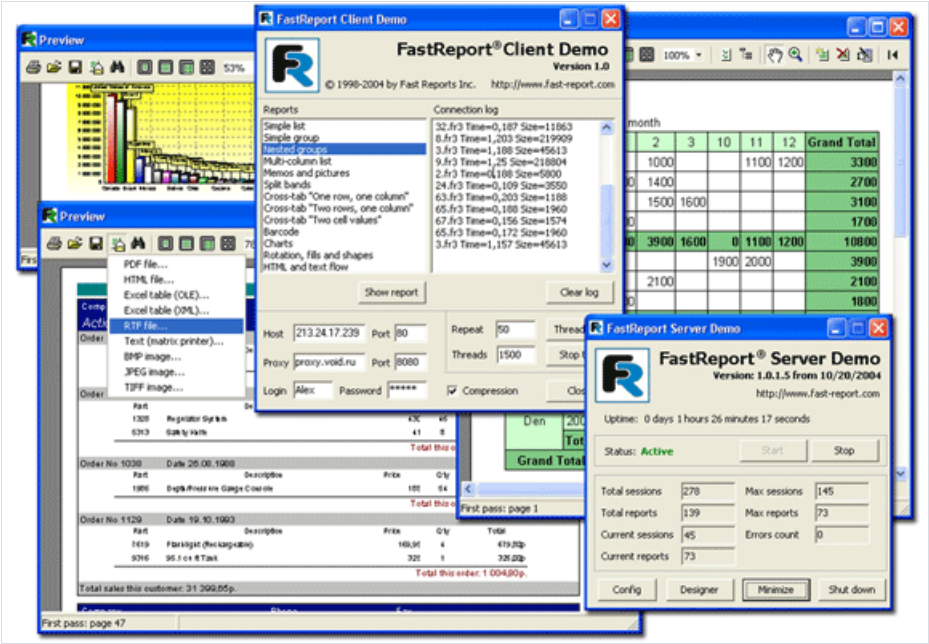
# Руководство по веб-разработке FastReport VCL

Версия 2022.2

# Введение

Данное руководство содержит информацию о дополнительных веб-компонентах библиотеки FastReport VCL. Данное расширение позволяет строить отчеты по технологии клиент-сервер с использованием стандартных компонентов FastReport VCL и дополнительных компонентов, которые предназначены для организации взаимодействия клиента и сервера.



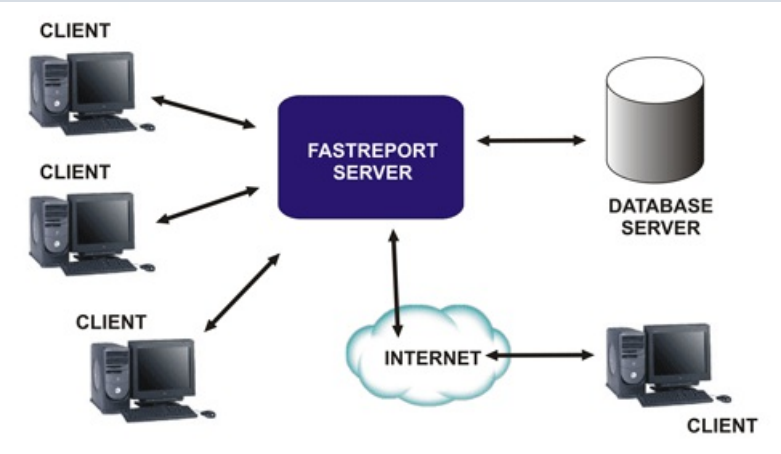

В руководстве описана структура клиентских и серверных компонентов, их свойства и методы, а также архитектура сервера отчетов и принципы его функционирования. Кроме того, даются рекомендации по оптимизации и использованию новых возможностей в уже существующих и новых приложениях.

Опытным пользователям FastReport VCL будут интересны рекомендации по увеличению скорости работы серверных компонентов, оптимизации отчетов для их корректного экспорта в различные табличные форматы, применению правил информационной безопасности для защиты приложения от несанкционированного доступа.

Мы постоянно совершенствуем компоненты FastReport VCL Enterprise. Поэтому есть вероятность, что некоторые возможности не будут упомянуты в данном руководстве. Описания всех изменений будут обязательно включены в следующую версию данного руководства.

# FastReport VCL Enterprise - клиент-серверные инструменты отчётности

Технология "клиент-сервер" основана на взаимодействии между клиентским приложением (которое запрашивает, анализирует и отображает запрошенную информацию) и серверным приложением, выполняющим основную работу, связанную с различными сложными вычислениями.




Существует несколько серьезных преимуществ использования технологии клиент-сервер в ваших приложениях:

- низкие требования к аппаратному обеспечению клиентских ПК;
- снижение сетевого трафика за счет уменьшения объема информации, передаваемой между клиентским приложением и сервером баз данных;
- простота управления системой;
- более высокий уровень защиты информации.

Однако технология клиент-сервер имеет ряд существенных недостатков:

- высокие требования к аппаратному обеспечению компьютера, используемого в качестве сервера;
- определенные трудности при разработке клиент-серверных приложений.

При разработке FastReport VCL Enterprise мы учли все основные требования, предъявляемые к клиент-серверным приложениям.

В FastReport VCL Enterprise есть следующие возможности:

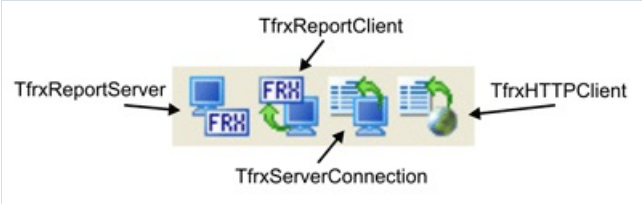
- можно запускать любые отчеты на стороне сервера по запросу клиента, без необходимости прямого подключения клиента к серверу базы данных;
- можно управлять несколькими клиентскими запросами одновременно в отдельных потоках; это минимизирует время отклика сервера;
- поскольку мы используем протокол передачи гипертекста (HTTP, RFC 2068 [2]), вы можете использовать различные существующие приложения, такие как веб-браузеры (Edge, Mozilla Firefox, Safari, Opera и др.), прокси-серверы, веб-серверы (IIS, Apache, Nginx и др.), вместе с FastReport VCL Enterprise без каких-либо дополнительных требований;
- сервер использует алгоритмы сжатия данных (GZip, RFC 1952 [6]). Это уменьшает сетевой трафик и увеличивает вычислительную мощность клиент-сервера;
- сервер использует алгоритм MD5 для MIC (Message Integrity Checksum, RFC 1321 [4], RFC 1864 [5]), что повышает целостность данных;
- сервер поддерживает совместимость с файлами отчетов FastReport VCL (с некоторыми ограничениями),

что позволяет легко перепроектировать приложение для использования технологии клиент-сервер;

- сервер может работать как автономное серверное приложение (без необходимости применения IIS, Apache или других веб-серверов);
- сервер можно использовать как простой HTTP-сервер для хранения и отображения любых HTML-документов;
- применение технологии Server Side Include (SSI) позволяет использовать сервер как движок для вашего web-сайта;
- сервер поддерживает ведение журналов соединений, журналов ошибок и/или любой дополнительной системной информации, что позволяет вести учет работы, быстро отслеживать ошибки и попытки несанкционированного доступа;
- сервер поддерживает использование аутентификации и списков IP "разрешить/запретить", что позволяет ограничить доступ к серверу;
- в одном отчете можно одновременно использовать несколько соединений с базой данных;
- можно использовать клиентские компоненты FastReport для взаимодействия клиентского приложения с сервером. Вы также можете использовать любой web-браузер;
- в ваших отчетах могут присутствовать диалоговые формы, которые будут использоваться для ввода значений перед запуском отчета;
- поддерживаемые форматы готовых отчетов: HTML, PDF, RTF, XML, XLS, JPEG и Text;
- вы можете использовать несколько режимов отображения подготовленного отчета в веб-браузере: одностраничный документ, разделенный на страницы с навигатором страниц.

# Компоненты FastReport Enterprise Edition

После установки пакетов FastReport VCL Enterprise будет доступна закладка "FastReport Client/Server" в палитре компонентов IDE Delphi/C++Builder.




Компоненты "FastReport VCL Client/Server":

- TfrxReportServer - серверный компонент, сервер отчетов и HTTP-сервер одновременно;
- TfrxServerConnection - клиентский компонент, который содержит информацию о соединении с TfrxReportServer;
- TfrxReportClient - клиентский компонент (аналог TfrxReport) запрашивает отчет на сервере, а затем отображает полученный отчет на стороне клиента;
- TfrxHTTPClient - клиентский компонент, предназначенный для запросов файлов по протоколу HTTP.

# TfrxReportServer

TfrxReportServer компонент сервера отчётов и одновременно HTTP сервер. Компонент не требует дополнительных компонентов.

## TfrxReportServer класс унаследован от TComponent

### Свойства

| Свойство | Тип | Описание |
|---|---|---|
| **Active** | Boolean | значение указывает на активность сервера. Может быть использовано для запуска сервера при установке значения в "True" |
| **Configuration** | TfrxServerConfig | конфигурация сервера (TfrxServerConfig класс описан ниже). Конфигурация применяется в момент запуска сервера. |
| **AllowIP** | TStrings | список разрешенных к доступу IP адресов. Одна строка должна содержать один адрес. Если сервер не найдет адрес клиента в списке, то клиенту будет отказано в доступе; если список пустой - разрешен доступ для всех адресов, кроме перечисленных в DenyIP |
| **DenyIP** | TStrings | lсписок запрещенных к доступу IP адресов. Одна строка должна содержать один адрес. |
| **PrintPDF** | Boolean | поведение кнопки печати в навигаторе отчёта на клиенте, если свойство установлено в True, то браузер получит PDF файл для печати, иначе будет сформирован HTML документ. |

*Следующие свойства недоступны в инспекторе объектов, но могут быть установлены из кода приложения:*

| Свойство | Тип | Описание |
|---|---|---|
| **Statistic** | TfrxServerStatistic | статистика сервера (TfrxServerStatistic описан ниже) |
| **Totals** | TStrings | статистика сервера в тектовом виде |
| **Variables** | TfrxServerVariables | внутренние переменные сервера (TfrxServerVariables описан ниже) |

### Методы

| Метод | Описание |
|---|---|
| **constructor Create(AOwner: TComponent)** | создание объекта |
| **procedure Open** | запуск сервера. В этот момент применяются все свойства конфигурации. |

| Метод | Описание |
|---|---|
| **procedure Close** | остановка сервера |

***Обработчики событий:***

OnGetReport: TfrxServerGetReportEvent - может быть использован для загрузки отчёта из любого места (BLOB-поля, файлы и т.д.).

Тип обработчика:

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

ReportName - имя запрашиваемого отчёта; может служить для идентификации того или иного отчёта;

Report - экземпляр объекта класса TfrxReport, в который должен быть загружен отчёт.

OnGetVariables: TfrxServerGetVariablesEvent - может быть использован для ручной обработки параметров, полученных от клиента.

Тип обработчика:

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

ReportName - имя отчёта. Можно использовать для фильтрации параметров в обработчике;

Variables - список параметров полученных от клиента.

# TfrxServerConfig класс унаследован от TPersistent

Объект этого класса содержит информацию о конфигурации сервера.

## Свойства

| Свойство | Тип | Описание |
|---|---|---|
| **Port** | Integer | TCP/IP порт для обслуживания клиентов, по умолчанию равен 80 |
| **IndexFileName** | String | имя файла по умолчанию, если запрашивается пустое имя файла. По умолчанию "index.html" |
| **SessionTimeOut** | Integer | время сохранения результатов построения отчёта на сервере (в секундах). По умолчанию равно 300. После достижения этого времени, результаты построения отчёта удаляются. Устанавливается в зависимости от специфики создаваемых отчетов и методов клиент-серверного взаимодействия; время ожидания активности клиента после его подключения в секундах, по истечении этого времени клиентская сессия будет удалена |
| **SocketTimeOut** | Integer | таймаут ожидания ответа клиента в секундах. По умолчанию 60. По истечении времени сессия будет завершена |

| Свойство | Тип | Описание |
|---|---|---|
| **Logging** | Boolean | логирование, "True" - включено (по умолчанию), "False" - выключено |
| **LogPath** | String | путь к папке с логами; текущая папка по умолчанию |
| **ReportPath** | String | путь к папке с шаблонами отчётов; текущая папка по умолчанию |
| **RootPath** | String | пусть к папке с HTML файлами и результатами построения отчётов |
| **Login** | String | имя пользователя для аутентификации. Если свойство пустое (по умолчанию), то аутентификация не запрашивается |
| **Password** | String | пароль для аутентификации, пустой по умолчанию |
| **Compression** | Boolean | сжатие передаваемых файлов, требуется поддержка клиентом; "True" по умолчанию |
| **MIC** | Boolean | подстчёт контрольной суммы MD5. "True" по умолчанию |
| **NoCacheHeader** | Boolean | отключение кэширования документов на клиенте, "True" по умолчанию |
| **OutputFormats** | TfrxServerOutputFormats | поддерживаемые форматы запрашиваемых отчётов, набор из одного или более из множества (sfHTM, sfXML, sfXLS, sfRTF, sfTXT, sfPDF, sfJPG, sfFRP). По умолчанию включены все поддерживаемые |
| **ReportCaching** | Boolean | включает кэширование отчётов на сервере |
| **ReportCachePath** | String | путь к папке с кэшем отчётов |
| **DefaultCacheLatency** | Integer | время хранения файлов в кэше отчётов (в секундах) |

### Методы

| Метод | Описание |
|---|---|
| **procedure LoadFromFile(const FileName: String)** | загружает конфигурацию из файла |
| **procedure SaveToFile(const FileName: String)** | сохраняет конфигурацию в файл |

## TfrxServerStatistic класс унаследован от TPersistent

### Свойства

| Свойство | Тип | Описание |
|---|---|---|
| **CurrentReportsCount** | Integer | количество строящихся отчётов в данный момент |

| Свойство | Тип | Описание |
|---|---|---|
| **CurrentSessionsCount** | Integer | количество сессий в данный момент |
| **MaxReportsCount** | Integer | максимальное количество отчётов, которые были построены одновременно |
| **MaxSessionsCount** | Integer | максимальное количество сессий, которые были подключены одновременно |
| **TotalErrors** | Integer | общее количество ошибок |
| **TotalReportsCount** | Integer | общее количество построенных отчётов |
| **TotalSessionsCount** | Integer | общее количество сессий |
| **UpTimeDays** | Integer | количество дней с момента запуска |
| **UpTimeHours** | Integer | количество часов с момента запуска |
| **UpTimeMins** | Integer | количество минут с момента запуска |
| **UpTimeSecs** | Integer | количество секунд с момента запуска |

# TfrxServerVariables класс унаследован от TCollection

Содержит переменные сервера.

Использованные имена переменных.

## Методы

| Метод | Описание |
|---|---|
| **function GetValue(const Name: String): String** | возвращает значение паеременной с именем Name |
| **procedure AddVariable(const Name: String; const Value: String)** | добавляет переменную с именем Name и значением Value |

# TfrxServerConnection

TfrxServerConnection - клиентский компонент содержащий информацию о подключении к серверу отчётов TfrxReportServer. Объект этого класса необходим для работы нескольких компонент TfrxReportClient.

## TfrxServerConnection класс унаследован от TComponent

### Свойства

| Свойство | Тип | Описание |
|---|---|---|
| **Host** | String | имя хоста сервера или IP адрес, по умолчанию - 127.0.0.1 |
| **Port** | Integer | порт сервера; "80" по умолчанию |
| **ProxyHost** | String | имя HTTP-прокси или IP адрес |
| **ProxyPort** | Integer | порт HTTP-прокси; "8080" по умолчанию |
| **Login** | String | имя пользователя для аутентификации |
| **Password** | String | пароль пользователя для аутентификации |
| **Timeout** | Integer | таймаут (в секундах); "120" по умолчанию |
| **RetryCount** | Integer | количество повторных попыток подключения; "3" по умолчанию |
| **RetryTimeout** | Integer | задержка между повторами подключения, "3" по умолчанию |
| **Compression** | Boolean | принимать сжатые файлы; "True" по умолчанию |
| **MIC** | Boolean | проверка контрольной суммы; "True" по умолчанию |

# TfrxReportClient

TfrxReportClient клиентский компонент предназначенный для запроса и показа отчёта.

Требуемый компонент: TfrxServerConnection.

TfrxReportClient это аналог класса TfrxReport в традиционных архитектурах приложений.

## TfrxReportClient класс унаследован от TfrxReport

### Свойства

| Свойство | Тип | Описание |
|---|---|---|
| **Connection** | TfrxServerConnection | ссылка на объект класса TfrxServerConnection |
| **ReportName** | String | имя запрашиваемого отчёта, можно использовать метод LoadFromFile для заполнения этого свойства (см. ниже) |
| **Variables** | TfrxVariables | переменные отчёта; сможет быть использовано для передачи переменных отчёта на сервер |
| **Errors** | TStrings | список ошибок |

### Методы

| Метод | Описание |
|---|---|
| **procedure LoadFromFile(FileName: String)** | устанавливает имя запрашиваемого отчёта, см. свойство ReportName |
| **function PrepareReport: Boolean** | выполняет подключение к серверу отчётов, запрашивает исполнение отчёта, передает параметры, получает результат, который помещается в свойство "PreviewPages". Результат функции равен "True" если задача выполнена успешно, иначе возвращает "False" |
| **procedure ShowPreparedReport** | Показывает полученный от сервера отчёт |
| **procedure ShowReport** | запрашивает отчёт и показывает его |

# TfrxHTTPClient

TfrxHTTPClient - клиентский компонент для получения любых файлов через HTTP протокол.

## TfrxHTTPClient класс унаследован от TComponet

### Свойства

| Свойство | Тип | Описание |
|---|---|---|
| **Active** | Boolean | выполняется запрос, если значение установлено в "True" |
| **Host** | String | имя или IP адрес хоста; по умолчанию "127.0.0.1" |
| **Port** | Integer | порт хоста, по умолчанию "80" |
| **ProxyHost** | String | имя или IP-адрес HTTP-прокси |
| **ProxyPort** | Integer | порт прокси |
| **RetryCount** | Integer | число повторов, по умолчанию - 3 |
| **RetryTimeOut** | Integer | задержка между повторами в секундах, по умолчанию - 5 |
| **TimeOut** | Integer | время ожидания ответа в секундах, по умолчанию - 30 |
| **ClientFields** | TfrxHTTPClientFields | поля заголовка запроса, TfrxHTTPClientFields описан ниже |
| **ServerFields** | TfrxHTTPServerFields | полученные поля заголовка ответа, TfrxHTTPServerFields описан ниже |
| **MIC** | Boolean | проверка контрольной суммы сообщения, по умолчанию - "True" |
| **Header** | TStrings | заголовки запроса; это свойство заполняется автоматически из значений ClientFields |
| **Answer** | TStrings | заголовки ответа, автоматически заполняет значения ServerFields |
| **Stream** | TMemoryStream | поток данных полученных от сервера |
| **Breaked** | Boolean | признак прерванного запроса |
| **Errors** | TStrings | текст ошибок |

### Методы

| Метод | Описание |
|---|---|
| **procedure Connect** | подключается к серверу, получает данные |
| **procedure Disconnect** | отключается от сервера |
| **procedure Open** | то же, что и "Connect" |
| **procedure Close** | то же, что и "Disconnect" |

# TfrxHTTPClientFields класс унаследован от TPersistent

## Свойства

| Свойство | Тип | Описание |
|---|---|---|
| **AcceptEncoding** | String | поддерживаемые форматы сжатия, по умолчанию - gzip |
| **FileName** | String | запрашиваемое имя файла |
| **Host** | String | адрес клиента; заполняется автоматически, если пустое |
| **HTTPVer** | String | версия протокола HTTP, по умолчанию - HTTP/1.1 |
| **Login** | String | имя пользователя для аутентификации |
| **Password** | String | пароль для аутентификации |
| **QueryType** | TfrxHTTPQueryType | тип запроса, qtGet - GET запрос, qtPost - POST запрос; "qtGet" по умолчанию |
| **Referer** | String | сслающийся адрес; пусто по умолчанию |
| **UserAgent** | String | имя клиентской программы, по умолчанию - FastReport/3.0 |

# TfrxHTTPServerFields класс унаследован от TPersistent

## Свойства

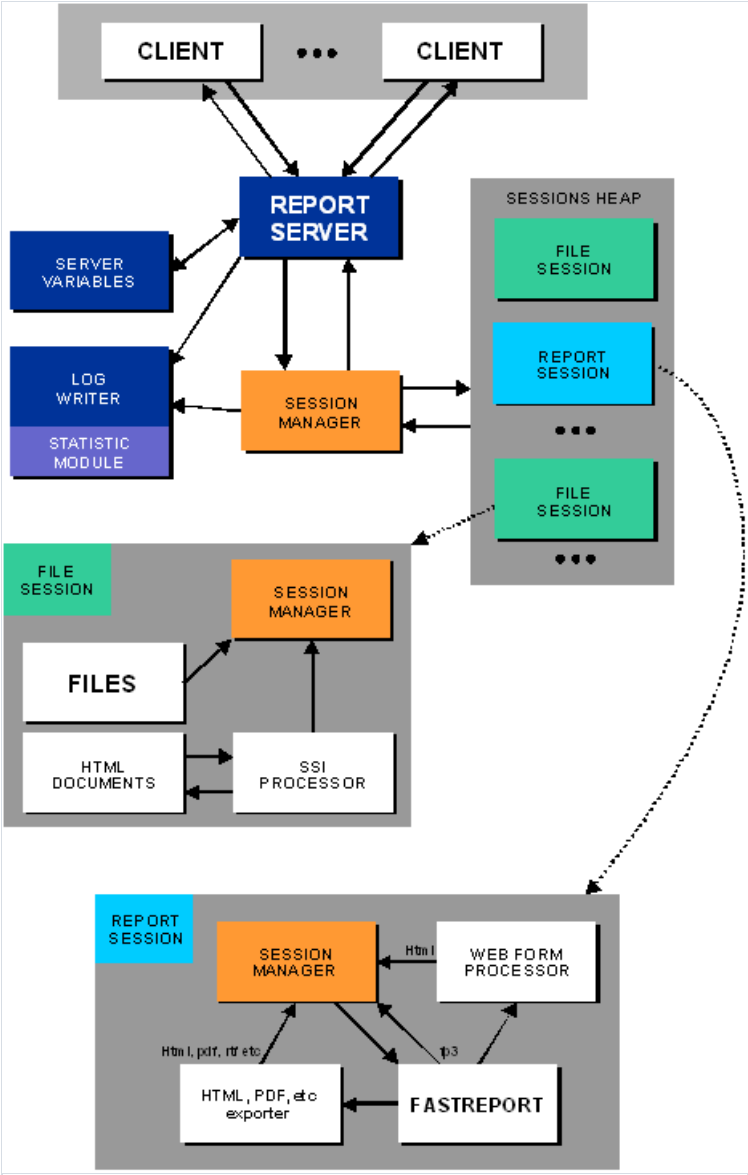
| Свойство | Тип | Описание |
|---|---|---|
| **AnswerCode** | Integer | код ответа сервера |
| **ContentEncoding** | String | тип сжатия полученного контента |
| **ContentMD5** | String | контрольная сумма MD5 |
| **ContentLength** | Integer | размер полученных данных |
| **Location** | String | адрес документа |

# Сервер отчётов

Серверная часть (компонент TfrxReportServer) представляет собой автономный HTTP-сервер с возможностью построения отчётов. Сервер отчётов может строить несколько отчётов одновременно, логировать события и собирать статистику.

# Внутренняя архитектура

На схеме показана внутренняя структура сервера:

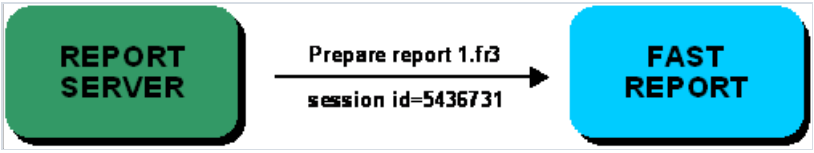



При поступлении запроса от клиента создается сессия с уникальным идентификатором. Строка запроса анализируется. Если запрашиваемый файл существует, то сервер посылает клиенту положительный ответ с файлом. Журналы обновляются новой записью об этом событии. Если запрос содержит запрос отчета, то создается специальная сессия отчета. После построения отчета результат сохраняется в папке с номером сессии в качестве имени. Сервер отвечает клиенту и сообщает новое местоположение файла. Клиент посылает новый запрос на новое местоположение файла и получает файл с результатом. Сессия с полученным файлом хранится на сервере до истечения времени сессии.

Ниже приведен пошаговый графический обзор операции запроса отчета с помощью веб-браузера:

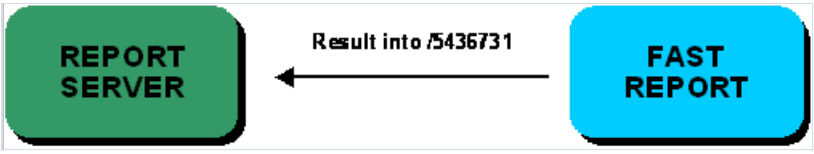
- клиент отправляет запрос; имя отчета - "1.fr3"

- сервер создает новый экземпляр FastReport и передает параметры запроса




- FastReport готовит отчет и экспортирует результаты в html-файл в папку, название папки совпадает с номером сессии


- сервер ожидает результатов от FastReport




- клиент получает перенаправление на местоположение результирующего файла


- клиент посылает новый запрос файла результата


- сервер доставляет файл результата клиенту

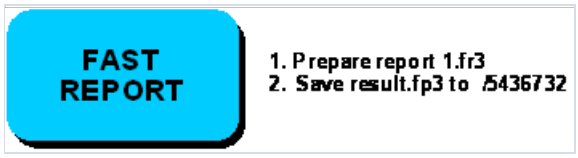

Пошаговый графический обзор операции запроса отчета с помощью FastReport (TfrxReportClient):

- клиент хочет показать отчет "1.fr3":

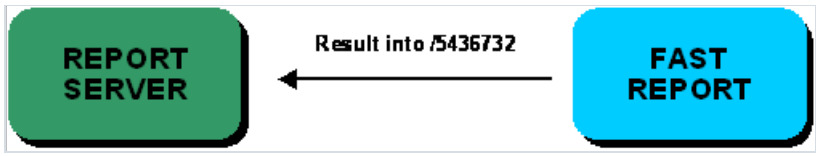

- клиентский компонент отправляет запрос с именем отчета "1.fr3" (собственный формат результата)


...

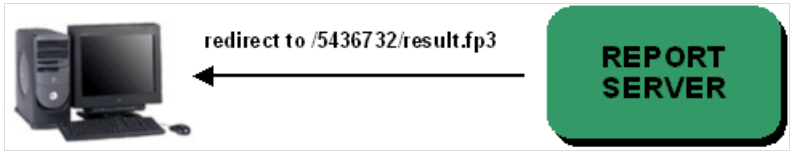
- FastReport подготавливает отчет и сохраняет результаты в файл fp3; имя папки совпадает с номером сессии




- сервер ожидает результатов от FastReport

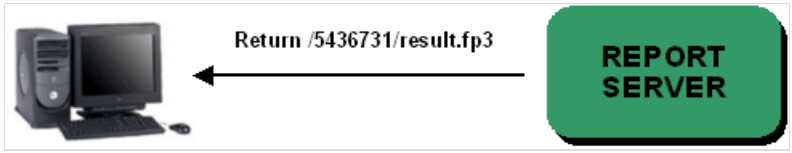



- клиент получает перенаправление на местоположение результирующего файла




...

- сервер отправляет файл результата клиенту

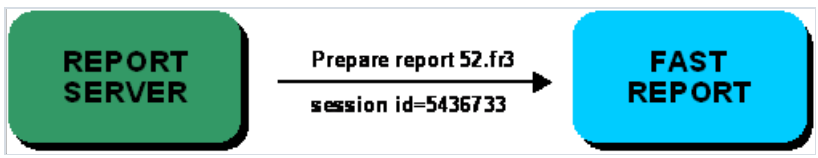



- клиент отображает отчет


Если запрашиваемый отчет содержит какие-либо формы, процесс усложняется:

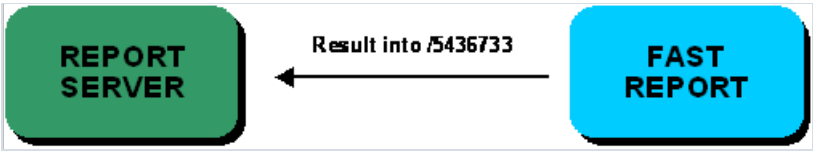
- клиентский компонент отправляет запрос с именем отчета "1.fr3"


- сервер создает новый экземпляр FastReport и передает параметры запроса




- FastReport готовит отчет и сохраняет веб-форму в папке с именем в соответствии с номером сессии

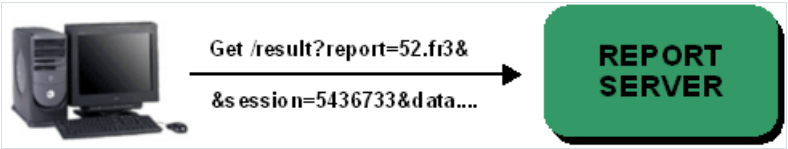

- сервер ожидает результатов от FastReport



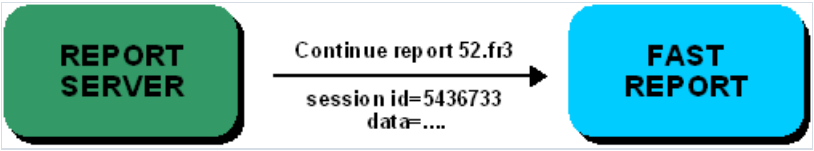

- сервер перенаправляет клиента на файл веб-формы


- клиент получает веб-форму, в то время как FastReport ждет


- клиент отправляет состояния элементов управления диалоговой веб-формы на сервер




- сервер передает значения управляющих элементов на сервер




- сервер доставляет полученную информацию в FastReport

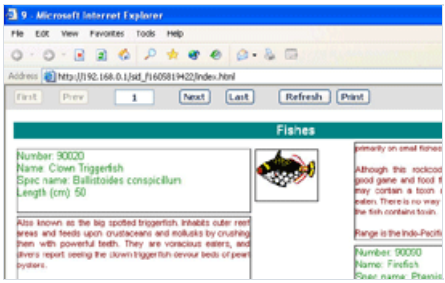

Формат строки запроса сервера, протоколирование, аутентификация и другие вопросы, касающиеся функционирования сервера, описаны ниже.

# Поддерживаемые форматы результатов отчета

FR3 - это собственный формат FastReport VCL. Он представлен в виде XML-документа. FR3 используется во время транзакции между TfrxReportServer и TfrxReportClient. Этот формат является наиболее подходящим для печати документов. В большинстве случаев использование этого формата сокращает как время транзакции, так и размер передаваемых файлов (за исключением отчетов, содержащих высококачественные изображения).

Нежелательно использовать дополнительные компоненты сжатия (TfrxGZipCompressor) на стороне сервера, так как это снижает общую производительность сервера, особенно если активирована опция сжатия трафика (TfrxReportServer.Configuration.Compression := True; TfrxReportClient.Connection.Compression := True).

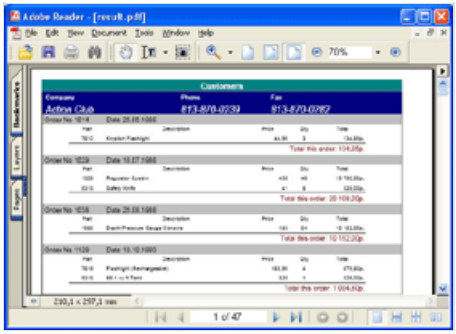
Формат HTML, используемый большинством веб-сайтов в сети Интернет, предназначен для предварительного просмотра документа в низком разрешении. Выполнить качественную печать документа с использованием этого формата достаточно сложно. Формат HTML удобен для большинства веб-браузеров. Если вы используете веб-браузер в качестве клиента, то вам подходит именно этот формат. Сервер FastReport создает веб-страницы с навигатором отчетов, с помощью которого можно переключать страницы.




Формат PDF от Adobe разработан специально для документов, предназначенных для печати.

FastReport осуществляет высококачественный экспорт в этот формат. Для просмотра и печати PDF-документов необходимо установить на компьютер программу Adobe Acrobat Reader.

Если установлено свойство "TfrxReportServer.PrintPDF := True", то при предварительном просмотре HTML страниц с результатами отчета формируется файл в этом формате (по нажатию кнопки "Печать" на панели навигатора отчета).



Сервер также поддерживает следующие форматы:

- Формат RTF. Документ RichText можно открыть в большинстве текстовых процессоров;
- XLS и XML. Это форматы электронных таблиц Excel;
- Текстовый файл (требуется для матричной печати);
- Графический файл jpeg.

Набор форматов, разрешенных для использования в запросах, задается свойством

"TfrxReportServer.Configuration.OutputFormats", которое может содержать одно или несколько значений из следующего набора: sfHTM - формат HTML, sfXML - формат XML, sfXLS - формат Excel, sfRTF - формат RichText, sfTXT - текстовый файл, sfPDF - формат Adobe Acrobat, sfJPG - jpeg картинка, sfFRP - собственный формат FastReport 3 (FR3).

Если тип возвращаемого формата не указан при запросе, то сервер генерирует результат в формате HTML.

# Синтаксис запросов

При использовании обычного веб-браузера в качестве клиента можно использовать параметры строки запроса:

**report=*name***

*name* - название отчета, доступного на сервере

Пример: */report=1.fr3* (запрос отчёта 1.fr3, результат - HTML).

**format=*name***

*name* - формат запрашиваемого файла, возможные варианты: **HTM** (HTML), **XML** (xml таблица), **XLS** (Excel таблица), **RTF** (rich-text файл), **TXT** (текстовый файл), **PDF**(Adobe Acrobat файл), **JPG** (jpeg изображение), **FRP** (внутренний формат FastReport).

По умолчанию формат **HTM** (HTML).

пример: */report=1.fr3&format=TXT* (запрос отчёта 1.fr3, результат - текстовый файл).

**pagerange=*value***

*value* - диапазон страниц (для FRP недоступно).

Пример диапазона: 1,3,5-12.

Пример строки запроса: */report=3.fr3&pagerange=20-25* (запрос отчёта 3.fr3, страница с 20 по 25, результат - HTML).

**multipage=param**

Только для формата HTM. Если значение параметра равно "1", то результирующий отчет будет представлен в виде нескольких страниц (по одному файлу на каждой странице). Если значение параметра равно "0", то будет сформирована одна результирующая страница, содержащая все страницы отчета. По умолчанию значение параметра равно "1".

Пример: */report=3.fr3&multipage=0* (запрос отчёта 3.fr3, результат - HTML, все страницы отчёта разместить на одной HTML странице).

**pagenav=*param***

Только для формата HTM. Чтобы включить навигатор страниц, установите значение параметра как "1". Если значение параметра равно "0", то навигатор страниц выключен. Для корректного отображения страницы используйте веб-браузер с поддержкой javascript и фреймов. По умолчанию значение этого параметра равно "1".

Пример: */report=9.fr3&multipage=0&pagenav=0* (отчет 9.fr3, результирующий формат - HTML, все страницы результатов на одной HTML-странице, навигатор страниц выключен).

# Передача параметров в отчет

Если в строке запроса представлены другие параметры (не перечисленные выше), то сервер интерпретирует их как параметры для построения отчета в виде внутренней переменной FastReport.

**Пример:**

`/report=myreport.fr3&param1=Hello%20World!` (запрос отчёта "myreport.fr3", параметр "param1" содержит значение "Hello World!")

*Ниже приведены некоторые ограничения, касающиеся передаваемых параметров:*

- все строки должнгы быть преобразованы в формат Unicode UTF-8 и совместимы со стандартом HTTP-запросов (используйте функцию Untf8Encode и функцию HTMLCodeStr, объявленне в файле frxServerUtils.pas);
- все параметры передаются в отчет в виде строк. Пожалуйста, помните об этом, когда будете использовать эти параметры в шаблоне отчета;
- все переменные, содержащиеся в TfrxReportClient.Variables, автоматически передаются на сервер.

# Внутренние переменные сервера

Во время работы сервера свойство "TfrxReportServer.Variables" содержит следующие автоматически созданные и обновляемые переменные:

`SERVER_NAME` - имя сервера;

`SERVER_COPYRIGHT` - авторское право;

`SERVER_SOFTWARE` - версия сервера;

`SERVER_LAST_UPDATE` - дата обновления сервера;

`SERVER_UPTIME` - время работы сервера;

`SERVER_TOTAL_SESSIONS` - всего количество сессий;

`SERVER_TOTAL_REPORTS` - всего количество отчётов;

`SERVER_TOTAL_ERRORS` - всего количество ошибок;

`SERVER_MAX_SESSIONS` - максимальное количество одновременных сессий;

`SERVER_MAX_REPORTS` - максимальное количество одновременных построений отчётов.

## Пример получения переменной SERVER_TOTAL_REPORTS

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

# Использование документов HTML

Сервер можно использовать как простой HTTP-сервер для хранения и просмотра любых HTML-документов или любых других файлов.

Поместите HTML-документы в папку, а затем установите свойство TfrxReportServer.Configuration.RootPath.

Имя документа по умолчанию должно быть указано в свойстве TfrxReportServer.Configuration.IndexFileName (по умолчанию index.html). Соответственно, документ с таким именем должен существовать в корневой папке.

**Описание команд SSI (Server Side Include).**

*Включить любой файл в документ.*

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

Включить файл с именем filename.html в текущую позицию документа. Путь к файлу указывается из RootPath.

Пример:

```
<!--#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>
...
```

*Вставить значение переменной сервера.*

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

Вставить значение переменной с именем "VARIABLE" в текущую позицию документа.

Пример:

```
...

<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>
...
```

Использование команд SSI упрощает разработку сайта.

Пример сайта с SSI вы можете увидеть в папке "\Demos\ClientServer\Server\htdocs".

# Логи

Если параметр свойства TfrxReportServer.Configuration.Logging равен "True", то сервер записывает журналы в папку, указанную в свойстве "TfrxReportServer.Configuration.LogPath".

Сервер поддерживает 5 журналов:

- журнал обратившихся клиентов "access.log" - содержит информацию о дате, времени, идентификаторе сессии, IP 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 подключенной программы типа "agent.log", содержит информацию о дате, времени, IP и имени программы. 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 ссылающихся URL "referer.log", содержит информацию о дате, времени, IP и ссылающемся URL. 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 ошибок "error.log", содержит информацию об ошибках:

```
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)
```

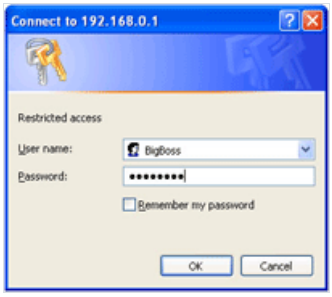
- журнал сервера "server.log", содержит сводную информацию о сервере:

```
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
```

Не забудьте заархивировать файлы журнала.

# Аутентификация

Сервер поддерживает базовую HTTP-аутентификацию. Чтобы активировать аутентификацию, установите свойства TfrxReportServer.Configuration.Login и TfrxReportServer.Configuration.Password. Если вы установили эти свойства, то заголовок запроса должен содержать информацию об аутентификации (RFC 2068 [2]). Если клиент получает ответ от сервера с кодом ошибки 401 "Unauthorized", то клиент должен повторить отправку запроса с правильными данными аутентификации. При этом веб-браузер просто показывает диалоговое окно с запросом логина и пароля:



# Ограничение доступа по IP-адресу

Сервер поддерживает ограничение по IP-адресу клиента.

Свойство TfrxReportServer.DenyIP может содержать список запрещенных IP-адресов клиентов.

Свойство TfrxReportServer.AllowIP может содержать список разрешенных IP-адресов клиентов. Каждый список должен содержать один IP-адрес в одной строке.

Пример такого списка:

```
192.168.0.10
192.168.0.12
192.168.0.54
```

Если списки "DenyIP" и "AllowIP" пусты, то всем клиентам разрешено подключаться к серверу.

Если список "DenyIP" пуст, а список "AllowIP" содержит IP-адрес, то только один клиент с этим IP-адресом может подключиться к серверу.

Если IP-адрес подключенного клиента не включен в список список "DenyIP", то сервер проверяет, включен ли этот адрес в список "AllowIP".

Маски IP-адресов не поддерживаются.

**Примеры:**

1. Только локальный хост может подключаться к серверу:

AllowIP:

```
127.0.0.1
```

DenyIP пуст.

2. IP-адреса 192.168.0.2 - 192.168.0.6 могут подключаться к серверу:

AllowIP:

```
192.168.0.2
192.168.0.3
192.168.0.4
192.168.0.5
192.168.0.6
```

DenyIP пуст.

3. IP-адреса из диапазона 192.168.0.8 - 192.168.0.10 не могут подключаться к серверу.

AllowP пуст.

DenyIP:

```
192.168.0.8
192.168.0.9
192.168.0.10
```

# Подключение к базе данных

Большинство отчетов используют данные из баз данных. Чтобы подключиться к базе данных, необходимо:

- указать компонент подключения к базе данных (например, TADOConnection) в одной из форм вашего приложения;
- использовать внутренние компоненты доступа к данным (например, TfrxADOTable, TfrxADOQuery) в вашем отчете. Для подключения к базе данных эти компоненты должны использовать соединение приложения.

В этом случае отчет будет потокобезопасным, используя одно соединение с базой данных. В случае если некоторые компоненты доступа к данным не поддерживают одновременную работу с базой данных через одно соединение, следует воспользоваться способом, описанным ниже.

Другой способ - использовать компонент подключения к базе данных (например, TfrxADODatabase) в каждом отчете. В этом случае вы сможете одновременно подключаться к разным базам данных. Мы не рекомендуем использовать этот способ, если вам не нужна эта функциональность, поскольку каждый раз при запуске отчета он будет пытаться подключиться к базе данных (на некоторых DB-серверах подключение может занять много времени).

Прочитайте "FastReport VCL - руководство пользователя" [9], чтобы узнать больше о создании отчетов с внутренними компонентами доступа к данным.

Не рекомендуется использовать BDE для подключения к базе данных. BDE имеет большое количество проблем при работе с несколькими потоками.

# Использование кэша отчетов

Кэширование отчетов позволяет достичь высокой эффективности за счет сохранения подготовленных отчетов во временных файлах сервера. В зависимости от конфигурации сервера, после подготовки результат может быть помещен в кэш.

По истечении заданного времени результат отчета будет удален из кэша.

Если в течение этого времени от клиента поступит запрос с *таким же именем отчета и такими же значениями параметров*, то ему будет немедленно возвращен ответ. Ответ будет основан на результате, сохраненном в кэше, и будет представлен в формате, запрошенном клиентом.

В этом случае сервер будет тратить время только на преобразование готового отчета в запрашиваемый формат, не занимаясь построением отчета. Это значительно повышает производительность.

В зависимости от задач, выполняемых сервером, можно назначить индивидуальное время хранения в кэше для каждого конкретного отчета.

Значение времени устанавливается администратором сервера, в соответствии с актуальностью отчета, через определенный промежуток времени.

Например, годовой отчет о деятельности предприятия может храниться в кэше достаточно долго, так как информация будет актуальна в течение длительного периода времени, и не устареет очень быстро. Напротив, отчет о складе крупной коммерческой организации будет актуален в течение небольшого периода времени, и поэтому его, соответственно, следует хранить в кэше не слишком долго.

Свойства кэша отчетов:

`TfrxServer.Configuration.ReportCaching` - включение кэша (True/False);

`TfrxServer.Configuration.ReportCachePath` - путь к папке с кэшем;

`TfrxServer.Configuration.DefaultCacheLatency` - таймаут кэша, по умолчанию 300 секунд.

Параметры файла конфигурации сервера:

```
[ReportsCache]
; включить кэширование отчетов с одинаковыми параметрами
Enabled=1
; путь к папке cache
CachePath=.\cache\
; задержка для кэширования результатов отчета в секундах
DefaultLatency=300
```

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

```
[ReportsLatency]
; задержка кэша для отчета 1.fr3 в секундах
1.fr3=10
; задержка кэша для отчета 2.fr3 в секундах
2.fr3=20
; добавьте ниже отчеты для настройки задержки кэша
```

Корректировка конфигурации параметров позволит минимизировать время работы клиентов и уменьшит общий трафик на сервере.

# Увеличение производительности сервера

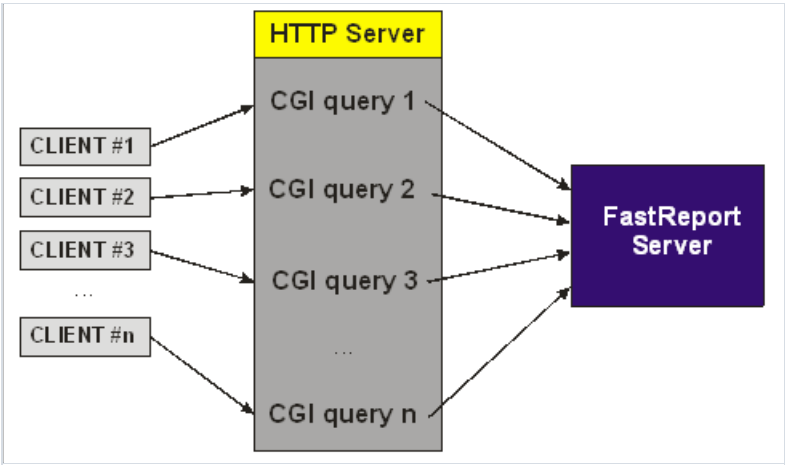
Используйте следующие рекомендации для увеличения производительности сервера отчетов:

- не используйте компонент сжатия (TfrxGZipCompressor). Он значительно замедляет работу сервера;
- оптимизируйте SQL-запросы. В некоторых случаях выполнение SQL-запроса может занять больше времени, чем выполнение отчета;
- не используйте в отчетах растровые изображения высокого разрешения - это увеличивает время выполнения отчета и сетевой трафик;
- не используйте сложные скрипты в отчетах;
- используйте компонент TfrxReportClient в вашем клиентском приложении. Он работает с родным форматом FP3 и позволяет сократить время отклика сервера (сервер не выполняет экспорт в HTML или другие форматы) и сетевой трафик.
- при разработке отчета учитывайте наши рекомендации;
- отключите контрольную сумму, TfrxReportServer.Configuration.MIC := False;
- увеличте объем памяти, используйте более быстрый процессор на сервере отчетов.

# Использование сервера FastReport вместе с другими HTTP-серверами (Apache, IIS и т.д.)

Для использования уже существующих решений на базе других HTTP-серверов возможна их интеграция с сервером FastReport с помощью механизма "CGI". Это дает преимущество по сравнению с использованием 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 (в HTTP сервере FastReport эта возможность пока недоступна).

Применяя такой метод, CGI становится промежуточным звеном для передачи запроса на сервер "FastReport", получения результатов с сервера отчетов, и возврата результатов клиенту.




Пример CGI-обертки вы можете найти в папке "Demos\ClientServer\CGI".

**Для использования CGI-обертки:**.

- скомпилируйте и скопируйте файл fastreport.exe в папку /cgi-bin HTTP-сервера;
- настройте HTTP-сервер (Apache, IIS или другой) для выполнения CGI-приложения. Подробнее об этом читайте в руководстве пользователя HTTP-сервера;

*Если HTTP и серверы отчетов работают на одном компьютере:*

- если TCP/IP порт 80 используется HTTP сервером, настройте FastReport сервер на другой порт 8097 (этот порт используется CGI приложением по умолчанию, если конфигурационный файл пропущен), если вы хотите использовать другой TCP/IP порт, читайте ниже об использовании конфигурационного файла CGI приложения;

*Если HTTP и FastReport серверы работают на разных компьютерах:*

- создайте конфигурационный файл CGI-приложения в папке /cgi-bin с именем fastreport.ini:

```
[REPORTSERVER]
; IP-адрес сервера FastReport
Host=192.168.0.34
; IP-порт сервера FastReport
Port=80
```

- запустите сервер FastReport и проверьте работу CGI-приложения.

Пример запроса к отчету с использованием CGI-приложения: http://127.0.0.1/cgi-bin/fastreport.exe?report=67.fr3&multipage=0&pagenav=0

Подробнее о синтаксисе строк запроса читайте тут. Замените ключевое слово "result" в этом пункте на конструкцию "cgi-bin/fastreport.exe".

Внимание: чтобы ограничить прямой доступ клиентов к серверу отчетов, необходимо указать IP-адрес HTTP-сервера, на котором работает CGI-приложение (127.0.0.1 или другой).

# Разработка отчётов

Разработка отчётов описана в руководстве пользователя FastReport VCL.

# Некоторые клиент-серверные ограничения

При разработке отчетов для клиент-серверного приложения, пожалуйста, помните, что:

- нельзя использовать обработчики событий сценариев для элементов управления диалоговых форм, так как диалоговые формы отображаются в браузере как веб-формы;
- нельзя использовать обработчики событий компонента TfrxReportClient (например, OnGetValue, OnUserFunction). Все такие обработчики должны находиться на стороне сервера;
- нельзя использовать общие компоненты доступа к данным, такие как TfrxDBDataSet (общие компоненты не могут одновременно использоваться несколькими отчетами). Каждый отчет должен иметь внутренние компоненты доступа к данным, такие как TfrxIBXTable, Query и так далее.

# Несколько советов по оформлению отчета

Многие из форматов документов используют табличный стиль представления данных. Для представления результирующих отчетов сервер использует такие форматы, как HTML, XLS и RTF.

Документы табличного стиля не могут иметь пересекающихся ячеек, в то время как документ FastReport может. FastReport использует произвольное расположение данных - нет никаких "строк", "ячеек таблицы", как в Word, Excel или других подобных форматах. Фильтры экспорта FastReport для табличных форматов (RTF, HTML и XLS) используют специальный алгоритм для преобразования пересекающихся ячеек в ячейки таблицы и оптимального их расположения. В местах, где объекты FastReport пересекаются друг с другом, фильтр экспорта может генерировать дополнительные строки и столбцы таблицы. Это необходимо для лучшего WYSIWYG, но может привести к увеличению количества строк и столбцов в результирующей таблице, что делает её малопригодной для дальнейшего чтения и замедляет процесс экспорта.

Помните об этих ограничениях экспорта при разработке отчета, если вы собираетесь экспортировать свой отчет в такие табличные форматы. Чтобы избежать пересечения объектов, используйте инструменты выравнивания дизайнера FastReport. Включите опцию "выравнивание по сетке".

При создании таблиц в отчете располагайте ячейки таблицы рядом друг с другом, если это возможно, и избегайте пересечения ячеек. Если ячейки пересекаются, алгоритм экспорта будет производить обрезку, и результат экспорта может отличаться от исходного отчета.

По возможности размещайте объекты вдоль горизонтальных и вертикальных направляющих линий. Для этого используйте направляющие линии дизайнера.

Следуя этим инструкциям, ваши отчеты будут выглядеть идеально при экспорте в любой из поддерживаемых форматов.

# Клиент

Существует два вида клиентов сервера FastReport:

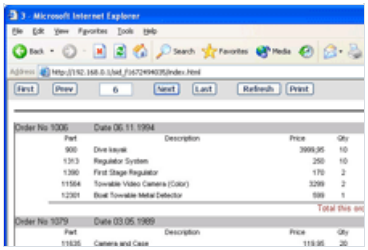
- приложения, использующие компонент TfrxReportClient;
- любые автономные HTTP-клиенты, такие как веб-браузеры.

# Клиент на основе TfrxReportClient

Компонент TfrxReportClient разработан специально для клиентских приложений. Этот компонент позволяет запрашивать отчет с сервера, передавая серверу некоторые параметры отчета (переменные). Он получает подготовленный отчет в формате FP3 (собственный формат FastReport). Подготовленный отчет может быть отображен и распечатан на стороне клиента. Вы также можете экспортировать подготовленный отчет в любой из поддерживаемых форматов, используя компоненты фильтра экспорта. В большинстве случаев это решение является оптимальным для клиентских приложений. Клиенты, использующие компонент TfrxReportClient, создают низкий сетевой трафик и используют меньше системных ресурсов сервера.

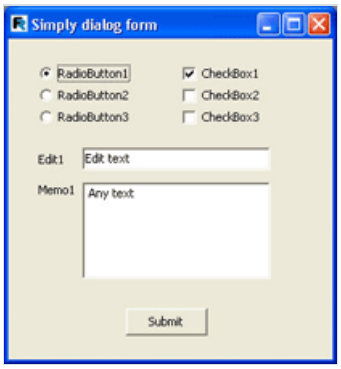
# Другие клиенты

Сервер FastReport предоставляет широкие возможности выбора клиентской программы благодаря использованию стандартного протокола HTTP. Вы можете использовать любой HTTP-совместимый клиент, например, веб-браузер, поддерживающий JavaScript, таблицы и фреймы.

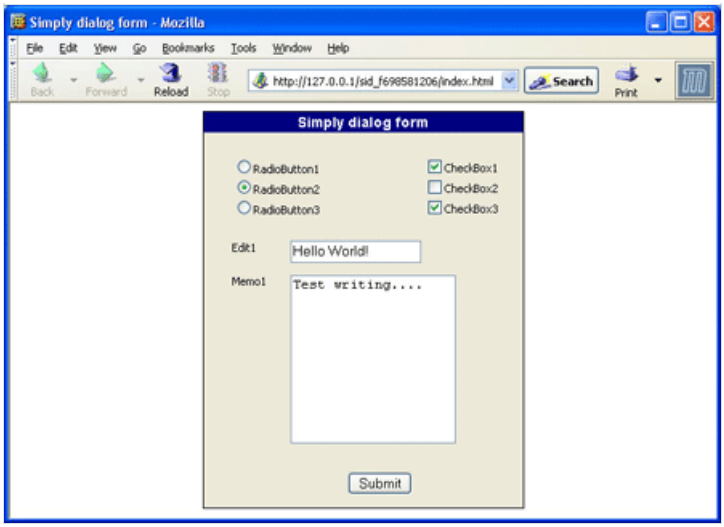


При использовании диалоговых форм в отчетах сервер преобразует их в веб-формы и передает клиенту. Клиент должен заполнить форму и вернуть ее серверу.

Вот как выглядит диалоговая форма при запуске отчета в простом (не клиент-серверном) приложении:




Такая же форма появляется в веб-браузере при запуске отчета в клиент-серверном приложении.



# Адаптация приложений под архитектуру клиент-сервер

При адаптации ранее разработанных приложений к архитектуре клиент-сервер используйте следующие рекомендации:

- Четко определите взаимодействие между сторонами клиента и сервера;
- Принять во внимание рекомендации по темам 1 и 2;
- При работе с базами данных учитывайте рекомендации из статьи;
- Чтобы повысить уровень информационной безопасности, учитывайте рекомендации.

# Пример простого клиент-серверного приложения

Для ознакомления с методами использования компонентов FastReport Enterprise посмотрите демонстрационные примеры, находящиеся в папке "\Demos\ClientServer".

# Серверное приложение

Вы можете найти все исходные файлы этого примера в папке \Demos\ClientServer\Server.

Компоненты, используемые в данной демонстрации: серверный компонент TfrxReportServer (Serv), компоненты подключения к базе данных TADOConnection и TfrxADOComponents, а также другие дополнительные компоненты FastReport.


Для удобства клиентов данные о конфигурации сервера хранятся в файле, который редактируется встроенным редактором.

Файл 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
```

Поля конфигурационного файла соответствуют именам полей свойства TfrxReportServer.Configuration.

Файлы "allow.conf" и "deny.conf" содержат строки с разрешенными и запрещенными адресами соответственно.

Файл базы данных хранится в папке "database".

В главном модуле определяются константы с именами конфигурационных файлов:

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

После запуска программы происходит подключение к базе данных через интерфейс MicrosoftJet OLE DB.

В переменных ConfFile, AllowFile, DenyFile мы храним пути к файлам конфигурации:

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

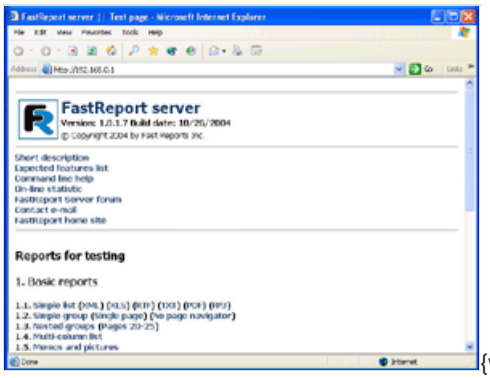
Загрузка файлов конфигурации в объекте Serv:

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

Запуск сервера:

```
Serv.Open;
```

После выполнения всех работ можно использовать сервер отчетов. Запустите любой веб-браузер и введите http://127.0.0.1 в адресной строке



{width="296" height="227"}

Вы можете разрабатывать отчеты с помощью внутреннего дизайнера FastReport:

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

# Клиентское приложение

Исходный код примера можно найти в папке \Demos\ClientServer\Client\Simple.

Это пример использования компонента TfrxReportClient и передачи параметров отчета на сервер.

Перед запуском этой программы запустите серверное приложение.


Нажмите кнопку "Показать отчет" и введите "1.fr3" в поле "Имя отчета" при запуске этого примера, чтобы программа выполнила приведенный ниже 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);
```

После успешного запроса отчета вы увидите окно предварительного просмотра результата.

# Клиентское многопоточное приложение

Вы можете найти все исходные файлы этого примера в папке "\\Demos\\ClientServer\\Client\\Advanced".

Этот пример показывает, как можно использовать компонент "TfrxReportClient" в потоках.


Класс потока:

```
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;
```

Конструктор класса TfrxClientTestThread:

```
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` отправляет на сервер серию запросов. Вся полученная информация отображается 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 запуском этой программы запустите серверное приложение, описанное выше.

По нажатию кнопки "Thread test" выполнится приведенный ниже 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.PrxoyHost := ProxyHost.Text;
    frxServerConnection1.ProxyPort := 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;
```

# Важные вопросы безопасности

1. При использовании сервера отчетов на платформе Microsoft Windows через Интернет рекомендуется использовать межсетевой экран между сервером и сетью Интернет.
2. Обязательно использовать аутентификацию клиентской программы.
3. Используйте функцию "allow/deny IP" для локальной сети.
4. Если в локальной сети есть шлюзы в Интернет, то включите IP-адреса этих шлюзов в список "deny" сервера отчетов.
5. Не передавайте параметры в соединение с базой данных с клиента, если вы используете отчеты с внутренними компонентами подключений к базам данных.
6. В папке reports храните только те отчеты, которые вы используете в своем приложении.
7. Не храните личные документы в корневой папке HTTP.
8. Если вы обнаружили какие-либо ошибки в системе безопасности FastReport Enterprise, обязательно сообщите об этом разработчикам продукта.

# Ссылки


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.

# Контакты

Если у вас есть какие-либо предложения по улучшению и развитию FastReport VCL Enterprise, пожалуйста, свяжитесь с нами:

e-mail: support@fastreport.ru

веб-сайт: https://www.fastreport.ru