
       <td><!--#echo var="SERVER_TOTAL_SESSIONS"--></td></tr>
<tr>   <td align="right"><b>Total reports:</b></td>
       <td><!--#echo var="SERVER_TOTAL_REPORTS"--></td></tr>
<tr>   <td align="right"><b>Max sessions:</b></td>
       <td><!--#echo var="SERVER_MAX_SESSIONS"--></td></tr>
<tr>   <td align="right"><b>Max reports:</b></td>
       <td><!--#echo var="SERVER_MAX_REPORTS"--></td></tr>
...
```

Применение директив SSI при разработке сайта может существенно облегчить работу при грамотном их применении. Пример сайта с применением SSI можно посмотреть в папке “\FastReport 3 CS\Demo\Server\htdocs”.

3.7. Журналы доступа, ошибок и служебной информации.

При установке свойства TfrxReportServer.Configuration.Logging := True, сервер будет осуществлять ведение журналов в папке, путь к которой указан в свойстве TfrxReportServer.Configuration.LogPath.

Всего поддерживается 5 журналов:

- журнал запросов “access.log” - содержит информацию о дате, времени, идентификаторе сессии, IP адресе клиента и строке запроса, фрагмент журнала:

```
10/26/2004 23:56:19 sid_f1672494035    192.168.0.2  result?report=3.fr3
10/26/2004 23:56:23 sid_f1340767011    192.168.0.2  sid_f1672494035/index.html
10/26/2004 23:56:23 sid_f1949776310    192.168.0.2  sid_f1672494035/index.nav.html
10/26/2004 23:56:23 sid_f1150188690    192.168.0.2  sid_f1672494035/index.1.html
```

- журнал регистрации наименований клиентов “agent.log”, содержит информацию о дате, времени, IP адресе клиента и его наименовании, фрагмент журнала:

```
10/26/2004 23:56:19 192.168.0.2 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
10/26/2004 23:56:23 192.168.0.2 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
10/26/2004 23:56:23 192.168.0.2 Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

- журнал регистрации ссылок “referer.log”, содержит информацию о дате, времени, IP адресе клиента и ссылке, откуда был произведен запрос документа, фрагмент журнала:

```
10/26/2004 23:56:19 192.168.0.2  http://192.168.0.1/
10/26/2004 23:56:23 192.168.0.2  http://192.168.0.1/
10/26/2004 23:56:23 192.168.0.2  http://192.168.0.1/sid_f1672494035/index.html
```

- журнал регистрации ошибок “error.log”, регистрирует возникающие ошибки, как по вине клиента, так и из-за критических ситуаций, возникающих на сервере, фрагмент журнала:

```
10/25/2004 13:30:52 192.168.0.2  588864044016/index.1.html document not found
10/26/2004 0:03:11  192.168.0.2  Software caused connection abort.(10053)
10/26/2004 0:43:42  192.168.0.2  Connection reset by peer.(10054)
```

- журнал работы сервера “server.log”, отражает моменты запуска и останова, а также сводную статистическую информацию о работе сервера, обновляется в момент запуска и останова сервера, фрагмент журнала:

```
10/25/2004 19:38:15 Started
10/25/2004 19:38:15 HTTP server created
10/25/2004 19:58:57 HTTP server closed
10/25/2004 19:58:57 Stopped
Uptime: 0 days 0 hours 20 minutes 42 seconds
Total sessions: 654
Total reports: 327
Total errors: 0
Max sessions: 84
Max reports: 42
```

Необходимо следить за своевременным архивированием журналов для экономии дискового пространства сервера.

3.8. Аутентификация.

Сервер поддерживает HTTP аутентификацию, для ее включения нужно задать имя пользователя и пароль в свойствах TfrxReportServer.Configuration.Login и TfrxReportServer.Configuration.Password. Если свойства установлены, то при диалоге с клиентом в принятом от него заголовке запроса ожидаются соответствующие данные аутентификации, согласно описанию протокола HTTP (RFC 2068 [2]). Если клиентом получено сообщение с кодом ответа 401 “Unauthorized”, он должен повторить запрос с корректными именем и паролем пользователя. Web-браузер в таком случае выдаст диалоговую форму с соответствующими полями ввода.

3.9. Ограничение доступа к серверу по IP адресу.

Сервер поддерживает ограничение доступа по IP адресу клиента.

Свойство TfrxReportServer.DenyIP должно содержать перечень IP адресов, которым запрещено подключение к серверу.

Свойство TfrxReportServer.AllowIP должно содержать перечень IP адресов, которым разрешено подключение к серверу.

Каждый из списков должен содержать в одной строке один IP адрес.

Пример списка:

```
192.168.0.10
192.168.0.12
192.168.0.54
```

Если списки DenyIP и AllowIP пусты - будут допущены к подключению все клиенты.

Если список DenyIP пуст, а список AllowIP содержит конкретные IP адреса, то только они будут допущены к подключению.

Если список DenyIP имеет записи и IP адрес подключаемого клиента не соответствует им, будет осуществлена проверка на наличие этого IP адреса в списке AllowIP.

Маски IP адресов не поддерживаются.

Примеры применения списков разрешения/запрета.

1. Необходимо разрешить подключения только с локальной машины, к примеру для целей взаимодействия сервера FastReport с другим работающим HTTP сервером.

Список AllowIP должен содержать одну строку:

```
127.0.0.1
```

Список DenyIP должен быть пуст.

2. Необходимо разрешить доступ с адресов 192.168.0.2 – 192.168.0.6 и запретить доступ с остальных адресов.

Список AllowIP должен содержать строки:

```
192.168.0.2
192.168.0.3
192.168.0.4
192.168.0.5
192.168.0.6
```

Список DenyIP должен быть пуст.

3. Необходимо запретить доступ с адресов 192.168.0.8 – 192.168.0.10 и запретить доступ с остальных адресов.

Список AllowP должен быть пуст.

Список DenyIP должен содержать строки:

```
192.168.0.8
192.168.0.9
192.168.0.10
```

3.10. Варианты подключений к базам данных.

Основная рекомендуемая схема работы с базами данных выглядит следующим образом:

- на одной из форм вашего приложения создать сессию для подключения к базе данных;

- в отчетах FastReport на диалоговых формах создать и настроить компоненты SQL запросов с использованием подключения по умолчанию, созданного в приложении.

При таком решении будет обеспечена необходимая многопоточность при единственном подключении к серверу баз данных. Возможно, некоторые компоненты доступа к базам данных не поддерживают одновременных запросов через одно подключение, в таком случае нужно будет применить способ описанный ниже.

Другим вариантом будет использование в каждом отчете собственных компонент доступа к базам данных, в таком случае будет возможно подключение к нескольким базам из разных отчетов FastReport. При отсутствии необходимости работы с несколькими базами данных, такой подход не оправдан, так-как будет каждый раз осуществляться подключение к базе данных при построении отчета FastReport (в некоторых СУБД на некоторых платформах это может занимать достаточно большой промежуток времени).

Процедура разработки отчетов с использованием встроенных компонент доступа подробно отражена на странице 134 книги “FastReport 3 – Руководство пользователя.”[9].

Настоятельно не рекомендуется использование компонент BDE для работы с базами данных из-за большого количества проблем связанных с нестабильной работой этих компонент в нескольких потоках.

3.11. Использование кеша отчетов.

Кеширование отчетов позволяет добиться высокой производительности за счет сохранения промежуточных результатов во временных файлах сервера. В зависимости от конфигурации сервера, после построения результат может быть помещен в кеш и в соответствующей внутренней таблице сервера будет сделана запись о имени отчета и полученных от клиента параметрах, при которых этот отчет был построен. По истечении определенного времени результат отчета будет удален из кеша. Если за это время поступит запрос от клиента *с таким же именем отчета и теми же значениями параметров*, то ему будет немедленно возвращен ответ на основе сохраненного в кеше результата в том формате, который запросил клиент. В таком случае сервер потратит время только на преобразование промежуточного результата в запрошенный формат без построения отчета. Это дает большой прирост производительности.

В зависимости от решаемых сервером задач для каждого отчета можно назначить индивидуальное время хранения в кеше. Время выбирается администратором сервера с точки зрения актуальности того или иного отчета

после прохождения определенного промежутка времени. К примеру, годовой отчет о деятельности предприятия может храниться в кеше достаточно долго, т.к. содержит информацию за большой промежуток времени и она не устареет очень быстро. Отчет о состоянии склада большой торговой организации напротив должен быть актуален за малый промежуток времени и поэтому должен храниться в кеше малое время.

Параметры кеширования отчетов:

TfrxServer.Configuration.ReportCaching – включение возможности кеширования, если True то кеширование работает;

TfrxServer.Configuration.ReportCachePath – путь к папке хранения временных файлов кеша отчетов;

TfrxServer.Configuration.DefaultCacheLatency – время хранения файла в кеше в секундах, по умолчанию 300 секунд.

В файле конфигурации эти параметры выглядят следующим образом:

```
[ReportsCache]
; enable caching of the reports with same params
Enabled=1
; path to chache folder
CachePath=.\cache\
; dafault delay for cache of the report results in seconds
DefaultLatency=300
```

Дополнительная секция конфигурационного файла сервера [ReportsLatency] служит для настройки времени хранения в кеше результатов того или иного отчета:

```
[ReportsLatency]
; cache delay for the 1.fr3 report in seconds
1.fr3=10
; cache delay for the 1.fr3 report in seconds
2.fr3=20
; add below the any reports for the custom cache delay setup
```

Правильное конфигурирование параметров кеширования уменьшит время обслуживания клиентов и снизит общую нагрузку на сервер.

3.12. Увеличение производительности сервера.

Для увеличения производительности сервера можно использовать следующие рекомендации:

- использовать кеширование отчетов, подробности описаны в п. 3.11.

- Не использовать компоненты сжатия (компрессоры) выходных файлов FastReport (таких как TfrxGZipCompressor). Это значительно замедляет работу сервера.

- Оптимизировать SQL запросы к базе данных. В некоторых случаях время выполнения запроса к базе данных может превышать время построения отчета на сервера отчетов.

- При разработке отчетов избегать внедрений изображений с высоким разрешением – это увеличит не только время построения отчета, но и объем передаваемых по сети данных.

- В отчетах применять по возможности простые скрипты для обработки каких-либо событий.

- В качестве клиентского приложения использовать компоненты FastReport 3 и формат результата FP3, во многих случаях это позволит уменьшить время построения отчета и объем передаваемых данных (если в отчетах нет картинок с высоким разрешением).

- При разработке отчетов следовать рекомендациям, перечисленным в п. 4.2.

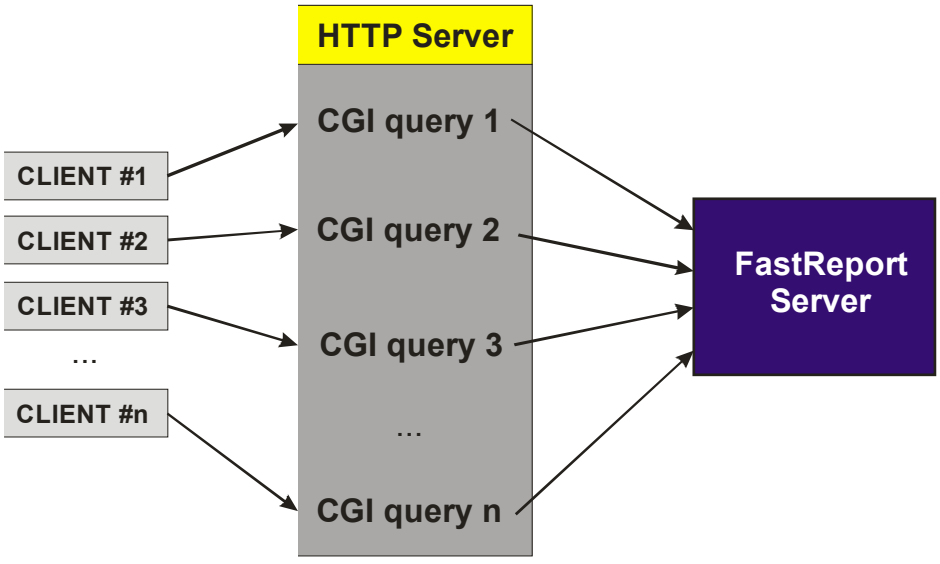
- Отключить подсчет контрольных сумм передаваемых файлов. Свойство TfrxReportServer.Configuration.MIC := False.

- Увеличить объем оперативной памяти и мощность процессора на компьютере, где установлен сервер отчетов.

3.13. Использование сервера FastReport совместно с HTTP серверами Apache, IIS и другими.

Для использования уже существующих решений на базе других HTTP серверов возможна их интеграция с сервером FastReport при помощи механизма CGI. Это дает некоторые преимущества по отношению к использованию встроенного HTTP сервера FastReport: отчеты могут быть встроены в уже работающую систему (сайт), HTTP сервер и сервер отчетов могут работать на разных компьютерах, возможно использование SSL шифрования для работы с HTTP сервером (данная возможность пока отсутствует в HTTP сервере FastReport).

При таком подходе CGI будет служить промежуточным звеном для передачи запроса на сервер FastReport, получения результатов от сервера отчетов и возврата результатов клиенту.



В папке Demos\ClientServer\CGI находится пример CGI приложения, которое выполняет эти функции.

Для исользования CGI необходимо:

- скопировать скомпилированный файл fastreport.exe в папку /cgi-bin HTTP сервера;

- сконфигурировать HTTP сервер (Apache, IIS или другой) для выполнения CGI приложений, более подробную информацию об этом можно получить в руководстве по настройке используемого HTTP сервера;

Если HTTP и FastReport сервер будут работать на одном компьютере:

- при использовании HTTP сервером TCP/IP порта 80 (по умолчанию), необходимо сконфигурировать FastReport сервер для использования порта 8097 (этот порт используется CGI приложением по умолчанию, если файл конфигурации CGI отсутствует), для этого в конфигурационном файле сервера отчетов нужно изменить параметр Port=8097 (возможно использование любого другого TCP/IP порта, но в этом случае необходимо использование файла настройки CGI приложения – см. ниже);

Если HTTP и FastReport сервер будут работать на разных компьютерах:

- создать в папке /cgi-bin файл конфигурации CGI приложения с именем fastreport.ini, в котором будут храниться настройки CGI клиента для доступа к серверу отчетов:

```
[REPORTSERVER]
Host=192.168.0.34
Port=80
```

Этот файл конфигурации можно использовать в случае совместной работы сервера отчетов и HTTP сервера на одном компьютере для указания другого порта для сервера отчетов.

- запустить FastReport сервер и проверить совместную работу с HTTP сервером.

Запросы к серверу отчетов должны иметь следующий вид:

http://127.0.0.1/cgi-bin/fastreport.exe?report=67.fr3&multipage=0&pagenav=0

Подробно синтаксис строки запросов к серверу отчетов рассмотрен в п. 3.3. Обратите внимание что ключевое слово “result” в данном случае просто заменено конструкцией “cgi-bin/fastreport.exe”.

Внимание: для ограничения прямого доступа к серверу отчетов с клиентских компьютеров необходимо в файле конфигурации FastReport сервера allow.conf прописать IP адрес HTTP сервера, на котором работает CGI приложение (127.0.0.1 или другой адрес).

4. Разработка отчетов.

Разработка отчетов FastReport очень подробно описана в книге “FastReport 3 – Руководство пользователя.” [9]

4.1. Ограничения при разработке отчетов для FastReport Client/Server.

При проектировании тестовых отчетов необходимо помнить о следующих ограничениях FastReport Enterprise:

- при использовании диалоговых форм в отчетах, нельзя применять никакие внутренние скриптовые обработчики внутри диалоговых форм FastReport;

- нельзя использовать обработчики событий TfrxReportClient для реализации таких возможностей, как пользовательские функции и пр.;

- нежелательно использование общих компонент доступа к базам данных через компоненты TfrxDataSet.

4.2. Рекомендации по разработке отчетов, результат которых будет представлен в табличных форматах.

Очень многие форматы используют табличное представление данных. Из применяемых в сервере отчетов к ним относятся форматы HTML, XLS, XML, RTF.

Никаких пересечений или наслоений ячеек в подобных форматах недопустимо (если брать в рассмотрение именно табличную разметку, это касается HTML и RTF), в отличии от свободы в процессе разработки шаблона отчета в дизайнере FastReport. Фильтры экспорта, как правило, максимально учитывают эти требования при переносе объектов из отчета FastReport в результирующий файл. Это реализуется при помощи специальных алгоритмов учета пересечений объектов и оптимального их расположения. В местах пересечений объектов возникают новые столбцы и строки в результирующей таблице. Это необходимо для сохранения точного позиционирования переносимых объектов FastReport. Большое количество пересекающихся объектов в отчете приводит к росту числа столбцов и строк в таблице, что усложняет ее дальнейшее использование в и замедляет процесс экспорта.

В процессе разработки шаблона отчета помните об этом, если Вы хотите в последующем экспортировать свои отчеты в какой-либо из форматов, отличный от внутреннего формата FastReport.

Избежать перекрытия ячеек помогут инструменты выравнивания текстовых объектов по сетке. Проследите за тем, чтобы было включено выравнивание по сетке. Для упрощения выравнивания можно увеличить шаг сетки.

При создании таблиц в отчетах проследите чтобы границы соседних ячеек соприкасались друг с другом. Важно, чтобы ячейки не пересекались. Алгоритм фильтра экспорта сделает отсечение ячеек, но результат экспорта может быть далек от желаемого (вы увидите не совсем то, что хотели).

Располагайте объекты так, чтобы они находились на одной линии, как по вертикали, так и по горизонтали. В этом могут помочь выносные линии.

Применение этих простых правил на практике поможет Вам создать отчет, который будет прекрасно выглядеть после экспорта в любой из форматов, которые используют табличную разметку для представления данных.

5. Клиент.

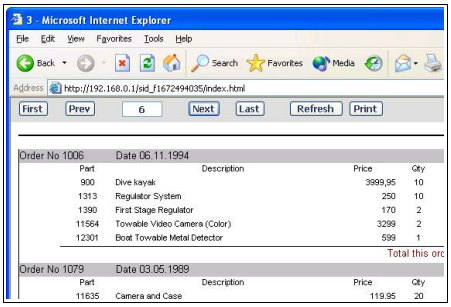
Можно определить два основных типа клиентских приложений для сервера отчетов FastReport, это приложения на основе компонент TfrxReportClient и автономные HTTP клиенты, такие как web-браузеры.

5.1. Клиент на основе FastReport 3.

TfrxReportClient специально предназначен для запроса отчетов с передачей параметров запроса (переменных) из клиентской программы, получения файла в формате FP3, предварительного просмотра и его последующей печати на принтере клиента. При желании разработчик может реализовать возможность экспорта полученного результата в один из форматов данных, поддерживаемых FastReport 3. Это достигается путем добавления необходимых компонент экспорта в проект клиентской программы. В большинстве случаев это решение является оптимальным исходя из позиции минимизации временных затрат и уменьшения сетевого трафика. Нагрузка на сервер при запросах с таких клиентов также существенно меньше.

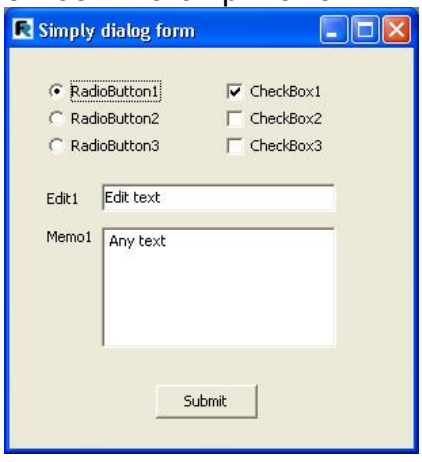
5.2. Другие возможные клиенты.

Сервер отчетов FastReport представляет широкие возможности вы выборе клиентской программы, благодаря использованию стандартного протокола передачи данных HTTP. Клиентом может выступать практически любой web-браузер, поддерживающий такие возможности, как javascript, таблицы и фреймы.

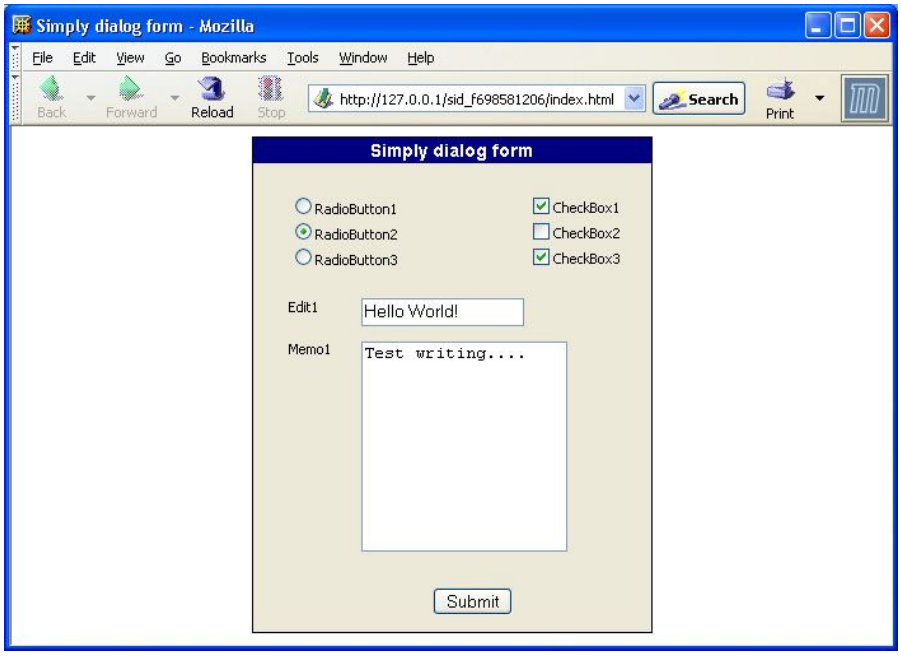


При использовании в отчетах FastReport диалоговых форм, они будут динамически преобразованы в web-формы и сервер будет ожидать результатов заполнения такой формы для продолжения исполнения отчета.

Так выглядит диалоговая форма при исполнении отчета FastReport под управлением обычного приложения.



Таже форма, динамически преобразованная в web-форму при запросе с web-браузера Mozilla.



6. Перевод существующих приложений на платформу Client/Server.

При переводе уже разработанных приложений на платформу клиент-сервер, необходимо придерживаться следующих рекомендаций:

- Четко определить схему взаимодействия клиентской и серверной программ.

- Учесть все требования и рекомендации, перечисленные в разделах 4.1. и 4.2 настоящего руководства.

- Если использовались обработчики событий TfrxReport, постараться реализовать построение отчета без их использования.

- Для работы с базами данных учесть требования, перечисленные в разделе 3.10 настоящего руководства.

- Учесть рекомендации по обеспечению информационной безопасности (раздел 8 настоящего руководства).

7. Примеры.

7.1. Пример простейшего Client/Server приложения.

Для ознакомления с методикой применения компонент FastReport Enterprise можно использовать идущие в комплекте поставки демонстрационные примеры, находящиеся в папке \FastReport 3 CS\Demo. В комплекте поставляется демонстрационная база данных формата Microsoft Access (MDB), содержащая знакомые всем Delphi/C++Builder разработчикам таблицы из комплекта DBDEMOS. В примерах охвачено большинство моментов применения поставляемых компонент.

7.1.1. Серверная часть.

Все исходные тексты примера находятся в папке \FastReport 3 CS\Demo\Server.

На главное форме приложения размещена необходимая для функционирования сервера компонента класса TfrxReportServer с именем Serv.

Для подключения к базе данных использованы компоненты TADOConnection, и интерфейс для работы с ADO из FastReport TfrxADOComponents.

Другие компоненты из комплекта FastReport включены для корректного выполнения отчетов, содержащих диалоговые формы, диаграммы, ширх-коды и пр.

Для удобства конфигурация сервера хранится в файле, который можно отредактировать встроенным редактором.

Содержимое файла конфигурации server.conf:

```
[Server]
; TCP/IP port for HTTP server
Port=80
; report session timeout in seconds
SessionTimeOut=600
; client connection timeout in seconds
SocketTimeOut=600
; index page filename
IndexFileName=index.html
; path to folder with logs
LogPath=.\logs\
; enable of log writing
WriteLogs=1
; maximum log files in history
MaxLogFles=5
; maximum log file size
MaxLogSize=1024
; path to folder with the reports (*.fr3)
ReportPath=.\reports\
; public document folder for documents and results
RootPath=.\htdocs\
; disable of the caching document by the web browser
NoCacheHeader=1
; GZIP compression enable
```

```
Compression=1
; MD5 message integrity check
MIC=1
; user login
Login=
; user password
Password=

[ReportsCache]
; enable caching of the reports with same params
Enabled=1
; path to chache folder
CachePath=.\cache\
; dafault delay for cache of the report results in seconds
DefaultLatency=300

[ReportsLatency]
; cache delay for the 1.fr3 report in seconds
1.fr3=10
; cache delay for the 1.fr3 report in seconds
2.fr3=20
; add below the any reports for the custom cache delay setup
```

Наименования полей файла конфигурации соответствуют наименованиям свойств TfrxReportServer.Configuration.

В файлах allow.conf и deny.conf хранятся списки разрешенных и запрещенных IP адресов соответственно.

Файл базы данных находится в папке database.

В главном модуле определены константы с именами файлов конфигурации:

```
const
  CONFIG_FILE = 'server.conf';
  ALLOW_FILE =  'allow.conf';
  DENY_FILE =   'deny.conf';
```

После старта программы производится подключение к базе данных через интерфейс MicrosoftJet OLE DB.

В переменные ConfFile, AllowFile, DenyFile помещаем путь к расположению конфигурационных файлов (каталог запуска программы):

```
  AppPath := ExtractFilePath(Application.ExeName);
  ConfFile := AppPath + CONFIG_FILE;
  AllowFile := AppPath + ALLOW_FILE;
  DenyFile := AppPath + DENY_FILE;
```

Загружаем файлы конфигурации в компоненту Serv:

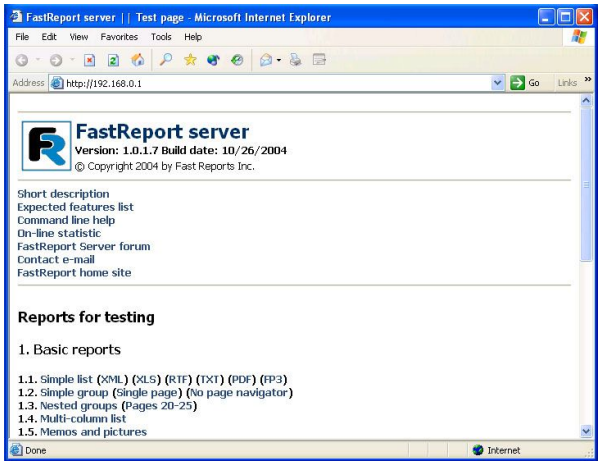
```
    Serv.Configuration.LoadFromFile(ConfFile);
    Serv.AllowIP.LoadFromFile(AllowFile);
    Serv.DenyIP.LoadFromFile(DenyFile);
```

И запускаем сервер:

```
    Serv.Open;
```

Если все прошло без проблем – сервер переходит в режим ожидания запросов от клиентов. После запуска приложения, можно убедиться в его работоспособности, запустив любой web-браузер и набрав в строке адреса строку http://127.0.0.1 – откроется демонстрационная страница со ссылками на примеры отчетов:

Отчеты можно открыть на редактирование в дизайнере FastReport, компонента которого также находится на главной форме, вызов дизайнера осуществляется при нажатии на кнопку Designer:




```
OpenDialog1.InitialDir := Serv.Configuration.ReportPath;
if OpenDialog1.Execute then
begin
  frReport1 := TfrxReport.Create(nil);
  frReport1.LoadFromFile(OpenDialog1.FileName);
  frReport1.DesignReport;
  frReport1.Free;
end;
```

7.1.2. Клиентская часть.

Все исходные тексты примера находятся в папке \FastReport 3 CS\Demo\Client\Simple.

Пример демонстрирует способы работы с компонентой TfrxReportClient, и передачей параметров на сервер.

На форме расположены компоненты frxServerConnection1 типа TfrxServerConnection, frxReportClient1 типа TfrxReportClient, поля ввода “Host”, “Port”, “Login”, “Password”, “RepName”, “Param1”, “Param1Value”, “Param1”, “Param1Value” типа TEdit.

Перед стартом программы нужно запустить серверную программу, рассмотренную в разделе 7.1.1.

Если запустить программу и ввести в поле “Report Name”, к примеру “1.fr3”, то при нажатии на кнопку “Show report” выполняется следующий код:

```
frxServerConnection1.Host := Host.Text;
frxServerConnection1.Port := StrToInt(Port.Text);
frxServerConnection1.Login := Login.Text;
frxServerConnection1.Password := Password.Text;
frxReportClient1.LoadFromFile(RepName.Text);
if Length(Param1Value.Text) > 0 then
  with frxReportClient1.Variables.Add do
  begin
    Name := Param1.Text;
    Value := Param1Value.Text;
  end;
if Length(Param2Value.Text) > 0 then
  with frxReportClient1.Variables.Add do
```

```
    begin
      Name := Param2.Text;
      Value := Param2Value.Text;
    end;
  if frxReportClient1.PrepareReport then
    frxReportClient1.ShowPreparedReport;
  Memo1.Lines.AddStrings(frxReportClient1.Errors);
```

При успешном исполнении запроса, выводится окно с предварительным просмотром отчета, в противном случае в Memo1 помещаются строки с кодами и наименованиями ошибок.

7.1.3. Клиентская часть с несколькими потоками.

Все исходные тексты примера находятся в папке \FastReport 3 CS\Demo\Client\Advanced.

Этот пример содержит функциональную часть аналогичную ранее рассмотренному, за исключением более продвинутого журнала действий клиента, полей для ввода информации о прокси-сервере и модуля многопоточного тестирования сервера.

Рассмотрим использование компоненты TfrxReportClient в протоке подробнее.

Для реализации многопоточности объявлен класс:

```
  TfrxClientTestThread = class
(TThread)
  protected
    procedure Execute; override;
  private
    CountRep: Integer;
    ErrorsCount: Integer;
    Log: TMemo;
    ThreadID: Integer;
    procedure AppendLog;
    procedure FinishLog;
  public
    Report: TfrxReportClient;
    constructor Create(C: TfrxServerConnection; RepName: String;
        Id: Integer; Rep: Integer; L: TMemo);
    destructor Destroy; override;
  end;
```

В конструкторе класса TfrxClientTestThread выполняется создание экземпляра TfrxReportClient и инициализация переменных потока:

```
constructor TfrxClientTestThread.Create(C: TfrxServerConnection; RepName:
String;
    Id: Integer; Rep: Integer; L: TMemo);
begin
  inherited Create(True);
  FreeOnTerminate := False;
```

```
  ErrorsCount := 0;
  ThreadId := Id;
  CountRep := Rep;
  Log := L;
  Report := TfrxReportClient.Create(nil);
  Report.EngineOptions.ReportThread := Self;
  Report.Connection := C;
  Report.ReportName := RepName;
  Resume;
end;
```

Метод TfrxClientTestThread.Execute выполняет запрос на сервер CountRep раз без показа результатов отчета, но с анализом количества ошибок исполнения, вся информация выводится в Memo1 методами AppendLog и FinishLog:

```
procedure TfrxClientTestThread.Execute;
var
 i: Integer;
begin
  inherited;
  for i := 1 to CountRep do
  begin
    if Terminated then break;
    Report.PrepareReport;
    if not Terminated then
    begin
      Synchronize(AppendLog);
      ErrorsCount := ErrorsCount + Report.Errors.Count;
    end;
  end;
  Synchronize(FinishLog);
end;
```

Перед стартом программы нужно запустить серверную программу, рассмотренную в разделе 7.1.1.

При нажатии на кнопку “Thread test” выполнится следующий код:

```
procedure TMainForm.TestBtnClick(Sender: TObject);
var
  i, j, k: Integer;
  Thread: TfrxClientTestThread;
begin
  frxServerConnection1.Host := Host.Text;
  frxServerConnection1.Port := StrToInt(Port.Text);
  frxServerConnection1.Login := Login.Text;
  frxServerConnection1.Password := Password.Text;
  frxServerConnection1.Compression := Compression.Checked;
  if (Length(ProxyHost.Text) > 0) then
  begin
    frxServerConnection1.PorxyHost := ProxyHost.Text;
    frxServerConnection1.PorxyPort := StrToInt(ProxyPort.Text);
  end;
  ClearThreads;
  Memo1.Lines.Add('Start test');
  j := StrToInt(Threads.Text);
  k := StrToInt(Rep.Text);
  for i := 1 to j do
  begin
    Thread := TfrxClientTestThread.Create(frxServerConnection1,
       ReportsList[ListBox1.ItemIndex], i, k, Memo1);
    ThreadList.Add(Thread);
  end;
end;
```

Будет создано количество потоков, заданное в поле ввода Threads.Text,

ссылки на потоки будут сохранены в списке ThreadList, для последующей очистки методом ClearThreads.

При тестировании данного примера, необходимо помнить о том, что большое количество одновременно созданных потоков (больше 100) может вызвать сильную загрузку центрального процессора и высокое потребление оперативной памяти при запуске клиентской и серверной программ на одном компьютере.

8. Важные замечания по безопасности.

Рекомендации по обеспечению информационной безопасности:

1. При использовании доступа к серверу отчетов FastReport, работающего под управлением Microsoft Windows, через сеть Internet, настоятельно рекомендуется использование промежуточного сетевого экрана (firewall), на котором будет открыт сетевой порт, используемый сервером отчетов.

2. Обязательно используйте аутентификацию клиентских программ при доступе к серверу (раздел 3.8).

3. При работе клиент-сервер приложений, имеющих четкую аудиторию в пределах локальной сети, рекомендуется конфигурирование доступа на уровне разрешенных/запрещенных к обслуживанию IP адресов (раздел 3.9).

4. При использовании сервера отчетов в локальной сети имеющей шлюзы в интернет, рекомендуется занести IP адреса этих шлюзов в список запрещенных к обслуживанию адресов (раздел 3.9).

5. При использовании отчетов со встроенными компонентами доступа к базам данных не предавайте параметры аутентификации к базе данных с клиентского приложения, лучше это сделать непосредственно на сервере.

6. Располагайте в папке хранения отчетов, только те отчеты, которые должно выполнять серверное приложение.

7. В папке хранения HTML документов не располагайте никаких важных файлов, доступ к которым может быть нежелателен.

8. При разработке серверного приложения сразу создайте разные каталоги для ведения журналов, хранения отчетов, хранения HTML документов. Путь к этим каталогам укажите в настройках серверной компоненты.

9. При обнаружении проблем с безопасностью компонент FastReport Enterprise сообщите разработчикам продукта.

9. Дополнительные источники информации.

1. Braden, R., "Requirements for Internet hosts - application and support", STD 3, RFC 1123, IETF, October 1989.
2. Berners-Lee, T., Fielding, R., and H. Frystyk, "Hypertext Transfer Protocol – HTTP/1.1" RFC 2068, January 1997.
3. Franks, J., Hallam-Baker, P., Hostetler, J., Leach, P., Luotonen, A., Sink, E., and L. Stewart, "An Extension to HTTP : Digest Access Authentication", RFC 2069, January 1997.
4. Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.
5. Meyers, J., and M. Rose "The Content-MD5 Header Field", RFC 1864, Carnegie Mellon, Dover Beach Consulting, October, 1995.
6. Deutsch, P., "GZIP file format specification version 4.3." RFC 1952, Aladdin Enterprises, May 1996.
7. Tzyganenko, A., “FastReport 3 – Developer manual.” Fast Reports Inc., September 2004. http://www.fast-report.com/pbc_download/DeveloperManual-en.pdf
8. Tzyganenko, A., “FastReport 3 – Programmer manual.” Fast Reports Inc., September 2004. http://www.fast-report.com/pbc_download/ProgrammerManual-en.pdf
9. Tzyganenko, A., “FastReport 3 – User manual.” Fast Reports Inc., September 2004. http://www.fast-report.com/pbc_download/UserManual-en.pdf

10. Связь с разработчиками.

Мы ждем ваших пожеланий и замечаний о совершенствовании и развитии FastReport Enterprise.

e-mail:	fediachov@fast-report.com
news:	http://fast-report.com/en/support/newsgroups.php
forum:	http://www.fast-report.com/en/forum/index.php?s=8
web site:	http://www.fast-report.com