Яндекс.Сервер 2010.09

Руководство по установке и эксплуатации

6.09.2010

Яндекс

Яндекс.Сервер 2010.09. Руководство по установке и эксплуатации. Версия 2010.09
Дата сборки документа: 6.09.2010.
Этот документ является составной частью технической документации Яндекса.
Сайт справки к сервисам Яндекса: http://help.yandex.ru

Предупреждение об исключительных правах

Контактная информация

ООО «ЯНДЕКС»
http://www.yandex.ru
Тел.: +7 495 739 7000
Email: pr@yandex-team.ru
Главный офис: 119021, Россия, г. Москва, ул. Льва Толстого, д. 16

Содержание

О данном руководстве

Данный документ содержит справочную информацию о продукте Яндекс.Сервер, его структуре, установке и настройке компонент продукта и вспомогательных утилит.

Целевой аудиторией документа являются разработчики и системные администраторы, отвечающие за установку, настройку и эксплуатацию Яндекс.Сервера.

Структура документа

Документ состоит из следующих разделов:

1. Введение: краткое описание компонент Яндекс.Сервера, их назначения и функциональных возможностей. Рассмотрены основные поисковые понятия: документ, поисковая зона, документный атрибут и т. д.
2. Поисковый модуль: описание возможностей компонента, его установки, настройки и запуска, а также директив его конфигурационного файла.
3. Индексатор: описание установки, настройки и запуска компонента, алгоритма его работы и директив конфигурационного файла.
4. Модули индексатора: подробное описание настройки индексатора для различных источников данных.
5. Настройка и использование поискового сервера: подробное описание настройки поискового модуля, метапоиска, страниц выдачи результатов поиска.
6. Язык запросов: описание основных операторов языка и алгоритмов фильтрации документов.
7. SDK внешних модулей индексатора: подробное описание структуры данных и внешних модулей, используемых в индексаторе.
8. Вспомогательные утилиты: описание различных внешних утилит, используемых при работе Яндекс.Сервера.

Введение

Основные концепции

- Компоненты Яндекс.Сервера.
- Основные возможности.
- Требования к аппаратному и программному обеспечению.
- Структура и формат конфигурационных файлов.

Компоненты Яндекс.Сервера

Яндекс.Сервер представляет собой продукт, который позволяет организовать полнотекстовый поиск документов в корпоративной сети и на веб-сайтах.

Яндекс.Сервер состоит из следующих компонентов.

Поисковый модуль

Поисковый модуль, или *поисковая машина* — это подсистема, осуществляющая разбор запроса, полученного от пользователя, поиск документов в индексных файлах, подготовленных ранее индексатором, и формирующая страницы отчета с результатами поиска. Каждой коллекции документов соответствует своя собственная поисковая машина, настройка которой задается в секции Collection конфигурационного файла.

Индексатор

Индексатор — это подсистема, анализирующая *документы* данной коллекции документов и сохраняющая информацию о них в специальных *индексных файлах*. Каждой коллекции документов соответствует свой набор индексных файлов, который хранится в отдельном каталоге, и свой собственный индексатор, настройка которого задается в секции Collection конфигурационного файла.

Конфигурационный файл

Работой Яндекс.Сервера можно управлять с помощью специальных команд, опций и директив, которые передаются программе посредством специального конфигурационного файла.

Коллекция документов

Яндекс.Сервер работает с одной или несколькими коллекциями документов, в которых осуществляется поиск. Поиск в каждой коллекции документов осуществляется независимо. Каждая коллекция характеризуется *уникальным именем* (текстовой строкой). Если используется только одна коллекция документов, ее имя может быть пустым. Правила, по которым составлена коллекция документов, указываются в секции Collection конфигурационного файла.

Источник данных

Индексатор позволяет получать документы из следующих источников: файловая система, базы данных и веб. С этой целью получение содержимого документа осуществляется с помощью отдельных модулей, по одному модулю на каждый формат хранилища документов. Эти модули, понимающие структуру хранилища документов и предоставляющие документы индексатору, в дальнейшем называются *источниками данных*. Доступна спецификация интерфейса, которая позволяет независимым производителям разработать нужные источники данных. Настройка источника данных задается в подсекции DataSrc секции Collection конфигурационного файла.

Интерпретатор формата документа

Индексатор разработан так, чтобы индексировать документы произвольного формата. С этой целью чтение документа и интерпретация его формата осуществляется с помощью отдельных модулей. Эти интерпретирующие формат документа модули в дальнейшем называются *парсерами*.

Яндекс.Сервер поставляется с уже настроенными парсерами. При необходимости, любой из парсеров может быть настроен дополнительно.

Для дополнительной настройки парсера задается подсекция DocFormat секции Collection конфигурационного файла. Более подробная информация о парсерах и их настройках размещена в разделах Секция DocFormat, Парсеры (анализаторы содержимого документа), Конфигурация HTML-парсера и Конфигурация XML-парсера.

Языковой модуль

Поисковый модуль и индексатор используют в своей работе *языковой модуль*. Основная задача языкового модуля — преобразовать слово к его словарной форме с учетом морфологии языка. Алгоритмы морфологического анализа и синтеза основаны на базовом словаре. Они позволяют находить словарную форму слова и строить гипотезы для слов, не содержащихся в базовом словаре. Поддерживается морфология русского и английского языков.

HTTP-сервер

Поисковый модуль является частью HTTP-сервера, который принимает запросы пользователей на поиск документов и возвращает результаты в виде веб-страниц.

Шаблоны страниц

Результаты поиска HTTP-сервер представляет в виде *веб-страниц*. Для их отображения используются *шаблоны*, указанные разработчиком сервиса. Методы создания шаблонов страниц описаны в разделе Формирование страниц с результатами поиска.

Примечание:
Основными логическими подсистемами Яндекс.Сервера являются индексатор и поисковый модуль. После создания индексных файлов индексатор прекращает работу, в то время как поисковый модуль после запуска находится в постоянном ожидании запросов на поиск. Для успешного старта поисковому модулю необходимы индексные файлы, созданные индексатором.

Если коллекция документов со временем изменяется за счет добавления, изменения или удаления документов, индексные файлы постепенно теряют актуальность. В этом случае необходимо время от времени запускать индексатор и обновлять индексные файлы. Обычно при этом используются настройки источника данных, позволяющие не переиндексировать неизменившиеся документы.

Основные возможности

Возможность	Описание
Возможности языка запросов	
Запрос на естественном языке.	Возможность обращаться к поисковой машине на естественном (русском или английском) языке (например, *играл ли Спартак с Динамо в марте?*).
Поиск всех вариантов слов.	Поиск идет с учетом морфологии языка, то есть может быть произведен по всем формам слова (например, если задан запрос *идти*, будут найдены документы со словами *идет*, *шел*, *шла*).
Поиск точной словоформы.	По умолчанию поиск учитывает все формы заданного слова, однако существует возможность поиска по точной словоформе.

Возможность	Описание
Корректная работа с неологизмами.	Построение словарных гипотез для слов, не содержащихся в базовом словаре (неологизмы, сленг, имена собственные и т.д.).
Поиск устойчивых словосочетаний.	Возможность находить документы, в которых заданные слова идут строго подряд.
Поиск с расстоянием.	Возможность находить документы, в которых указанные слова находятся на заданном расстоянии друг от друга, или на расстоянии, не большем (не меньшем) данного. Расстояние может быть задано в словах или в предложениях.
Логические операторы.	Поддержка логических операторов И, ИЛИ и НЕ, как в пределах документа, так и в пределах предложения.
Поиск по дате.	Возможность искать документы с датой последнего изменения в заданном диапазоне.
Поиск в зонах.	Возможность искать в определенных зонах документа, заданных при индексировании. В частности, можно искать в заголовках и аннотациях HTML-документов.
Поиск в атрибутах.	Возможность учитывать при поиске как атрибуты целого документа, заданные при индексировании, так и атрибуты отдельных частей документа. Например, можно искать документы с заданными ключевыми словами, в заданном каталоге или содержащие заданную картинку.
Поиск в найденном.	Возможность искать документы, удовлетворяющие новому запросу, среди документов, полученных в результате предыдущего запроса.
Поиск похожего документа.	Возможность искать документы, похожие на заданный документ, с использованием специального эвристического алгоритма.
Многоязыковая поддержка.	Возможность одновременного распознавания нескольких языков с учетом кодировок.
Возможности поискового сервера	
Множественные коллекции.	Поддержка нескольких независимых коллекций документов. Возможность поиска в одной или нескольких коллекциях.
Тематические разделы.	Возможность логически структурировать информацию, организовывая поиск по тематическим разделам коллекции документов.
Метапоиск.	Возможность распределенного поиска, со сливанием результатов, полученных из разных поисковых источников.
Кластеризация результатов.	Возможность сгруппировать найденные документы в соответствии с их внешними атрибутами, такими как хост или категория каталога.
Ранжирование результатов.	Возможность сортировки найденных документов по степени соответствия запросу, по дате или по одному из атрибутов, определенных при индексировании. Пользователь может повлиять на порядок сортировки, используя операторы веса и уточнения запроса.
Ссылочное ранжирование.	Возможность учитывать внешние ссылки на документ при расчете его релевантности, обеспечивать поиск документа по текстам ссылок. Реализуется за счет построения ссылочного индекса.

Возможность	Описание
Навигационный источник.	Возможность перемещать найденные по запросу документы в первые и последние позиции выдачи. Для конкретного запроса можно определить один и более URL'ов и действие по их перемещению в выдаче: сделать первым/последним. Если для запроса определено несколько URL'ов, перемещаемых "наверх", то они займут первые позиции выдачи в соответствии со своей исходной релевантностью для данного запроса. Аналогично для URL'ов, перемещаемых "вниз".
Многостраничные результаты.	Результаты поиска могут быть представлены в виде последовательности HTML-страниц, с возможностью произвольного перемещения между страницами и возможностью показывать произвольное число найденных документов на одной странице.
XML-представление.	Результаты поиска могут быть представлены в виде XML-документа с определенной схемой.
Дизайн по умолчанию.	Поисковый сервис может быть быстро запущен с использованием представления результатов поиска в дизайне по умолчанию.
Настройка дизайна.	Возможность полностью настроить дизайн страницы с результатами поиска с использованием скриптов, написанных на Perl, C++ или XSLT. Возможность показывать или не показывать различные свойства документа.
Расширенный поиск.	Возможность реализовать HTML-формы, представляющие "расширенный поиск" для пользователей, которые не хотят использовать язык запросов. Поля формы преобразуются в строку поискового запроса с помощью специальной процедуры.
Подсветка фрагментов.	Возможность выделять слова, найденные в заголовках, на странице с результатами поиска. Возможность показывать отдельные предложения документа, содержащие найденные слова, на странице с результатами поиска.
Подсветка найденных слов.	Возможность просмотреть найденный документ с выделенными поисковыми словами. Возможность пролистывать документ к следующему или предыдущему найденному слову. Статистика найденных слов для каждого документа.
Возможности индексатора	
Независимые процессы индексации и поиска.	Возможность проводить индексирование без остановки поискового сервиса.
Неограниченный размер.	Не содержит ограничений на число индексируемых документов, их размер и суммарный размер индекса.
Быстрый компактный индекс.	Полнотекстовый индекс без учета точных словоформ занимает менее одной трети от объема проиндексированного текста и создается со скоростью около 40 Мб/мин на однопроцессорной машине класса Pentium IV, 512МБ ОЗУ.
Ссылочный индекс.	См. Ссылочное ранжирование.
Множественные типы документов.	Поддержка форматов Plain text, HTML, XML, RTF, PDF, MP3, FLASH, MS Word, MS Excel, MS PowerPoint и дополнительных, определенных пользователем.
Зоны и атрибуты.	"Умное" распознавание HTML и XML-форматов, возможность гибкой настройки, позволяющей индексировать произвольные зоны и атрибуты. Возможность соотнесения с документами дополнительных "внешних" атрибутов.

Возможность	Описание
Поддержка HTTP-протокола.	Содержимое индексируемых документов может быть получено в результате запроса к HTTP-серверу. Индексируемый документ может представлять собой результат работы серверного скрипта, принимающего параметры по методу GET. Автоматическая поддержка редиректов.
Настройка HTTP-соединения.	Возможность настраивать время разрыва HTTP-соединения, используемый прокси-сервер, пароли доступа и посылаемые HTTP-заголовки для различных групп индексируемых документов.
Поддержка локальной сети.	Содержимое индексируемых документов может быть получено из файловой системы локальной сети.
Поддержка произвольных источников данных.	Содержимое индексируемых документов может быть получено обращением к произвольной базе данных, в частности, MySQL и MS SQL.
Использование гипертекстовых ссылок.	Адреса документов для индексирования могут быть получены с помощью индексирующего "паука", начинающего индексирование с одного или нескольких заданных документов, и собирающего гипертекстовые ссылки для дальнейшего индексирования.
Распознавание кодировок.	Возможность автоматически распознавать язык и кодировку индексируемого документа.
Гибкая настройка индексатора.	Возможность переиндексировать только измененные документы. Возможность оставлять в индексе документы, временно недоступные во время индексирования. Независимая настройка многочисленных параметров индексирования для разных групп документов.
Гибкие фильтры.	Возможность применения различных фильтров для того, чтобы индексировать или не индексировать документы, адреса которых удовлетворяют заданным шаблонам.

Требования к аппаратному и программному обеспечению

Поддерживаемые платформы

Операционная система	Версии	Разрядность
FreeBSD	6.3, 7.0	32/64
Linux	glibc-2.3 и выше, kernel >= 2.5.66	32/64
Windows	2000/XP/2003	32/64

Для установки Яндекс.Сервера требуется следующее аппаратное и программное обеспечение:

- 300 Мб свободного дискового пространства для установки программных модулей и конфигурационных файлов системы.
- Свободное дисковое пространство для размещения создаваемых системой файлов индекса и временных файлов.

Примечание:
При расчете объема свободного дискового пространства нужно учитывать следующее: общий объем индексных файлов может достигать от 30% до 90% суммарного объема проиндексированных документов. При переиндексации потребуется еще примерно столько же места. Кроме того, при обработке запросов поисковый сервер может создавать временные файлы, количество которых зависит от интенсивности запросов.

Структура и формат конфигурационных файлов

Конфигурационный файл определяет работу Я.Сервера. Для удобства работы конфигурационный файл можно разделить на несколько файлов.

Структура конфигурационного файла следующая:

- секции и подсекции;
- директивы – названия параметров (регистронезависимые);
- аргументы – значение параметра (регистрозависимые).

Например:

```
<Секция1>
   Директива11: Аргумент11
   Директива12: Аргумент121, Аргумент122
   <Подсекция1>
       Директива111: Аргумент1111
       Директива112: Аргумент1112
   </Подсекция1>
</Секция1>
<Секция2>
   Директива21  Аргумент211
   Директива22  Аргумент221 Аргумент222
</Секция2>
…
<СекцияN>
   ДирективаN1=АргументN11
   ДирективаN2=АргументN21, АргументN22, АргументN23, АргументN24, АргументN25
```

Секции и подсекции начинаются со строки <Имя секции> и заканчиваются строкой </Имя секции>. Внутри секций и подсекций располагаются группы директив и аргументов.

Между директивой и аргументом могут стоять: пробел " ", двоеточие ":" или равно "=". Между аргументами – пробел " " или запятая ",". Аргументы можно переносить на следующую строку используя обратный слеш "\" в конце текущей строки. В одной строке может быть не более одной директивы. В некоторых директивах конфигурационного файла могут задаваться дополнительные конфигурационные файлы.

Строки конфигурационного файла, расположенные внутри секции и начинающиеся с символов "#", "!" или ";", рассматриваются как комментарии и игнорируются. Однако, если указанные символы встречаются в середине или конце строки, они считаются значимыми. Помимо этого, в любом месте конфигурационного файла могут быть расположены комментарии в XML/HTML-стиле, начинающиеся с символов "<!-"- и заканчивающиеся символами "-->".

Пустые строки и пробельные символы в начале строки игнорируются, поэтому могут использоваться, чтобы сделать запись более ясной.

Ниже приведен фрагмент конфигурационного файла:

```
<Server>WorkDir : /yandex/server
    IPAddress : 192.168.0.1
    Host : myhost
    Port : 17000
    Threads : 5
    QueueSize : 20
    ServerLog : server.log
    <Authorization>UserName : user
        UserPassword : password
    </Authorization>
</Server>

<Collection id="abc" autostart="yes">
    IndexDir : workindex
    TempDir : newindex

    MorphFixFile :
```

```
    GlobalOptions : Update StoreArchive IgnoreWordFreqs DiscardStopWords
StoreIndexingDate
    DocProperty :
    Groups :
    <IndexLog>Filename : index.log
        Level : Config Warning Info
    </IndexLog>
    <DocFormat>MimeType : text/html
        Config : html.cfg
        ...
    </DocFormat>
    ...
    <DataSrc id="ftds">
        Name : ftds
        Config : config.cfg
        Module : libydftds.so
        Symbol : FTDS_DATASRC_LIB
        ...
    </DataSrc>
    <SearchPageTemplate>Method : binary
        Module : libreport.so
        Options :
    </SearchPageTemplate>
    ...
</Collection>

<Collection id="def" autostart="no">
    ...
</Collection>

...
```

Документы, зоны и атрибуты

- Понятие документных атрибутов.
- Поисковые зоны и атрибуты.
- Типы поисковых атрибутов.
- Конфигурирование поисковых зон и атрибутов.
- Сортировка и группировка найденных документов.
- Типы группировок.
- База группировочных атрибутов.
- Сравнение поисковых и группировочных атрибутов.

Понятие документных атрибутов

В данной документации термин *документ* используется для обозначения предмета поиска. Документ содержит сведения, нужные пользователю. Результаты поиска представляют собой список документов, возможно, разбитый на группы. Для каждого документа известен некоторый набор свойств, таких как заголовок, дата создания, размер, принадлежность к определенному разделу сайта и т.п. Такие свойства документа будут называться *документными атрибутами*, или просто атрибутами. Атрибуты могут быть непосредственно не связаны с текстом документа, видимым при его просмотре.

Каждый атрибут документа получает идентификатор в виде текстовой строки, который будет называться *именем атрибута*. Свойство документа, выражаемое атрибутом, будет называться *значением атрибута*. Документ может иметь произвольное число атрибутов. Каждый атрибут документа может иметь несколько различных значений. Для различных документов могут быть определены различные наборы атрибутов.

Пример

Рассмотрим коллекцию из двух документов, включающих литературные произведения. Каждый из них имеет документный атрибут *datecreated*, имеющий значение даты написания произведения его автором. Первый документ имеет атрибут *author* со значением "Pushkin", а у второго документа этот атрибут имеет два значения — "Sheckley" и "Zelazny", так как произведение написано в соавторстве. Наконец, у первого документа имеется атрибут *in_meter* со значением "iambus", а у второго документа этот атрибут отсутствует.

По способам использования документные атрибуты можно разделить на три группы.

- *Поисковые* атрибуты. Используются для поиска документов. Имена атрибутов используются в языке запросов. Поисковые документные атрибуты являются частным случаем зонных атрибутов и подробно рассмотрены ниже в разделе Поисковые зоны и атрибуты.
- *Группировочные* атрибуты. Используются для сортировки и группировки найденных документов. Имена атрибутов используются в полях поисковой формы запроса.
- *Свойства* документа. Используются для отображения на странице с результатами поиска. Имена атрибутов используются в аргументе функции DocProperty.

Поисковые зоны и атрибуты

Каждый документ имеет внутреннюю структуру — деление на параграфы, абзацы, заголовки и т.п. Например, формат электронного письма подразумевает наличие в нем полей *from*, *to*, служебных областей, поля сообщения и т.п. Документы принятого в интернете формата HTML также имеют внутреннюю структуру — заголовок, тело документа, ключевые слова. В теле могут присутствовать заголовки различных уровней, ссылки, картинки и т.д. Различные части структурированных таким образом документов будут называться *зонами*.

Каждая зона может иметь одно или несколько скрытых свойств, не связанных непосредственно с ее текстом. Например, зона *to* в электронном письме могла бы иметь свойство **количество адресов рассылки**, а зона *ссылка на другой документ* в HTML-документе имеет свойство **URL документа**. Такие свойства зон будут называться *зонными атрибутами*.

Основная задача парсера — выделить из документа нужный для индексирования текст. Текст, выделяемый парсером, может быть помечен как принадлежащий определенной зоне документа, или как имеющий определенные свойства (атрибуты). На основании элементов форматирования документа парсер может указать границы предложений и абзацев, а также вес данного отрывка текста. Ниже мы рассмотрим зоны и атрибуты более подробно.

Документ размечается на зоны во время индексирования в соответствии с метками, возвращаемыми парсером. Каждая зона получает уникальное имя (текстовую строку), используемое в языке запросов, и может быть предметом поиска независимо от других зон. Эти размеченные области документа будут называться *поисковыми зонами*. Каждая поисковая зона имеет точки начала и конца в теле документа. Начало и конец зон всегда приходятся на границы слов. Разные зоны могут быть вложены друг в друга.

Зонные атрибуты также могут быть помечены в качестве независимых объектов поиска. Такие атрибуты будут называться *зонными поисковыми атрибутами*, или просто атрибутами. Каждый атрибут имеет уникальное имя (текстовую строку) и значение, которое может быть различного типа, в зависимости от способа его обработки при индексировании. Зона, в общем случае, может иметь произвольное число атрибутов. Каждый атрибут может иметь несколько различных значений. Разумеется, не все атрибуты, определенные в данном массиве документов, должны быть определены для данной зоны. Язык запросов Яндекса позволяет искать в нужной зоне с нужным значением атрибута. Значения атрибутов назначаются зонам во время индексирования в соответствии с метками, возвращаемыми парсером, и хранятся в том же индексном файле, что и слова, встречающиеся в документе.

Существуют два важных частных случая зон — документ как целое и зона нулевой длины. Атрибуты документа как целого называются *поисковыми документными атрибутами*. Примерами документных атрибутов являются его автор или дата создания. Поиск по таким атрибутам является важным частным случаем зонно-атрибутивного поиска.

Назначение документных атрибутов возможно не только на основании информации, получаемой от парсера, но и прямо в источнике данных, который может получать информацию о необходимых атрибутах, например, из своего конфигурационного файла.

Зона нулевой длины введена для того, чтобы иметь возможность назначить атрибуты определенной точке внутри документа. Это может быть полезным в случае, когда документ имеет "вложенные потоки",

отличающиеся форматом или медиа-типом и не индексируемые с помощью парсера. Например, у картинок в HTML-документе может быть всплывающий сверху текст — параметр *alt* тега <*img*>. Этот текст — свойство (атрибут) точки документа, в которой расположена картинка.

Типы поисковых атрибутов

По способам распознавания и обработки различаются следующие типы атрибутов:

- Значение атрибута распознается как текст, состоящий из последовательности слов, каждое слово обрабатывается с учетом морфологии и участвует по отдельности в индексировании и поиске. Такие атрибуты будут называться атрибутами типа *TEXT*.
- Значение атрибута распознается как неделимая последовательность символов, участвующая в индексировании и поиске как единое целое. Правила морфологии к такой последовательности не применяются. Такие атрибуты будут называться атрибутами типа *LITERAL*. Для данного типа атрибутов возможен поиск в интервале значений, с учетом лексикографического сравнения.
- Значение атрибута распознается как дата или время. Такие атрибуты будут называться атрибутами типа *DATE*.
- Значение атрибута распознается как "Uniform Resource Locator". Такие атрибуты будут называться атрибутами типа *URL*.

Пример

Некоторый документ имеет атрибут abstract типа *TEXT* со значением "A general formula is derived for the main gravitomagnetic clock effect in the case of slow motion along an arbitrary elliptical orbit in the exterior field of a slowly rotating mass", атрибут field типа *LITERAL* со значением "General Relativity and Quantum Cosmology", атрибут *publication_date* типа *DATE* со значением "12 Oct 2001 00:00:02 GMT" и атрибут *gr-qc* типа *LITERAL* со значением "0110055", идентифицирующий этот документ в международной базе научных публикаций xxx.lanl.gov.

Конфигурирование поисковых зон и атрибутов

Конфигурирование зон и атрибутов является частью настройки парсера соответствующего документного формата. В конфигурации зон и атрибутов должно быть определено, какие части документа и при каких условиях следует считать поисковыми зонами, какие свойства этих зон следует считать поисковыми атрибутами и индексировать, какой они имеют тип и по каким дополнительным правилам их надо преобразовывать перед занесением в индексные файлы. Кроме того, зонам и атрибутам присваиваются имена для того, чтобы иметь возможность обратиться к ним при помощи языка запросов.

Конфигурация зон и атрибутов, встроенная в парсеры по умолчанию, достаточна для подавляющего большинства случаев.

Для парсера формата HTML по умолчанию поисковыми атрибутами считаются все пары имя/значение, перечисленные в HTML тегах <*META NAME="имя" CONTENT="значение"*>. Кроме того, если в конфигурации коллекции документов присутствуют директивы Groups или DocProperty, то также будут созданы соответствующие группировочные атрибуты и их значения будут сохранены в документном архиве.

Пример

Допустим индексируемые документы имеют формат HTML и имеют тег: <*META NAME="attr" CONTENT="value"*> Тогда в процессе индексирования будет создан поисковый атрибут с именем *attr*. При задании директивы *Groups : attr* будет создан также группировочный атрибут с таким же именем, а при задании директивы *DocProperty : attr* значение атрибута *value* будет сохранено в документном архиве.

Однако если необходимо работать с какими-либо специфическими данными, содержащимися в документах, можно изменить эту конфигурацию, описав новое поведение в конфигурационном файле соответствующего парсера. Для этого в секциях DocFormat конфигурационного файла индексатора нужно задать файл конфигурации парсера соответствующего формата. После этого нужно отредактировать конфигурацию парсера в соответствии с его документацией. Настройка парсера формата HTML описана в разделе Конфигурация HTML-парсера, а настройка парсера формата XML описана в разделе Конфигурация XML-парсера.

Особые названия зон и атрибутов

Ряд поисковых документных атрибутов попадают в индекс независимо от настроек парсеров и не требуют никаких действий по их описанию. К таким атрибутам относятся дата последнего изменения файла с документом (*date*), дата последнего переиндексирования документа (*idate*), медиатип документа (*mime*) и некоторые другие.

Функция, возвращающая заголовок документа на странице с результатами поиска, работает только в том случае, если при индексировании документа была определена поисковая зона *title*, содержащая не менее одного предложения и являющаяся границей абзаца.

Функция, возвращающая аннотацию документа на странице с результатами поиска, фактически возвращает документный атрибут *abstract*, если он был определен в настройках парсера, в противном случае — первые несколько предложений документа.

Сортировка и группировка найденных документов

В процессе формирования результатов поиска множество найденных документов всегда сортируется. По умолчанию сортировка происходит по убыванию релевантности, определяемой поисковой системой. Однако может быть задана сортировка по значению группировочного атрибута, в этом случае она может быть убывающей или возрастающей.

Имеется возможность не только по-разному сортировать список найденных документов, но и структурировать его, т.е. *группировать* этот список по значению атрибута, что делает результат поиска более обозримым и осмысленным. При *группировке* по атрибуту создается несколько множеств, называемых группами, состоящих из найденных документов. Каждая группа соответствует одному конкретному значению данного атрибута. Группы могут пересекаться, то есть если документ имеет несколько значений атрибута, он попадёт в несколько соответствующих групп. Заметим, что группировка по значениям атрибута, как правило, является осмысленной в случае, когда количество значений атрибута невелико по сравнению с количеством документов.

После выполнения группировки полученные группы сортируются либо по убыванию релевантности, либо по значению некоторого группировочного атрибута, который может отличаться от атрибута, по которому проведена группировка. При сортировке по релевантности, релевантность группы определяется особым алгоритмом, принимающим во внимание релевантности документов, входящих в эту группу. При сортировке по значению атрибута, его значение для группы определяется максимумом значения атрибута среди всех документов группы (при сортировке по убыванию), либо его минимумом (при сортировке по возрастанию). Если для одного документа имеется несколько значений атрибута, то будет использовано одно из них: минимальное или максимальное — в зависимости от "направления" сортировки.

Для одного поискового запроса можно задать сразу несколько группировок или вариантов структурирования результата. Т.е. можно получить не один "список найденного", а несколько, и каждый список будет соответствовать одной группировке. При этом способ сортировки результата поиска для всех вариантов может быть только один.

Возможность осуществить вложенную группировку (т.е. сгруппировать уже сгруппированный список) отсутствует. Нельзя, например, взять список хостов, получившийся в результате группировки по хостам, и сгруппировать его по категориям каталога. Таким образом группировки результата поиска осуществляются независимо и только по списку найденных документов.

Пример

Результаты поиска по интернету на www.yandex.ru сгруппированы по хостам и по категориям интернет-каталога. Так как документов в интернете примерно на три порядка больше, чем хостов, группировка по хостам значительно увеличивает качество и юзабилити поиска. Структурирование результата поиска по категориям интернет-каталога помогает снимать неоднозначность ответа при неточно сформулированном запросе.

Типы группировок

Значения группировочных атрибутов могут образовывать древовидную иерархию. Иерархией будет называться отношение ребёнок → родитель, заданное на множестве значений данного атрибута и отображающее его в себя, такое, что если некоторому документу соответствует значение ребёнок данного атрибута, то ему также соответствует значение родитель. При этом имеется выделенное значение "кор-

ня", обязательно равное 0, не имеющее родителя. Группировочные атрибуты, для которых задана иерархия, будут называться иерархическими.

Группировка по одному атрибуту может быть осуществлена одним из 3-х способов.

Пустая

Отсутствие группировки. Данную группировку можно понимать как группировку специального типа, при которой каждому документу соответствует одна и только одна группа. Понятие пустой группировки позволяет поддержать общность интерфейса при получении несгруппированного списка документов в порядке запрошенной сортировки.

Плоская

Группировка, при которой создаются несколько несвязных групп. Каждый документ может попасть в одну или несколько таких групп. Используется для неиерархических атрибутов.

Иерархическая

Группировка, в которой группы документов образуют иерархию в соответствии со значениями иерархического атрибута. Иерархическая группировка, в свою очередь, бывает двух типов:

- *Широкая*. Документы попадают в группы, находящиеся на один уровень ниже перед заданной группой, которая далее будет называться *текущей группой*.
- *Глубокая*. Документы попадают на самый нижний уровень, лежащий под текущей группой (в листовые группы).

Понятие *текущая группа* позволяет осуществлять навигацию по иерархически организованным структурам наподобие каталога.

Пример. Построение иерархического атрибута

Допустим документам приписана одна из категорий в следующей иерархии. В скобках указаны числовые значения группировочного атрибута.

```
Каталог (0)
    Развлечения и отдых (1)
        Спорт (3)
            Горные лыжи (8)
            Плавание (9)
        Знакомства (4)
        Путешествия (5)
            Туры (10)
            Отели (11)
            Билеты (12)
                Авиа (13)
                Ж/д (14)
    Компьютеры (2)
        Hardware (6)
        Software (7)
```

Допустим текущая группа — *Развлечения и отдых*. Тогда широкая группировка разложит документы по трём группам: *Спорт*, *Знакомства*, *Путешествия*, а глубокая группировка — по семи группам: *Горные лыжи*, *Плавание*, *Знакомства*, *Туры*, *Отели*, *Авиа*, *Ж/д*.

Допустим теперь текущей группой будет корень каталога (*Каталог*) Тогда широкая группировка возвращает две группы: *Развлечения и отдых* и *Компьютеры*, а глубокая группировка — *Горные лыжи*, *Плавание*, *Знакомства*, *Туры*, *Отели*, *Авиа*, *Ж/д*, *Hardware*, *Software*.

База группировочных атрибутов

Группировочные атрибуты документов хранятся в специальных файлах, не зависящих от основного индекса, и могут создаваться или изменяться как в процессе индексирования, так и с помощью постиндексирующих процедур, но до начала поиска. База группировочных атрибутов состоит из следующих файлов:

indexaof, indexatr

Основные файлы базы группировочных атрибутов. Во время индексирования могут быть созданы из поисковых атрибутов, имена которых перечислены в директиве Groups конфигурации индексатора.

Чтобы использовать в качестве группировочного атрибут типа *DATE*, он должен быть распознан с помощью *parse_data_integer*. После этого в группировках и сортировках будет использовано целое число, которое показывает количество секунд, прошедших после 1 января 1970 года.

Пример группировочного атрибута

Будем считать, что иерархия из примера Построение иерархического атрибута соответствует группировочному атрибуту с именем category, и все индексируемые документы имеют формат HTML и содержат тег

```
<META NAME="category" CONTENT="значение">
```

где "значение" — одно из целых чисел, указанных в примере.

В конфигурации HTML-парсера укажем:

```
<Attributes>
    ...
    category : TEXT,doc,,ignore/meta.category
</Attributes>
```

Это означает, что значение meta-тега с именем *category* будет распознано парсером как документный атрибут с именем *category*. Значение этого атрибута станет известно в результате индексирования документа, но не будет сохранено в основном индексном файле (из-за наличия аргумента *ignore*). Аргумент *ignore* позволяет уменьшить размер индексного файла за счет атрибутов, которые не нужны в языке запросов.

В конфигурации индексатора укажем директиву:

```
Groups : category
```

Это означает, что документный атрибут *category*, определенный парсером при индексировании документа, будет преобразован в одноименный группировочный атрибут, который будет сохранен в базе группировочных атрибутов.

Пример конфигурации группировочного атрибута

Допустим индексируемые документы являются законодательными актами в формате HTML. Каждому документу соответствует дата его принятия законодателем, по которой требуется искать и сортировать документы. По умолчанию поддерживается поиск и сортировка по дате последнего изменения файла на диске, которая не совпадает с нужной датой. Поэтому сохраняем дату принятия законодателем в следующих meta-тегах:

```
<META NAME="pub_date_group" CONTENT="19970127">

<META NAME="pub_date_search" CONTENT="значение">
```

где "значение" — дата в одном из следующих форматов:

```
19970127T142435 или
Mon, 27 Jan 1997 14:24:35 или
Mon Jan 27 14:24:35 1997 или
Monday, 27-Jan-97 14:24:35
```

Используются два разных тега для одинаковой даты, так как даты необходимо по-разному преобразовать для поисковых и группировочных атрибутов.

В конфигурации HTML-парсера укажем:

```
<Attributes>
    ...
    pd : LITERAL,doc,parse_data_integer,ignore/meta.pub_date_group
    pdat : DATE,doc,parse_http_expires/meta.pub_date_search
</Attributes>
```

В конфигурации индексатора укажем:

```
Groups : pd
```

Поисковый атрибут *pdat* будет сохранен в индексе в формате, позволяющем задавать поисковые запросы вида *#pdat>="19990614"*. Группировочный атрибут pd будет сохранен в базе группировочных атрибутов, что позволит сортировать документы по дате принятия.

Пример поискового/группировочного атрибута

Допустим индексируемые документы принадлежат разным интернет-сайтам и в результатах поиска требуется группировать документы по этим сайтам, то есть показывать только по одному или по несколько документов с каждого сайта.

Рассмотрим случай двух сайтов. В конфигурации индексатора определим две секции, описывающие индексирование каждого сайта:

```
<IndexedArea>HttpPrefix : http://www.site1.ru/
    Options : set host=site1
</IndexedArea>

<IndexedArea>HttpPrefix : http://www.site2.ru/
    Options : set host=site2
</IndexedArea>
```

Для каждого сайта указана начальная страница для обхода по ссылкам и литеральный поисковый атрибут с именем *host* и произвольным значением, идентифицирующим сайт.

Для преобразования этих значений в группировочные атрибуты укажем в конфигурации индексатора директиву :

```
Groups : host
```

с тем же именем атрибута. В данном примере может использоваться конфигурация парсера по умолчанию, так как значения атрибутов определяются в конфигурации индексатора, а не в теле документов. В результате работы индексатора с указанной конфигурацией поисковый атрибут *host* будет сохранен в индексе в формате, позволяющем находить все документы с данного сайта с помощью запросов вида *#host="site1"*. Одновременно будут автоматически созданы целочисленные значения группировочного атрибута host, которые будут сохранены в базе группировочных атрибутов, что позволит группировать документы по сайтам. Кроме того, будет создан файл соответствия имен *host.c2n* следующего содержания:

```
1 site1
2 site2
```

имя_атрибута.c2n

В некоторых случаях значения группировочных атрибутов, представленные целыми числами, бессмысленны с точки зрения пользователя поискового сервиса. В этом случае в результатах поиска требуется показать в качестве значений атрибутов некоторые строки-идентификаторы. Соответствие между значением атрибута и строковым идентификатором этого значения задается в файле с расширением c2n, имя которого совпадает с именем атрибута. Для каждого атрибута создаётся отдельный файл. Каждая строка указанного файла задает одно соответствие.

Эта строка имеет формат:

```
(целое число, значение атрибута)(символ табуляции)(текстовая_строка)(символ
перевода строки)
```

Если для значений атрибута не предусмотрено соответствия текстовым строкам, то c2n-файл для этого атрибута отсутствует.

Заданные в c2n-файле имена значений могут быть получены при формировании страниц с результатами поиска на C++ и Perl с помощью функции *CategName*.

В XML-представлении результатов поиска они указываются в атрибуте *attr* элемента *categ*.

Пример attr.c2n

Для иерархии из примера Построение иерархического атрибута c2n-файл имеет вид:

```
 1 Развлечения и отдых
 2 Компьютеры
 3 Спорт
 4 Знакомства
 5 Путешествия
 6 Hardware
 7 Software
 8 Горные лыжи
 9 Плавание
 10 Туры
 11 Отели
 12 Билеты
 13 Авиа
 14 Ж/д
```

имя_атрибута.c2p

Файлы c2p используются только для иерархических атрибутов и состоят из строк формата:

```
(целое число, значение "ребенок")(символ табуляции)(целое число, значение
"родитель")(символ перевода строки)
```

Каждая строка задает одно иерархическое соответствие *ребёнок → родитель*. Для каждого атрибута создаётся отдельный файл.

Если атрибут не является иерархическим, то c2p-файл для него отсутствует.

Заданные в c2p-файле значения могут быть получены при формировании страниц с результатами поиска на C++ и Perl с помощью функции *CategParent*.

В XML-представлении результатов поиска иерархия атрибутов для данного документа представлена в виде вложенных элементов *categ*.

Пример attr.c2p

Для иерархии из примера Построение иерархического атрибута c2p-файл имеет вид

```
 1 0
 2 0
 3 1
 4 1
 5 1
 6 2
 7 2
 8 3
 9 3
 10 5
 11 5
 12 5
 13 12
 14 12
```

Все описанные файлы должны находиться в том же каталоге, что и основные индексные файлы.

Замечание:
С точки зрения эффективности реализации желательно, чтобы диапазон значений группировочных атрибутов был как можно более "компактен" и его нижняя граница была недалека от нулевого значения.

Максимальное число группировочных атрибутов равно 255.

Сравнение поисковых и группировочных атрибутов

Поисковые атрибуты	Группировочные атрибуты
Используются для поиска документов.	Используются для сортировки и группировки найденных документов.

Поисковые атрибуты	Группировочные атрибуты
Имена атрибутов используются в языке запросов.	Имена атрибутов используются в полях поисковой формы, формирующей параметры запроса, и в аргументах функций, используемых при формировании страниц с результатами поиска на C++ и Perl, или в атрибутах элементов XML-представления результатов поиска.
Значения атрибутов сопоставляются с документом при его индексировании. Они либо извлекаются из текста документа в течение индексирования, либо назначаются дополнительно перед окончанием индексирования документа.	Значения атрибутов сопоставляются с документом независимо от индексирования и могут быть изменены без переиндексирования. Тем не менее, они также могут быть извлечены из текста документа при индексировании.
Значения атрибутов могут быть последовательностью слов, обрабатываемых с учетом или без учета морфологии, произвольными строками, числами или датами.	Значения атрибутов могут быть только целыми числами. Для удобства конечных пользователей каждому из этих числовых значений может быть сопоставлено строковое имя.
Документ может иметь несколько независимых значений данного атрибута.	Документ может иметь несколько значений данного атрибута, которые могут быть независимыми или образовывать иерархию.
Значения атрибутов хранятся в том же индексном файле, что и слова, встречающиеся в документе.	Значения атрибутов хранятся в специальном дополнительном файле.

Поисковый модуль

Общие сведения

- Возможности поискового модуля.
- Быстрое знакомство.
- Запуск и особенности настройки.

Возможности поискового модуля

Поиск

- Не содержит лицензионных ограничений на число поисков на сервере в целом и на число поисков в единицу времени, а также на число клиентских машин, взаимодействующих с одним сервером.
- Позволяет искать в нескольких независимых коллекциях документов.
- Предоставляет возможность распределенного поиска, с объединением результатов, полученных из разных поисковых источников.

Язык запросов

- Поддерживает все возможности языка запросов, ранжирования результатов поиска и подсветки найденных слов.
- Предоставляет возможность настроить "расширенный поиск" по тематическим разделам и сгруппировать найденные документы по определенным признакам.

Результаты поиска

- Позволяет получать результаты поиска в XML формате.
- Позволяет изменять дизайн страниц результатов поиска посредством Perl, C++ или XSLT.
- По умолчанию представляет результаты поиска во встроенном дизайне, созданном Студией Артемия Лебедева.

Индексация

- Не содержит лицензионных ограничений на число индексируемых документов, их размер или суммарный размер индекса.
- Позволяет индексировать документы через HTTP-соединение, из файловой системы и в базах данных.
- Поддерживает форматы XML, RTF, PDF, MP3, FLASH, MS Word, MS Excel, MS PowerPoint и другие.
- Позволяет индексировать группы документов с независимой настройкой параметров индексирования.
- Позволяет сделать тонкую настройку индексируемых зон и атрибутов в HTML-документе.
- Позволяет получать индексируемые документы из произвольных источников данных, реализующих API источника данных.

Быстрое знакомство

Установка и первый запуск

- Установите программу или пакет. Запустите поисковый модуль.

Windows

Установите программу в произвольную папку, которая в дальнейшем будет называться <yandex>.

Для настройки программы отредактируйте файл yandex.cfg, расположенный в каталоге <yandex>, как описано в следующем разделе.

Установите и запустите системный сервис либо вручную:

```
cd <yandex>
yandex-server.exe -i
net start Yandex.Server
```

либо с помощью batch-файла:

```
cd <yandex>
start.bat
```

Linux

Установите пакет при помощи команды:

```
rpm -i имя_rpm-пакета
```

или

```
dpkg -i имя_deb-пакета
```

Если в вашем дистрибутиве такой команды нет, распакуйте в корневой каталог tgz-архив, используя команду tar:

```
tar -zxvf имя_tgz-архива -C /
```

Для настройки программы отредактируйте файл yandex.cfg, расположенный в каталоге <yandex>, как описано в следующем разделе.

Выполните команду

```
/etc/init.d/yandex-server.sh start
```

которая запустит поисковый модуль в режиме демона.

FreeBSD

Установите пакет при помощи команды pkg_add:

```
pkg_add имя_пакета
```

либо распакуйте tgz-архив при помощи команды tar:

```
tar -zxvf имя_tgz-архива -C /
```

Для настройки программы отредактируйте файл yandex.cfg, расположенный в каталоге <yandex>, как описано в следующем разделе.

Выполните команду

```
/usr/local/etc/rc.d/yandex-server.sh start
```

которая запустит поисковый модуль в режиме демона.

Примечание:
Для установки и обновления пакетов необходимо иметь права пользователя root.

- Откройте в браузере веб-страницу http://localhost:17000/admin и нажмите на кнопку *Запустить*, расположенную рядом с заголовком *Индексатор: остановлен*. Дождитесь окончания процесса индексирования. После окончания индексирования в подчиненном каталоге <workindex> рабочего

каталога должны быть созданы шесть файлов, имеющих ненулевую длину — indexcfg, indexinv, indexkey, indexdir, indexarc, indexdat.

- На веб-странице http://localhost:17000/admin нажмите на кнопку Запустить, расположенную рядом с заголовком *Поиск: остановлен*. Теперь страница http://localhost:17000/ содержит форму для поиска.

Настройка конфигурационного файла

Для настройки поискового модуля найдите или создайте в рабочем каталоге файл yandex.cfg и отредактируйте его в любом текстовом редакторе. yandex.cfg представляет собой текстовый файл, в котором перечислены директивы, задающие параметры поискового модуля.

В простейшем случае yandex.cfg имеет следующий вид:

```
<Server>
</Server>
  <Collection>
    <DataSrc id="webds">
        Config : -w www.firma.ru/index.html
    </DataSrc>
  </Collection>
```

В качестве значения опции *-w* параметра директивы Config укажите адрес веб-страницы, которая будет проиндексирована первой. Адреса последующих страниц будут получены индексатором в результате анализа гипертекстовых ссылок. В итоге будут проиндексированы все страницы в домене www.firma.ru, на которые можно перейти по ссылкам с первой страницы.

Пример поисковой формы

Ниже приведен пример простой HTML-формы, которую вы можете разместить на страницах вашего веб-сервера для ввода данных для поиска:

```
<!-- форма поиска -->
<form name="search" method="get" action="http://www.firma.ru:17000/">
<b>Поиск:</b><br>
<input size="15" name="text" value="" maxlength="200">
<input type="submit" value=" Найти ">
</form>
```

Задайте вместо www.firma.ru имя машины, на которой у вас установлен поисковый модуль. Также задайте свой номер порта, на котором работает поисковый модуль, если он отличается от принятого по умолчанию значения 17000.

Запуск и особенности настройки

Запуск поискового модуля

Windows-версия. Поисковый модуль реализован в виде исполняемого файла yandex-server.exe и является сервисом операционной системы. Для Windows 2000/XP/2003 запуск и остановка сервиса может быть осуществлены посредством приложения **Service** панели управления или с помощью команды **NET**. В параметрах сервиса может быть указан автоматический запуск. Во всех случаях сервис должен быть предварительно установлен с помощью запуска программы с ключом **-i**. При запуске yandex-server.exe без ключей под Windows 2000/XP/2003 программа работает как обычное консольное приложение.

Unix-версия. Поисковый модуль реализован в программном модуле /usr/local/sbin/yandex-server. Для автоматического запуска этого модуля при старте операционной системы проще всего воспользоваться готовым управляющим скриптом yandex-server.sh, входящим в комплект поставки. Этот скрипт должен находиться в каталоге, содержащем скрипты запуска приложений при загрузке вашего компьютера:

- в Linux — обычно это каталог /etc/rc.d/init.d. Кроме того, на файл yandex-server.sh должны вести символические ссылки из каталогов /etc/rc.d/rc0.d, /etc/rc.d/rc2.d и /etc/rc.d/rc3.d для корректного запуска и остановки поискового модуля на разных уровнях выполнения операционной системы.
- в FreeBSD-6 или выше — это каталог /usr/local/etc/rc.d

Для остановки поискового модуля запустите скрипт yandex-server.sh с опцией stop.

Для перезапуска поискового модуля запустите скрипт yandex-server.sh с опцией restart.

Ключи командной строки

Ключи командной строки, сопровождаемые устанавливаемым значением, дублируют директивы секции Server конфигурационного файла, но имеют более высокий приоритет. Более детальные описания этих параметров даны в разделе Директивы секции Server.

Ключ	Описание
-a ***значение***	IP-адрес, на котором работает поисковый модуль. В отсутствие ключа IP-адрес может быть установлен в директиве IPAddress конфигурационного файла.
-p ***значение***	Порт, на котором работает поисковый модуль. В отсутствие ключа порт может быть установлен в директиве Port конфигурационного файла.
-h ***значение***	Хост, на котором работает поисковый модуль. В отсутствие ключа хост может быть установлен в директиве Host конфигурационного файла.
-t ***значение***	Максимальное количество одновременно выполняемых поисковых запросов. В отсутствие ключа данное значение может быть установлено в директиве Threads конфигурационного файла.
-q ***значение***	Максимальный размер очереди поисковых запросов, ожидающих начала выполнения. В отсутствие ключа данное значение может быть установлено в директиве QueueSize конфигурационного файла.
-d	Запускает сервис как обычное консольное приложение.
-v	Печатает номер версии, после чего завершается.
-i	Устанавливает yandex-server.exe как сервисное приложение операционной системы Windows, после чего завершается. *Только для Windows-версии*.
-r	Отменяет установку yandex-server.exe как сервисного приложения операционной системы Windows, после чего завершается. *Только для Windows-версии*.

Последним параметром командной строки служит путь к файлу конфигурации. Если путь к файлу конфигурации не указан, то используется имя **yandex.cfg**. Для операционных систем Windows поиск файла осуществляется в том же каталоге, в котором находится yandex-server.exe, для Unix-систем файл ищется в каталоге, из которого запущена программа.

Особенности Unix-версии

Если используется управляющий скрипт yandex-server.sh, необходимо правильно выставить переменные окружения:

- **AUTH**
  Должна соответствовать директивам**UserName**, **UserPassword** и иметь вид *UserName:UserPassword*.
- **NET_LOC**

Имя или IP-адрес хоста должны соответствовать директиве Host либо IPAddress соответственно (по умолчанию *localhost*).

- **PORT**
  Должна соответствовать директиве Port конфигурационного файла (по умолчанию *17000*).

Пример. Вызов yandex-server.sh

Ниже приведен пример вызова скрипта yandex-server.sh для секции конфигурационного файла, в которой заданы настройки веб-сервера.

```
AUTH=useradmin:asdf12345 PORT=80 NET_LOC='yandex-search.mysite.ru' yandex-server.sh start
```

Запуск/остановка индексатора и поискового сервера

Для каждой коллекции документов соответствующие индексатор и поисковый сервер могут быть запущены или остановлены независимо друг от друга.

Перед первым стартом поискового сервера необходимо запустить индексатор и дождаться, пока он завершит построение индексных файлов. Повторный запуск индексатора требуется для обновления индексных файлов, если коллекция документов изменилась за счет добавления, изменения или удаления документов.

После создания индексных файлов индексатор самостоятельно прекращает работу. В процессе своей работы индексатор также может быть остановлен внешней командой. В этом случае полного обновления индексных файлов не произойдет.

В отличие от индексатора, поисковый сервер после запуска находится в постоянном ожидании запросов на поиск. Для успешного старта поисковому серверу необходимы индексные файлы, созданные индексатором.

По умолчанию после старта поискового модуля индексатор остановлен, а поисковый сервер, при наличии индексных файлов, автоматически стартует. Чтобы поисковый сервер не стартовал в момент запуска поискового модуля, секция Collection должна иметь атрибут autostart со значением *no*.

Если при старте поискового модуля поисковый сервер для какой-либо коллекции документов не стартовал (например, по причине отсутствия индексных файлов), веб-сервер продолжает работать. На поисковые запросы по этой коллекции сервер отвечает, не смотря на то, что поиск остановлен. Однако, если атрибут autostart секции Collection имеет значение *must*, поисковый модуль автоматически завершит работу. Эта настройка может оказаться важной для инструментов автоматического мониторинга работоспособности поиска.

CGI-параметры административной страницы

Для запуска и остановки индексатора или поискового сервера служит специальная административная страница поискового модуля, имеющая адрес http://.../admin. С помощью расположенных на этой странице кнопок можно выполнить требуемые действия. Использование указанных кнопок эквивалентно следующим HTTP-запросам к поисковому модулю (если задано более одного HTTP-запроса, можно использовать любой из них):

Действие	HTTP-запрос
Начать индексирование	`/admin?id=value&action=bi` `/admin?id=value&action=startindexer`
Остановить индексирование	`/admin?id=value&action=ei` `/admin?id=value&action=stopindexer`
Начать поиск	`/admin?id=value&action=bs` `/admin?id=value&action=startsearch`
Остановить поиск	`/admin?id=value&action=es` `/admin?id=value&action=stopsearch`
Перезапустить поиск	`/admin?id=value&action=restartsearch`

Действие	HTTP-запрос
Очистить кеш поисковых запросов	`/admin?id=value&action=clearcache`
Остановить поисковый модуль	`/admin?action=shutdown`

В указанных запросах значение *value* служит для обозначения названия коллекции документов, совпадающего со значением атрибута id соответствующей секции Collection конфигурационного файла сервиса. Если имеется только одна коллекция документов с пустым именем, параметр id в административном запросе можно опустить.

В ответ на указанные запросы поисковый модуль возвращает административную страницу с новым состоянием кнопок, полученным после выполнения запрошенной команды. Если запросы выполняются из внешнего планировщика задач, удобно запретить формирование ответной HTML-страницы. Для этого нужно к запросу добавить параметр *brief* со значением *yes*.

Чтобы узнать статус индексатора и поискового сервера, задайте один из следующих HTTP-запросов:

Действие	HTTP-запрос
Запретить формирование ответной HTML-страницы	`/admin?brief=yes&action=_действие_из_предыдущей_таблицы`
Запросить статус всех коллекций поискового модуля	`/admin?action=statusall`
Запросить статус определенной коллекции поискового модуля	`/admin?id=value&action=status`

Использование статических картинок

Если дизайн страниц с результатами поиска требует использования картинок, эти картинки можно разместить на каком-либо внешнем HTTP-сервере и указать их веб-адреса в скрипте, создающем страницу результатов (см. раздел Формирование страниц с результатами поиска). Тем не менее, чтобы сделать поисковый модуль самодостаточным, предусмотрена возможность выдачи статических картинок, пути веб-адресов которых начинаются с ***/images/***. С этими адресами будут выдаваться все картинки с расширениями gif, jpg и png, расположенные либо в подчиненном каталоге ./images каталога, в котором находится выполняемый модуль yandex-server.exe для Windows, либо в каталоге, из которого запущена программа, для Unix.

Конфигурационный файл поискового модуля

- Секция Server.
- Подсекция Authorization.
- Секция Collection.

Секция Server

Конфигурация поискового модуля задается в обязательной секции **Server** конфигурационного файла. Изменения в этой секции, а также изменения в числе секций **Collection** и их атрибутах распознаются только при старте сервиса.

Конфигурации индексатора и поискового сервера задаются в секции **Collection** конфигурационного файла сервиса. Изменения в этих секциях распознаются при каждом запуске индексирования или при каждом запуске поисковой сессии.

Секция **Server** состоит из директив **IPAddress**, **Port**, **Host**, **Threads**, **QueueSize**, **WorkDir**, **ServerLog** и подсекции **Authorization**. Ни одна из этих директив и подсекций не является обязательной.

Пример. Описание секции Server

```
<Server>IPAddress :
   Port :
   Host :
   Threads :
   QueueSize :
   WorkDir :
   ServerLog :
   LoadLog :
   <Authorization>UserName :
      UserPassword :
   </Authorization>
</Server>
```

Директивы секции Server

Директива	Описание	Значение
IPAddress	Устанавливает IP-адрес, на котором работает поисковый модуль. Значение должно соответствовать одному из допустимых IP-адресов компьютера.	По умолчанию поисковый модуль работает на всех IP-адресах компьютера.
Port	Устанавливает порт, на котором работает поисковый модуль. Поисковые HTTP-запросы необходимо будет посылать на этот порт.	Значение по умолчанию:*17000*
Host	Устанавливает хост, на котором работает поисковый модуль.	Значение по умолчанию: *официальное имя хоста*. Например, если используется локальный файл hosts, это будет первая запись после IP-адреса.
Threads	Определяет максимальное количество одновременно выполняемых поисковых запросов. Если уже выполняется определяемое данной директивой количество запросов, выполнение вновь поступивших запросов откладывается до тех пор, пока не будут выполнены текущие запросы.	Значение по умолчанию: *5*
QueueSize	Определяет максимальный размер очереди поисковых запросов, ожидающих начала выполнения. В случае нулевого значения директивы максимальный размер очереди запросов бесконечен. Если начала выполнения уже ожидает определяемое данной директивой количество запросов, на вновь поступившие запросы сервер отвечает "HTTP/1.0 503 Service Unavailable" и не выполняет их. При максимальной загрузке системы уменьшение значения данной директивы приводит, с одной стороны, к сокращению времени выполнения запроса, за счет уменьшения времени ожидания в очереди, но с другой стороны, ведет к росту числа отказов.	Значение по умолчанию: *0*

Директива	Описание	Значение
WorkDir	Рабочий каталог поискового модуля. Должен быть указан абсолютный путь. Если в других директивах конфигурационного файла заданы относительные пути, они будут приведены к абсолютным относительно этого каталога.	Значение по умолчанию: *каталог, из которого запущена программа* (Unix-системы) или *каталог, в котором находится yandex-server.exe* (Windows)
ServerLog	Путь к файлу, в который будут выводиться сообщения поискового модуля, абсолютный или относительно WorkDir.	Значение по умолчанию: *yandex.log*
LoadLog	Путь к файлу, в который будут выводиться поисковые запросы пользователей, абсолютный или относительно WorkDir.	Значение по умолчанию: не задан.

Подсекция Authorization

Подсекция <Authorization> предназначена для задания параметров авторизации административного режима, предназначенного для управления веб-сервером. Авторизация проходит по схеме BASIC. В секцию входят следующие директивы:

Директивы секции Authorization

Директива	Описание
UserName	Имя пользователя.
UserPassword	Пароль пользователя.

Если секция задана, доступ к странице административного режима возможен с любого компьютера сети.

При первом обращении к странице административного режима браузер покажет стандартное диалоговое окно авторизации, в котором следует ввести указанные в данной секции имя пользователя и пароль.

Если секция отсутствует, доступ к административному режиму будет возможен только с того компьютера, на котором установлен веб-сервер.

Пример конфигурации HTTP-сервера

Ниже приведен пример секции конфигурационного файла, в которой заданы настройки веб-сервера.

```
<Server>Port : 80
    Host : yandex-search.mysite.ru
    Threads : 4
    QueueSize : 20
    <Authorization>UserName : useradmin
        UserPassword : asdf12345
    </Authorization>
</Server>
```

Секция Collection

Каждой коллекции документов соответствует одна секция **Collection**, в которой определяются документы, входящие в коллекцию, настраиваются индексатор и поисковый сервер. Директивы и подсекции секции **Collection** рассмотрены в разделах Директивы конфигурационного файла индексатора и Настройка поискового модуля.

Атрибуты секции Collection

Атрибут	Описание	Значение
id	Определяет имя коллекции документов. Это имя считается *именем индексатора* и именем *поискового сервера*, соответствующих данной коллекции документов. Имя коллекции документов может быть произвольной текстовой строкой, за исключением строки "admin", которая используется в специальных HTTP-запросах, управляющих поисковым модулем, строки "images", которая зарезервирована для виртуального каталога с картинками и строки "hl", используемой для подсвеченных документов. Если коллекций документов более одной, то каждая секция Collection должна иметь атрибут id.	
file	Указывает имя дополнительного файла, в котором определяется коллекция документов. При наличии атрибута **file** тело секции должно быть пустым, в противном случае оно должно содержать директивы, требующиеся для настройки. Дополнительный файл должен содержать корректное определение коллекции, при этом атрибуты коллекции берутся из основного конфигурационного файла. Параметр необязательный.	
autostart	Позволяет опредилить порядок работы индексатора и поиска.	• **Yes** – автоматически стартует поиск при запуске Я.Сервера. • **No** – не стартует поиск при запуске Я.Сервера. • **Must** – завершает работу Я.Сервера, если одина из коллекций не стартовала для поиска.
class	Позволяет указать тип коллекции. Используется только в том случае, если требуется выполнять индексацию и поиск в памяти. В этом случае, документы доступны для поиска с момента индексации, а при правильно организованном процессе — с момента поступления. Значение атрибута в этом случае должно быть равно **memory**.	По умолчанию значение не задано.

Примеры описания коллекций

```
Примеры описания коллекций
# Секция с безымянной коллекцией документов.
<Collection>
    # директивы секции Collection
    # ...
</Collection>
# Секция с безымянной коллекцией документов.
# Все настройки задаются в файле news_collection.cfg.
<Collection file="news_collection.cfg">
</Collection>
# Секция с коллекцией документов news.
<Collection id="news">
    # директивы секции Collection
    # ...
</Collection>
<Collection autostart="yes">
# директивы секции Collection
# ...
</Collection>
<Collection class="memory">
# директивы секции Collection
# ...
</Collection>
```

Индексатор

Общие сведения

Комплект поставки и установка

Индексатор может входить в состав программного комплекса Яндекс.Сервера или представлять собой отдельную программу Индексатор, предназначенную для создания индексных файлов, совместимых с поисковым модулем Яндекс.Сервера.

Если индексатор входит в состав Яндекс.Сервера, следуйте инструкциям, приведенным в документации к этому пакету. Установка Индексатора сводится к распаковке архива в произвольный каталог.

Выполняемый модуль Индексатора представляет собой файл с именем **dsindexer** (UNIX) или **dsindexer.exe** (Windows). По умолчанию для настройки программы используется конфигурационный файл dsindexer.cfg, который должен находиться в рабочем каталоге, то есть каталоге, из которого запущена программа. В отсутствие конфигурационного файла основные режимы работы Индексатора могут быть заданы ключами **-w**, **-f** и **-r** командной строки.

Настройка с помощью конфигурационного файла

Для настройки Индексатора найдите или создайте в рабочем каталоге файл dsindexer.cfg и отредактируйте его в любом текстовом редакторе. dsindexer.cfg представляет собой текстовый файл, в котором перечислены директивы, задающие параметры индексатора.

Индексатор позволяет индексировать документы как на веб-страницах, так и в локальных каталогах.

Для индексирования веб-страниц напишите в dsindexer.cfg следующее:

```
<Collection>
    <DataSrc id="webds">
        Config : -w www.firma.ru/index.html
    </DataSrc>
</Collection>
```

В качестве значения опции **-w** директивы Config укажите адрес веб-страницы, которая будет проиндексирована первой. Адреса последующих страниц будут получены индексатором в результате анализа гипертекстовых ссылок. В итоге будут проиндексированы все страницы в домене www.firma.ru, на которые можно перейти по ссылкам с первой страницы.

Для индексирования локальных каталогов напишите в dsindexer.cfg следующее:

```
<Collection>
    <DataSrc id="ftds">
        Config : -f ../mybook
    </DataSrc>
</Collection>
```

В качестве значения опции **-f** директивы Config укажите каталог, файлы в котором надо проиндексировать. Можно указать абсолютный путь или путь относительно рабочего каталога. Будут проиндексированы все файлы с расширениями .htm, .html и .shtml, размещенные в указанном каталоге и ему подчиненных.

Запуск

Запустите индексатор, набрав в командной строке имя выполняемого файла. Если запуск произойдет успешно, то вы получите сообщение, похожее на это:

```
Yandex Indexer 3.1.5
Use config-file 'dsindexer.cfg'.
Start indexing...
Indexing was started at Fri Nov 29 19:16:33 2002
```

А затем, через некоторое время:

```
455 documents have been added.
Total 455 documents have been indexed.
Indexing was finished at Fri Nov 29 19:17:07 2002
Index contains 455 documents with the total size 3282224 bytes
```

Это говорит о том, что индексатор нашел 455 документов, обработал их и успешно завершил работу.

Использование ключей командной строки

Для быстрого запуска Индексатора могут быть использованы ключи командной строки, описанные ниже. При наличии ключей **-f** или **-w** конфигурационный файл может отсутствовать.

```
dsindexer [-h] [-l] [-k]  [-w WEB_PAGE] [-f DIR_TO_INDEX] [-r] [-i OLD_INDEX_DIR]
[CONFIG_FILE]
  -h - вывести подсказку
  -l - напечатать текущие ограничения и конфигурацию без индексирования
  -k - сохранить предыдущий индекс
  -w - начать индексирование со страницы WEB_PAGE
       (эквивалентно директиве "StartUrls" источника данных WEBDS)
  -f - начать индексирование с локального каталога DIR_TO_INDEX
       (эквивалентно директиве "Folder/Path" источника данных FTDS )
  -r - переиндексировать предыдущий индекс
       (эквивалентно директиве "GlobalOptions Reindex" в CONFIG_FILE)
  -i - обновить предыдущий индекс в OLD_INDEX_DIR
       (эквивалентно директиве "IndexDir" в CONFIG_FILE)
CONFIG_FILE - конфигурационный файл, "dsindexer.cfg" по умолчанию.
```

Ключ	Описание
-h	Вывести подсказку.
-l	Напечатать текущие ограничения и конфигурацию без индексирования.
-w WEB_PAGE	Указывает адрес веб-страницы, которая будет проиндексирована первой. Эквивалентен директиве StartUrls конфигурационного файла источника **webds**.
-f DIR_TO_INDEX	Указывает локальный каталог, файлы в котором надо проиндексировать. Эквивалентен директиве Path секции Folder конфигурационного файла источника **ftds**.
-i OLD_INDEX_DIR	Указывает каталог, в котором могут находиться рабочие индексные файлы, созданные при предыдущем индексировании. Эквивалентен директиве IndexDir конфигурационного файла. В случае использования данного ключа указанная директива конфигурационного файла игнорируется.
-r	Полностью переиндексировать предыдущий индекс (индекс создается заново, старый удаляется).

Ключ	Описание
	Эквивалентен директиве GlobalOptions : Reindex конфигурационного файла. В случае использования данного ключа указанная директива конфигурационного файла игнорируется.
-k	Если с помощью ключа **-r** или с помощью директивы IndexDir конфигурационного файла указан каталог с индексными файлами, созданными при предыдущем индексировании, то после успешного окончания нового индексирования предыдущие индексные файлы будут по умолчанию автоматически заменены на новые. В случае задания ключа **-k** предыдущий индекс будет оставлен неизменным, а новый индекс сохранен в каталоге TempDir для временных файлов.

Алгоритм работы индексатора

Индексные файлы

В процессе выполнения запроса поисковая система читает заранее подготовленные *индексные файлы*, поэтому, чтобы попасть в результаты поиска, документ должен быть предварительно проиндексирован.

Все индексные файлы одной коллекции располагаются в одном каталоге, по умолчанию это ./workindex.

Индексные файлы разных коллекций всегда располагаются в разных каталогах.

Набор индексных файлов в одном каталоге в дальнейшем будет называться индексом. Имена индексных файлов начинаются одинаково префиксом index.

Для каждого слова в документе запоминается его позиция в виде идентификатора документа, номера предложения и номера слова в предложении. Список таких троек (словопозиций) хранится в файлах indexinv и indexkey. В этих же файлах хранятся зоны и атрибуты документов, используемые при поиске по зонам и атрибутам (например, html-заголовок или подпись к картинке), а также некоторая служебная информация. Кроме того, в файлах indexarc и indexdir по умолчанию сохраняется текст документов без элементов форматирования. Эта информация используется при поиске, если требуется получать отрывки текста документа, содержащие найденные слова. Наконец, могут быть созданы необязательные файлы indexatr и indexaof, которые содержат информацию о группировочных атрибутах документов. Наличие этих файлов позволяет группировать и сортировать найденные документы по значению атрибута.

URL и содержимое документа

Каждый проиндексированный документ характеризуется уникальным URL (Uniform Resource Locator). В процессе своей работы индексатор обращается к источникам данных, чтобы получить URL и содержимое индексируемого документа. В качестве URL документа источник данных предоставляет произвольную текстовую строку, уникально идентифицирующих документ в этом источнике.

Пользователю поискового сервиса показывается модифицированный URL с префиксом http, соответствующий скрипту поискового сервиса или ссылке на веб-сервер, предоставленной источником данных.

Область индексирования — это множество документов, индексируемых единым образом. Каждый источник данных может включать одну или несколько областей индексирования.

Области индексирования обычно задаются префиксом URL, то есть все документы, URL которых начинается с заданного префикса, принадлежат одной области индексирования.

Области индексирования могут быть вложенными. В этом случае область индексирования, заданная более длинным префиксом, наследует все свойства "родительской" области, если они явно не переопределены.

Все свойства областей индексирования, то есть параметры индексирования соответствующих документов, задаются в конфигурационном файле источника данных.

В комплект поставки входят источники данных для файловой системы, веб-страниц и баз данных, доступных через ODBC, а также базы данных MySQL. Описание протокола для модуля связи с источником данных входит в состав данной документации, поэтому такие модули могут быть разработаны независимыми поставщиками для произвольных источников данных.

Конфигурационный файл индексатора

- Секция Collection.
- Секция IndexLog.
- Секция DataSrc.
- Секция DocFormat.

Секция Collection

В этом разделе описаны директивы, относящиеся к процессу индексирования в целом. Если какая-либо директива отсутствует в секции **Collection**, для соответствующих параметров будут использованы указанные значения по умолчанию.

Пример конфигурации:

```
<Collection>WorkDir:
    IndexDir:
    TempDir:
    MorphFixFile:
    GlobalOptions:
    PortionDocCount:
    DocProperty:
    PassageProperty:
    Groups:
    NavigationSource:
    <IndexLog>Filename :
        Level :
    </IndexLog>
    <DataSrc id="dsid">
        Name :
        Module :
        Symbol :
        Config :
    </DataSrc>
    <DocFormat>MimeType :
        Config :
    </DocFormat>
</Collection>
```

Директивы секции Collection

Директива	Описание	Значение
WorkDir	Рабочий каталог. Должен быть указан абсолютный путь или путь относительно каталога, из которого запущен индексатор. Если в других директивах конфигурационного файла заданы относительные пути, они будут приведены к абсолютным относительно этого каталога. Для поискового модуля директива **WorkDir** задается в секции Server.	Значение по умолчанию:*каталог, из которого запущен индексатор*.

Директива	Описание	Значение
IndexDir	Каталог, в котором будут размещены вновь созданные индексные файлы. Должен быть указан (абсолютный путь или путь относительно **WorkDir**). Если каталог отсутствует, будет сделана попытка ее создать. Если в указанном каталоге находятся рабочие индексные файлы, созданные при предыдущем индексировании ("старый индекс"), они будут учтены при построении нового индекса в соответствии с аргументами других директив конфигурационного файла, и заменены новым индексом перед окончанием работы индексатора. В противном случае индекс будет создан заново, а ключи, относящиеся к режиму обновления индекса, проигнорированы.	Значение по умолчанию: *./workindex*. Пример: `IndexDir : myindex`
TempDir	Каталог, в котором будут храниться временные данные, необходимые индексатору (абсолютный путь или путь относительно **WorkDir**). Если каталог отсутствует, будет сделана попытка ее создать. После окончания индексирования временные файлы удаляются. Сформированный индекс помещается в **IndexDir**. **Внимание!** • **TempDir** и **IndexDir** должны быть различны. • **TempDir** и **IndexDir** должны располагаться в одном разделе файловой системы (Unix) или на одном логическом диске (Windows).	Значение по умолчанию: *./newindex*. Пример: `TempDir : /var/tmp/yandex`
MorphFixFile	Путь к файлу, который содержит правильное морфообразование слов, отсутствующих в словаре.	Значение по умолчанию: *не задан*. Пример см. ниже.
GlobalOptions	Директива, которая позволяет определить параметры индексирования. Может иметь несколько аргументов, которые разбиты на группы. Внутри каждой группы аргументов, указанных ниже, нужно выбрать один.	

Директива	Описание	Значение
PortionDocCount	Определяет максимальное число проиндексированных документов для хранящейся в памяти порции индекса. Чем больше это число, тем быстрее происходит индексирование, но объем памяти, требуемый индексатору, возрастает. При достижении максимального размера порция индекса записывается на диск в каталог, указанный в директиве **TempDir**, и в памяти создается новая порция. Все временные порции индекса сливаются в итоговый индекс на заключительном этапе индексирования.	Значение по умолчанию: *250*.
DocProperty	Директива имеет один или несколько аргументов, каждый из которых задает имя документного поискового атрибута, значение которого должно быть сохранено в архиве документов (см. раздел Документы, зоны и атрибуты). Имена и критерии, определяющие эти атрибуты, задаются в конфигурации парсера, а значения определяются во время индексирования документа (см. раздел Конфигурирование поисковых зон и атрибутов). Дополнительно к атрибутам, указанным в данной директиве, в архиве документов автоматически сохраняются особые зоны и атрибуты (см. подробности в разделе Особые названия зон и атрибутов). Значение каждого документного атрибута, сохраненного в архиве документов, может быть получено на странице с результатами поиска с помощью функции DocProperty. Суммарный объем значений атрибутов, указанных в данной директиве, заголовка, аннотации и URL документа не может превышать 8 Кб.	
PassageProperty	Директива имеет один или несколько аргументов в формате *zone#attr*, каждый из которых задает имя документного поискового атрибута *attr* поисковой зоны *zone*, значение которого должно быть сохранено в архиве документов для каждого вхождения поисковой зоны *zone*. Пример см. ниже.	Значение по умолчанию: *не задан*.
Groups	Директива имеет один или несколько аргументов, каждый из которых задает имя документного поискового атрибута, из которого должен быть автоматически создан группировочный атрибут (см. раздел Документы, зоны и атрибуты).	

Директива	Описание	Значение
	Группировочные атрибуты являются целыми числами и дают возможность сгруппировать или отсортировать найденные документы по критериям, не зависящим от текста документа (см. раздел Сортировка и группировка найденных документов). Если значения поисковых атрибутов, определенных в настройках парсера, являются целыми числами или последовательностью целых чисел, то значения группировочных атрибутов будут такими же. Для литеральных поисковых атрибутов из директивы Options значения группировочных атрибутов будут генерироваться автоматически. Группировочные атрибуты хранятся в файлах типа: имя_атрибута.c2n. В каждом аргументе директивы сразу после имени атрибута через двоеточие может быть указано одно из чисел 1, 2, 3, 4, означающее максимальное число байт, которое может занимать значение данного атрибута. Значение по умолчанию — 4. Указание меньшего числа уменьшает размер базы группировочных атрибутов.	

NavigationSource	Путь к файлу, который содержит навигационный источник.
	
	Формат файла:
	
	`Текст_запроса URL Действие`
	`Текст_запроса URL Действие`
	`...`
	
	Действие:
	
	• *1* — переместить URL в начало выдачи.
	• *-1* — переместить URL в конец выдачи.
	
	Разделитель полей: табуляция (\t).
	
	Разделитель строк: новая строка (\n).
	
	Кодировка файла: только Windows-1251.
		Значение по умолчанию: *не задан*.
		Пример см. ниже.

Пример файла MorphFixFile

Путь к данному файлу определяется в директиве MorphFixFile.

```
#Заголовок содержит номер версии и кодировку (однобайтовую или UTF-8).
#Номер версии и кодировка могут отсутствовать,
#тогда по умолчанию кодировка = yandex, версия = 0

Version: 1
Encoding: windows-1251

#Зона языка начинается с имени языка в квадратных скобках на отдельной строке.
#Каждая строка в зоне языка - формы одной леммы, через пробел или запятую.
#Восклицательный знак перед формой означает,
#что эту форму можно рассматривать как лемму.
```

```
[Russian]
!автоваз автоваза автовазу автовазом автовазе !автовазы автовазов автовазам автовазами
автовазах

[English]
!am

[Ukrainian]
!вісімдесятий !вісімдесяте вісімдесятого вісімдесятому вісімдесятим вісімдесятім !
вісімдесята вісімдесятої вісімдесятій вісімдесяту вісімдесятою !вісімдесяті вісімдесятих
вісімдесятими
```

Пример файла навигационного источника

Путь к данному файлу определяется в директиве NavigationSource.

Кодировка файла: только Windows-1251, прочие не поддерживаются.

```
Бразилия   http://dhcp172-110-red.yandex.net/suit1.html   1
КНР   http://dhcp172-110-red.yandex.net/suit3.html   -1
```

Аргументы директивы GlobalOptions

Внутри каждой группы аргументов GlobalOptions, указанных ниже, нужно выбрать один.

Аргумент	Описание	Значение
Построение ссылочного индекса		
StoreLinkIndex	При индексировании документов коллекции учитываются междокументные ссылки. Ссылочный индекс сохраняется в каталоге IndexDir в индексных файлах вида index.ref*.	Значение по умолчанию: *не задан*.
Использование старого индекса		
Update	Если в каталоге, заданном в IndexDir, существует индекс, созданный при предыдущем индексировании, то будет производиться доиндексация: новые документы будут добавляться, измененные будут переиндексироваться, удаленные будут удалены, а неизменные останутся в текущем состоянии.	
Reindex	Индекс создается заново. Старый индекс удаляется.	Значение по умолчанию: *Update*.
Сохранение документных архивов		
StoreArchive	При индексировании текст документов сохраняется без элементов форматирования. Эта информация используется во время поиска при получении отрывков текста документа, содержащих найденные слова. Архив с сохраненными текстами может иметь размер до 30-40% от суммарного размера индексируемых документов.	

Аргумент	Описание	Значение
DiscardArchive	Не сохранять текст индексируемых документов. Используется для уменьшения объема индексных файлов и увеличения скорости индексирования в случае, если показывать фрагменты текста с найденными словами не требуется. Однако, если в настройках поиска задана директива DownloadMissingPassages, отсутствующий текст будет загружен для найденных документов.	Значение по умолчанию: *StoreArchive*.
Анализ частотных распределений слов		
AnalyseWordFreqs	Задает режим анализирования частотных распределений слов в документе с целью повышения качества ранжирования.	
IgnoreWordFreqs	Анализ частотных распределений не производится.	Значение по умолчанию: *IgnoreWordFreqs*
Сохранение даты индексирования		
StoreIndexingDate	Для каждого индексируемого документа создается поисковый документный атрибут с именем *idate*, типом *DATE* и значением даты и времени последнего индексирования документа. Поисковые атрибуты обсуждаются в разделе Документы, зоны и атрибуты.	
DiscardIndexingDate	Поисковый документный атрибут *idate* не создается.	Значение по умолчанию: *StoreIndexingDate*.
Обнаружение границ предложений и абзацев на основе пунктуации		
AllowPunctBreaks	Разрешить распознавание границ предложений и абзацев по знакам пунктуации — точкам, пробелам, переводам строк и т.п.	
IgnorePunctBreaks	Границами предложений и абзацев считать только теги, разбивающие абзац в языке разметки или заданные в конфигурации парсера. Никакие естественные границы (например, точка+пробел+Большая_буква или два перевода строки и абзацный отступ внутри тега <pre> в HTML) не разбивают предложений и абзацев. Однако следует учитывать, что максимальная длина предложения ограничена, поэтому слишком длинные предложения все равно будут разбиты на несколько частей.	Значение по умолчанию: *AllowPunctBreaks*.

Пример использования директивы PassageProperty

Примечание:
Описание директивы см. выше.

Имеем некоторый документ blog.html следующего вида:

```
  <html>
  <head>
  <title>gt;Заголовок!</title>
  </head>gt;
  <body>gt;
  <div>gt;
    Принцип восприятия философски ассоциирует конфликт, несмотря на мнение авторитетов.
  </div>gt;
  <div yx:replies="yes">
    <div yx:reply="yes" yx:replier="veged">
      Современная ситуация вырождена.
    </div>

    Тут еще какой-то текст.

    <div yx:reply="yes" yx:replier="druxa">
      Информация категорически транспонирует онтологический закон внешнего мира.

      <div yx:comment="yes" yx:commenter="vsolon">
        С кем это вы сейчас разговаривали?
      </div>

      Тут снова умные фразы.

    </div>
  </div>
  </body>
  </html>
```

Допустим мы хотим, чтобы для каждого предложения, находящегося в тегах с атрибутом yx:replier либо атрибутом yx:commenter, в архиве сохранялся атрибут replier со значением, равным значению атрибута yx:replier/yx:commenter.

```
  blogparser.cfg:

<HtmlParser><Zones>
        title : title
        reply : div/is_reply
        comment : div/is_comment
    </Zones>
    <Attributes>
        is_reply : LITERAL,reply,,ignore/div.yx:reply
        replier : LITERAL,reply/div.yx:replier
        is_comment : LITERAL,reply,,ignore/div.yx:comment
        replier : LITERAL,reply/div.yx:commenter
    </Attributes>
</HtmlParser>
```

Определим желаемые свойства пассажей в файле yandex.cfg:

```
PassageProperty : reply#replier comment#replier
```

После индексации выполняем команду **tarcview workindex\index** и получаем следующий результат:

```
  ~~~~~~ Document 0 ~~~~~~

  Принцип восприятия философски ассоциирует конфликт, несмотря на мнение авторитетов.
  Современная ситуация вырождена.   replier veged
  Тут еще какой-то текст.
  Информация категорически транспонирует онтологический закон внешнего мира.      replier
druxa
  С кем это вы сейчас разговаривали?       replier druxa   replier vsolon
  Тут снова умные фразы.    replier druxa
```

Секция IndexLog

Секция позволяет указать путь к текстовому файлу, в который будет записан лог индексации, и параметры вывода информации. Секция должна включать директивы **FileName** и **Level**.

Пример секции IndexLog

```
<IndexLog>Filename : index.log
    Level : MoreDebug Verbose
</IndexLog>
```

Директивы секци IndexLog

Директива	Описание	Значение
FileName	Путь к файлу протокола индексирования, абсолютный или относительно WorkDir.	Значение по умолчанию: *стандартный поток вывода*.
Level	Уровень выдачи тестовой информации. Директива полезна при отладке и тонкой настройке конфигурации. Директива имеет произвольное число аргументов из перечисленных ниже.	

Аргументы директивы Level

Аргумент	Описание	Значение
Config	Печать конфигурации, использованной при индексировании.	
MoreConfig	Вся информация, выдаваемая в случае *Config*, плюс дополнительные сведения.	
Warning	Печать информации о некорректных случаях обработки данных, некритичных для работы программы в целом.	
MoreWarning	Вся информация, выдаваемая в случае *Warning*, плюс дополнительные сведения.	
Info	Печать информации о проиндексированных документах.	
MoreInfo	Вся информация, выдаваемая в случае *Info*, расширенная информация о проиндексированных документах и дополнительные сведения.	
Debug	Печать отладочной информации	
MoreDebug	Вся информация, выдаваемая в случае *Debug*, плюс дополнительные сведения.	
Verbose	Синоним для последовательности аргументов *MoreConfigMoreWarningMoreInfo*.	Значение по умолчанию: *не задан*.

Секция DataSrc

Секция описывает источник данных и размещается в секции Collection. Секция **Collection** может включать несколько секций **DataSrc**.

Пример. Определение секции DataSrc

```
<DataSrc id="dsid">
    Name : myname
    Module : mymodule.so
    Symbol : MYSYMBOL
    Config : myconfig.cfg
    <dsid>
        ...
    </dsid>
</DataSrc>
```

Каждая такая секция должна обязательно должна включать либо директиву **Name**, либо атрибут *id*, соответствующий одному из источников данных, поставляемых вместе с Яндекс.Сервером. Также могут присутствовать необязательные директивы **Module**, **Symbol** и **Config**.

Вложенная конфигурация

Для источников данных, поставляемых вместе с Яндекс.Сервером, описание секции **DataSrc** может быть выполнено в текущем конфигурационном файле, при этом директива **Config** будет игнорироваться.

Например:

```
<DataSrc id="ftds">
    Name : myname
    <Ftds><Folder>Path : D:\MyFiles
        </Folder>
        <Extensions>
            application/pdf: .pdf
        </Extensions>
    </Ftds>
</DataSrc>
```

Директивы секции DataSrc

Директива	Описание	Значение
Name	Задает имя источника данных, уникально идентифицирующее этот источник данных. Может состоять только из латинских букв [a-zA-Z], чисел [0-9] и символа подчеркивания '_'.	Пример: `Name : ftds`
Module	Задает локальный путь к источнику данных, абсолютный или относительно WorkDir. Применяется в случае, если источник данных реализован в виде разделяемой библиотеки.	Пример: `Module : ../bin/ydftds.dll`
Symbol	Задает имя символа, который должен быть загружен из библиотеки источника данных. Применяется в случае, если источник данных реализован в виде разделяемой библиотеки.	
Config	Задает строку инициализации, которая будет передана источнику данных при его инициализации. Формат этой строки определяется документацией к модулю связи с источником данных. Для источников данных, поставляемых вместе с Яндекс.Сервером, эта директива является обязательной и определяет путь к файлу конфигурации источника данных, формат которого описан в документации к источнику. В случае, когда конфигурация источника тривиальна, директива **Config** может содержать строку параметров, кратко определяющих начальную область индексирования. Примеры конфигурации источников см. ниже.	Значение по умолчанию: *не задан* Пример: `Config : ftds.cfg`

Пример. Тривиальная конфигурация источника ftds

```
<DataSrc id="ftds">
    Name : myname
    Config : -f C:\MyFiles
</DataSrc>
```

Пример. Тривиальная конфигурация источника webds

```
<DataSrc id="webds">
    Name : myname
    Config : -w http://localhost/docs/index.html
</DataSrc>
```

Секция DocFormat

Секция определяет необходимость дополнительной настройки **Парсера** (анализатора содержимого документа) для коллекции документов, в которой она размещена.

Для каждого типа парсера необходимо задавать отдельную секцию **DocFormat**.

Если секция не задана либо задана не полностью, для коллекции документов используются стандартные (по умолчанию) настройки парсеров.

Подробная информация о парсерах и их настройках размещена в разделах Парсеры (анализаторы содержимого документа), Конфигурация HTML-парсера и Конфигурация XML-парсера.

В секцию **DocFormat** входят обязательные директивы **MimeType** и **Config**.

Пример секции DocFormat

```
<DocFormat>MimeType : text/html
    Config : html.cfg
</DocFormat>
```

Директивы секции DocFormat

Директива	Описание
MimeType	Определяет тип настраивомого парсера. Возможные значения: • *text/html* — HTML-парсер. • *text/xml* — XML-парсер. Обязательная директива.
Config	Определяет локальный путь к конфигурационному файлу парсера, абсолютный или относительно WorkDir. Форматы конфигурационных файлов описаны в документации к соответствующим парсерам: Конфигурация HTML-парсера и Конфигурация XML-парсера. Обязательная директива.

Модули индексатора

Источники данных

Источник данных в Яндекс.Сервере представляет собой загружаемый модуль, реализующий специальный программный интерфейс (API) (см. раздел Модуль источника данных). Источник данных предоставляет индексатору/поисковому серверу содержимое документа и необходимую метаинформацию о нем.

Ниже приведена таблица модулей стандартных источников данных, поставляемых с Яндекс.Сервером.

Стандартные источники данных Яндекс.Сервера

Источник (DataSrc id)	Модуль (Module) Windows	Модуль (Module) Unix	Загружаемый символ (Symbol)	Область применимости
ftds	ydftds.dll	libydftds.so	FTDS_DATASRC_LIB	Индексирование файловых каталогов.
webds	ydwebds.dll	libydwebds.so	WEBDS_DATASRC_LIB	Индексирование веб-страниц.
odbcds	ydodbc2.dll	libydodbc2.so	ODBC_DATASRC_LIB	Индексирование данных через интерфейс ODBC.
mysqlds	ydmysql2.dll	libydmysql2.so	MYSQL_DATASRC_LIB	Индексирование баз данных MySQL.

Индексирование файловых каталогов

- Конфигурационный файл индексатора.
- Конфигурационный файл источника ftds.
- Общие директивы.
- Директивы, определяющие область индексирования.
- Директивы обработки документа.
- Конфигурация области индексирования (Секция <Folder>).
- Секция <Extensions>.
- Директива Options.

В этом разделе описаны директивы, относящиеся к процессу индексирования через стандартный источник **ftds**.

Если какая-либо директива отсутствует в конфигурационном файле источника **ftds**, для соответствующих параметров будут использованы значения по умолчанию.

Конфигурационный файл индексатора

Пример. Определение источника данных ftds

a)

```
<DataSrc>Name : myname
    Module : libydftds.so
    Symbol : FTDS_DATASRC_LIB
    Config : ftds.cfg
</DataSrc>
```

b)

```
<DataSrc>Name : myname
    Module : libydftds.so
    Symbol : FTDS_DATASRC_LIB
    <Ftds>
        ...
    </Ftds>
</DataSrc>
```

c)

```
<DataSrc id="ftds">
    Name : myname
    Config : ftds.cfg
</DataSrc>
```

d)

```
<DataSrc id="ftds">
    Name : myname
    <Ftds>
        ...
    </Ftds>
</DataSrc>
```

Конфигурационный файл источника ftds

Пример

```
<Ftds>
    DefaultOptions :
    AllowFollow :
    DisallowFollow : /\?C=[M|N|S|D];O=[A|D]
    Pipe : msxsl.exe "$1" "C:/foo/bar/transform.xsl"
    IndexPipe : test.exe
    SearchPipe : python.exe foobar.py
    <Folder inherited="no">
#         Path : C:\jdk1.5.0_05\docs\api\java\util
        Options : Set jdk=jdk15, Set group=api, Set group=java, Set group=util
    </Folder>
    <Folder inherited="no">
        Path : api/java/applet/
        Options : Set jdk=jdk15, Set group=api, Set group=java, Set group=applet
    </Folder>
    <Folder inherited="no">
        Path : api/java/awt/
        Options : Set jdk=jdk15, Set group=api, Set group=java, Set group=awt
    </Folder>
    ...
    <Extensions>
        text/html : .html, .htm, .shtml
        text/plain : .txt, .java
        text/rtf : .rtf
        application/msword : .doc
        application/pdf : .pdf
        application/vnd.ms-excel : .xls
        application/vnd.ms-powerpoint : .ppt
        application/x-shockwave-flash : .swf
        audio/mpeg : .mp3
    </Extensions>
</Ftds>
```

Общие директивы

Директива	Описание	Значение
DefaultOptions	Задает значение по умолчанию, которое будет использоваться в директиве Options в секции Folder. Аргументы этой директивы описаны в разделе Options.	Значение по умолчанию: *не задано*.

	Пример:
	`<Ftds>`
	`  DefaultOptions : utf-8`
	`  ...`
	`</Ftds>`	

Директивы, определяющие область индексирования

Директива	Описание	Значение
AllowFollow	Определяет индексатору правила обхода сайта через регулярное выражение. Обходиться будут только те документы, URL которых содержит подстроку, удовлетворяющую этому регулярному выражению.	Значение по умолчанию: *не задано*.

	Примеры:
	`! проиндексировать файлы только из архивов за 1990-1999 годы и`
	`! из каталога archives`
	`AllowFollow : /(199[0-9]|archive)/`
	
	`! проиндексировать файлы только из тех каталогов, имя`
	`! которых состоит из 4-х цифр`
	`AllowFollow : /([0-9]{4})/`	

DisallowFollow	Запрещает индексатору обходить страницы сайта, URL которых содержит подстроку, удовлетворяющие заданному регулярному выражению.	Значение по умолчанию: *не задано*.

	Пример:
	`! Исключать из индексирования файлы _index.html,`
	`! _index.htm, default.html и default.htm`
	`DisallowFollow /(_index|default)[.]htm[l]?`
	
	`! Исключать из индексирования файлы из каталогов, имя`
	`! которых состоит из 4-х цифр`
	`DisallowFollow /([0-9]{4})/`	

Директивы обработки документа

Директива	Описание	Значение
Pipe	Задает шаблон команды, которая выполняется для каждого индексируемого файла. Если в шаблоне есть $1 — оно заменяется на имя файла; если нет — имя файла добавляется к шаблону через пробел. Затем команда запускается, и ее стандартный вывод направляется в парсер вместо содержимого файла.	Значение по умолчанию: *не задано*.

Директива	Описание	Значение
	Можно отдельно задавать фильтры для индексации и для поиска директивами **IndexPipe** и **SearchPipe**, в том же формате.	

	Примеры:
	`! обработать каждый документ используя XSL шаблон`
	`Pipe : msxsl.exe "$1" C:\foo\bar\transform.xsl`
	
	`! для каждого документа выполнить скрипт foo_bar.py с аргументом;`
	`! в качестве 1 аргумента программы передать имя документа`
	`Pipe : /usr/bin/python foo_bar.py`	

IndexPipe	Задает программу-фильтр предобработки документа при индексировании.	Значение по умолчанию: *не задано*

	Пример:
	`! Обработать каждый документ используя XSL шаблон`
	`IndexPipe : transform.exe "$1" transform.xsl`	

SearchPipe	Задает программу-фильтр постобработки документа при поиске.	Значение по умолчанию: *не задано*

	Пример:
	`! Обработать каждый документ используя XSL шаблон`
	`SearchPipe : transform.exe "$1" transform.xsl`	

Конфигурация области индексирования (Секция <Folder>)

Конфигурационный файл может включать несколько секций **Folder**, каждая из которых задает область индексирования.

Директива	Описание
Path	Директива определяет локальный путь в файловой системе. Каждая секция **Folder** должна включать не более одной директивы **Path**.
Options	Параметры индексирования документов в данной области индексирования. Параметры индексирования сначала наследуются от области индексирования верхнего уровня, если такая есть, или от значения директивы DefaultOptions, или от значения по умолчанию, а затем дополняются параметрами, указанными в данной директиве. Каждая секция **Folder** должна включать не более одной директивы **Options**. Аргументы директивы описаны в разделе Директива Options.

Секция **Folder** может иметь атрибут *inherited*. Значение атрибута, равное "no", отменяет наследование значений директив и поисковых атрибутов.

Примеры:

```
<Folder>
    Path : /book/part1/
    Options : Set group=part1
</Folder>
<Folder>
    Path : /book/part1/chapter1/
    Options : Set group=chapter1
</Folder>
```

В приведенной конфигурации вторая секция наследует значение "part1" поискового атрибута *group* от предыдущей области индексирования (потому что путь /book/part1/chapter1/ входит в путь /book/part1/).

Чтобы этого не происходило, используйте:

```
<Folder inherited="no">
    Path : /book/part1/chapter1/
    Options : Set group=chapter1
</Folder>
```

Секция <Extensions>

Секция **Extensions** позволяет задать фильтр документов по их расширениям, и установить связь между значениями MimeType секции DocFormat и расширением файла.

Значения по умолчанию:

MIME	Расширение
text/html	.html, .htm, .shtml
text/plain	.txt
audio/mpeg	.mp3
text/rtf	.rtf
application/pdf	.pdf
application/msword	.doc
application/vnd.ms-excel	.xls
application/vnd.ms-powerpoint	.ppt
application/x-shockwave-flash	.swf

Директива Options

Директива **Options** позволяет задать набор документных атрибутов типа LITERAL, дополнительно к атрибутам, назначаемым парсером во время индексирования документа.

```
Options : Set name=value
```

Определение документного атрибута для данной области индексирования

```
Options : Unset name=value
```

Отмена документного атрибута для данной области индексирования.

Использование данных аргументов позволяет включить документы в определенные тематические разделы на основании структуры каталогов файловой системы, в которых находятся документы. Альтернативно, во время индексирования документы могут получить поисковые документные атрибуты в соответствии с их содержанием. См. обсуждение в разделе Документы, зоны и атрибуты.

Строка name=value не должна включать пробелы. Чтобы удалить для данной области индексирования все унаследованные атрибуты, используйте атрибут *inherited* секции Folder.

Пример:

```
<Folder inherited="no">
```

Директива **Options** позволяет задать кодировку, используемую в документах.

recognize

Всегда распознавать кодировку символов автоматически.

<значение>

Использовать одно из указанных ниже в таблице кодировок значений.

Кодировка	Обозначение
WinCyrillic	*windows-1251*, *cp1251*
MacCyrillic	*MacCyrillic*, *MacRussian*
DOSCyrillic	*IBM855* или *cp855*
DOSCyrillicRussian	*IBM866*, *cp866*
ISOLatinCyrillic	*ISO-8859-5*, *iso-ir-144*
WinLatin1	*windows-1252*, *cp1252*
WinLatin2	*windows-1250*, *cp1250*
KOI8R	*KOI8-R*, *csKOI8R*
ISO8859_2	*iso-2*, *iso_8859-2*
UTF8	*utf8*, *utf-8*

Значение по умолчанию: *recognize*

Индексирование веб-страниц

- Конфигурационный файл индексатора.
- Конфигурационный файл источника webds.
- Директивы, определяющие области индексирования.
- Конфигурация области индексирования (Секция <IndexedArea>).
- Директива Options.
- Конфигурация HTTP-запросов (Секция <HttpOptions>).
- Примеры настройки источника.
- Правила индексирования, не описываемые в конфигурационном файле.

В этом разделе описаны директивы, относящиеся к процессу индексирования через стандартный источник **webds**. Если какая-либо директива отсутствует в конфигурационном файле, для соответствующих параметров будут использованы указанные значения по умолчанию.

Механизм индексирования с получением новых ссылок из ранее проиндексированных документов ("сетевой паук") работает, только если определены атрибуты *link*, и в качестве ссылок используются значения этих атрибутов.

Конфигурационный файл индексатора

Пример. Определение источника данных webds

a)

```
<DataSrc>Name : myname
    Module : libydwebds.so
    Symbol : WEBDS_DATASRC_LIB
    Config : webds.cfg
</DataSrc>
```

b)

```
<DataSrc>Name : myname
    Module : libydwebds.so
    Symbol : WEBDS_DATASRC_LIB
    <Webds>
        ...
    </Webds>
</DataSrc>
```

c)

```
<DataSrc id="webds">
    Name : myname
    Config : webds.cfg
</DataSrc>
```

d)

```
<DataSrc id="webds">
    Name : myname
    <Webds>
        ...
    </Webds>
</DataSrc>
```

Конфигурационный файл источника webds

Пример

```
<Webds>
    AllowFollow :  /(199[0-9]|archive)/
    DisallowFollow  : /(_index|default)[.]htm[l]?
    AllowIndex  :
    DisallowIndex  :
    IgnoreCGIParameters :
    DefaultAreaOptions : UrlCaseFold FindLinks
    DefaultHttpPrefix : http://www.host.ru/
    StartUrls : http://www.host.ru/
    IpMask :
    <IndexedArea>HttpPrefix : http://www.company.ru/path
        Options : GetHttp:httpOptions
    </IndexedArea>
    <IndexedArea>HttpPrefix : http://www.company.ru:8080/documents
        FilePrefix : C:\path\to\www\company\ru\documents
        <HttpOptions><Authorization>UserName : user
                UserPassword : password
            </Authorization>
        </HttpOptions>
    </IndexedArea>
    <HttpOptions name="httpOptions">
        Timeout : 50
        Delay : 0
        ProxyUrl : http://proxy.company.ru:8080
        <Headers>
            User-Agent : Yandex.Server/3.x
            From : N/A
            Accept-Language : ru,*;q=0.1
        </Headers>
        <Authorization>UserName : user
            UserPassword : password
        </Authorization>
    </HttpOptions>
</Webds>
```

Директивы, определяющие области индексирования

Индексатор начинает свою работу с получения начального списка URL документов, подлежащих индексированию, поэтому необходимо, чтобы список начальных URL был явно задан в ключе **StartUrls** или чтобы присутствовала хотя бы одна из секций IndexedArea.

<table>
<tr><th>Директива</th><th>Описание</th><th>Значение</th></tr>
<tr><td>StartUrls</td><td>Задает один или несколько URL документов, с которых индексатор начинает индексирование (указывать начальный префикс “http://” необязательно).

Если в секциях IndexedArea и директиве DefaultAreaOptions не указано по-другому, будет реализовано следующее поведение по умолчанию. URL других документов, кроме указанных, будут получены в результате распознавания гипертекстовых ссылок в тексте уже проиндексированных документов. Будут проиндексированы только документы из тех же веб-страниц, в которых лежат указанные начальные URL, и документы из подчиненных страниц. Содержимое документов будет получено по протоколу HTTP, кодировка документов будет распознана автоматически. При переиндексировании будут повторно индексироваться только новые и изменившиеся документы, а недоступные (удаленные) документы будут удалены из индекса.</td><td>Значение по умолчанию: не задан.</td></tr>
<tr><td colspan="2">Пример 1:

<code>StartUrls : www.host.name/</code>

Будут проиндексированы все документы сайта http://www.host.name, на которые можно перейти с главной страницы по гипертекстовым ссылкам за один или несколько "кликов". Данная конфигурация полностью эквивалентна следующей (см. IndexedArea):

<pre><IndexedArea>
    HttpPrefix : www.host.name/
</IndexedArea></pre>Пример 2:

<code>StartUrls : www.host.name/docs/doc.html</code>

Будут проиндексированы все документы в каталоге http://www.host.name/docs/, на которые можно перейти с документа http://www.host.name/docs/doc.html. Документы, URL-ы которых начинаются не с http://www.host.name/docs/, проиндексированы не будут.

Пример 3:

<pre>StartUrls : http://www.host.name/news/, www.host.name/
conference/conf.html</pre>Будут проиндексированы документы, URL-ы которых начинаются с http://www.host.name/news/ или с http://www.host.name/conference/, и на которые можно перейти за один или несколько "кликов" хотя бы с одной из страниц http://www.host.name/news/ или http://www.host.name/conference/conf.html.</td></tr>
</table>

Директива	Описание	Значение
IgnoreCGIParameters	Определяет один или несколько CGI параметров, которые будут игнорироваться индексатором при обходе. Директива позволяет игнорировать часто меняющиеся параметры в URL, не влияющие на контент страницы. Обычно это сессионные параметры.	Значение по умолчанию: *не заданы*.

	Пример:
	
	`IgnoreCGIParameters : id sid`
	
	При индексировании документов из всех полученных URL будут предварительно удаляться параметры *id* и *sid*.
	
	`http://somehost/somepath/somescript?`
	`id=someid&param1=value1&sid=12343454567`
	
	Для такого URL будут удалены параметры *id* и *sid*, в результате чего индексатор получит URL:
	
	`http://somehost/somepath/somescript?param1=value1`	

DisallowFollow	Запрещает индексатору обходить страницы сайта, URL которых содержит подстроку, удовлетворяющие заданному регулярному выражению.	Значение по умолчанию: *не задан*.

	Примеры:
	
	`! Исключать из обхода индексатором файлы _index.html,`
	`! _index.htm, default.html и default.htm`
	`DisallowFollow : /(_index|default)[.]htm[l]?`
	
	`! Исключать из обхода индексатором скрипт /lists/`
	`showfolder.asp, если`
	`! первым в списке cgi-параметров идет параметр с именем`
	`fid`
	`DisallowFollow : /lists/showfolder.asp[?]fid=.*`	

AllowFollow	Определяет индексатору правила обхода сайта через регулярное выражение. Обходиться будут только те документы, URL которых содержит подстроку, удовлетворяющую этому регулярному выражению.	Значение по умолчанию: *не задан*.

	Примеры:
	
	`! обходить индексатором файлы только из архивов за`
	`1990-1999 годы и`
	`! из каталога archives`
	`AllowFollow : /(199[0-9]|archive)/`
	
	`! обходить индексатором файлы только из тех каталогов, имя`
	`! которых состоит из 4-х цифр`
	`AllowFollow : /([0-9]{4})/`	

DisallowIndex	Запрещает индексировать страницы сайта, URL которых содержит подстроку, удовлетворяющую заданному регулярному выражению. Такие URL будут обходиться индексатором с целью получения новых URL для дальнейшего обхода, но не будут индексироваться.	Значение по умолчанию: *не задан*.
	Примеры:	

<table>
<tr><th>Директива</th><th>Описание</th><th>Значение</th></tr>
<tr><td></td><td colspan="2">

```
! Исключать из индексирования файлы _index.html,
! _index.htm, default.html и default.htm
DisallowIndex : /(_index|default)[.]htm[l]?

! Исключать из индексирования скрипт /lists/
showfolder.asp, если
! первым в списке cgi-параметров идет параметр с именем
fid
DisallowIndex : /lists/showfolder.asp[?]fid=.*
```

</td></tr>
<tr><td rowspan="2">AllowIndex</td><td>Определяет правила индексирования сайта через регулярное выражение. Проиндексированы будут только те документы, URL которых содержит подстроку, удовлетворяющую этому регулярному выражению.</td><td>Значение по умолчанию: не задан.</td></tr>
<tr><td colspan="2">Примеры:

```
! проиндексировать файлы только из архивов за 1990-1999
годы и
! из каталога archives
AllowIndex : /(199[0-9]|archive)/

! проиндексировать файлы только из тех каталогов, имя
! которых состоит из 4-х цифр
AllowIndex : /([0-9]{4})/
```

</td></tr>
<tr><td rowspan="2">DefaultHttpPrefix</td><td>Задает префикс URL по умолчанию, относительно которого может быть задан аргумент директивы HttpPrefix в секциях IndexedArea. Дает возможность задавать относительные URL в секциях IndexedArea.</td><td>Значение по умолчанию: http://127.0.0.1/.</td></tr>
<tr><td colspan="2">Пример:

```
DefaultHttpPrefix : myhost.ru
```

</td></tr>
<tr><td rowspan="2">DefaultAreaOptions</td><td>Задает значение по умолчанию, которое будет использоваться в директиве Options в секции IndexedArea. Это же значение задает способ индексирования при использовании директивы StartUrls, если определяемые ею веб-страницы не входят в дерево, определенное в секциях IndexedArea.

Аргументы этой директивы описаны в разделе Options.</td><td>Значение по умолчанию: use_content_type update.</td></tr>
<tr><td colspan="2">Пример:

```
DefaultAreaOptions : windows-1251
```

</td></tr>
<tr><td rowspan="2">IpMask</td><td>Определяет индексатору диапазон IP-адресов, разрешенных для обхода.

Позволяет решить задачу индексации всех сайтов в пределах заданного диапазона IP-адресов, если на эти сайты индексатору удалось найти ссылки. Отменяет ограничения, определенные директивой HttpPrefix (если определена в секции IndexedArea).</td><td></td></tr>
<tr><td colspan="2">Пример:</td></tr>
</table>

Директива	Описание	Значение
	`IpMask : 192.168.[1-10].* 10.*.*.* 192.*.*.1` В примере показано, как задать индексатору обход IP-адресов, начинающихся на 192.168 с третьим числом от 1 до 10 включительно и произвольным четвертым числом, начинающихся на 10 и начинающихся на 192 и заканчивающихся числом 1.	

Конфигурация области индексирования (Секция <IndexedArea>)

Секция **IndexedArea** задает область индексирования. В секции **Webds** может быть указано несколько секций **IndexedArea**.

Секция **IndexedArea** может иметь атрибут *inherited*. Значение атрибута, равное "no", отменяет наследование значений директив и поисковых атрибутов.

Пример:

```
<IndexedArea>HttpPrefix : http://myhost/mysite/theme1/
    Options : Set group=theme1
</IndexedArea>
<IndexedArea>HttpPrefix : http://myhost/mysite/theme1/theme2/
    Options : Set group=theme2
</IndexedArea>
```

В приведенной конфигурации вторая секция наследует значение "theme1" поискового атрибута *group* от предыдущей области индексирования.

Чтобы этого не происходило, используйте:

```
<IndexedArea inherited="no">
    HttpPrefix : http://myhost/mysite/theme1/theme2/
    Options : Set group=theme2
</IndexedArea>
```

Директива	Описание	Значение
HttpPrefix	Префикс URL документов, абсолютный или относительно пути, заданного в DefaultHttpPrefix. Все документы, имеющие данный префикс, индексируются по правилам, указанным в Options. Если указан относительный путь, изменение директивы DefaultHttpPrefix при переиндексировании не вызывает переиндексирования данной области индексирования.	

	Пример:
	`<IndexedArea>`
	`  HttpPrefix : /`
	`</IndexedArea>`	

FilePrefix	Локальный путь, соответствующий значению HttpPrefix. Дает возможность получать содержимое документов с помощью чтения файлов. Должен быть указан абсолютный путь или путь относительно рабочего каталога индексатора. Обязательно наличие директивы HttpPrefix либо DefaultHttpPrefix.	Значение по умолчанию: *не задан*.
	Пример:	

Директива	Описание	Значение
	<IndexedArea> HttpPrefix : www.myhost.ru/ FilePrefix : C:\Inetpub\wwwroot </IndexedArea>	
Options	Параметры индексирования документов в данной области индексирования. Параметры индексирования сначала наследуются от области индексирования верхнего уровня, если такая есть, или от значения директивы DefaultAreaOptions, или от значения по умолчанию, а затем дополняются параметрами, указанными в данной директиве. Аргументы этой директивы описаны в разделе Директива Options.	Значение по умолчанию: *не задан*.
	Пример: <IndexedArea> HttpPrefix : / FilePrefix : C:\Inetpub\wwwroot Options : windows-1251 </IndexedArea>	
AllowFollow	Директивы имеют ту же семантику, что и для секции webds.	
AllowIndex		
DisallowFollow		
DisallowIndex		
IpMask		

Директива Options

В этом разделе описаны аргументы директивы **Options**, которая может встречаться в секции IndexedArea, а также директивы DefaultAreaOptions из главной секции конфигурационного файла источника. С помощью аргументов директивы **Options** можно задать следующие параметры областей индексирования.

Аргумент	Описание
Режим индексирования поддоменов	
IndexSubdomains	Сообщает индексатору о необходимости индексировать поддомены стартового домена. Эта опция отменяет ограничение, накладываемое параметром StartUrls или HttpPrefix. В результате будут проиндексированы все документы домена и всех поддоменов, доступные из стартового URL.
Режим получения URL документа	
NoUrlCaseFold	Считать URL документов регистрозависимыми, в соответствии со стандартом.
UrlCaseFold	Получать URL документов регистро-независимыми, например, при индексировании документов с веб-серверов под Windows.
IndexFollow	Индексировать документы и распознавать гипертекстовые ссылки для получения URL-ов новых документов.
IndexNofollow	Индексировать документы, но не распознавать гипертекстовые ссылки для получения URL-ов новых документов.

Аргумент	Описание
NoindexFollow	Не индексировать документы, но просматривать их и распознавать находящиеся в них гипертекстовые ссылки для получения URL-ов новых документов.
UseDirectUrls	Сообщает индексатору о необходимости сохранять в индекс прямую ссылку на документ. По умолчанию в индекс сохраняется ссылка на сохраненную копию документа. Этот параметр нужно использовать, если предполагается переход пользователя на сайт со страницы с результатами поиска. По умолчанию параметр отключен.
Режим использования мета-тега robots	
AllowMetaRobots	Учитывать при индексировании содержимое мета-тега *robots*. Подробнее об этом написано в разделе Мета-тег robots. Используется по умолчанию.
IgnoreMetaRobots	Игнорировать мета-тег robots.
Режим получения содержимого документа	
GetHttp:configid	Получать содержимое документов с помощью HTTP-протокола, посылая заголовки, сконфигурированные в секции HttpOptions, имеющей идентификатор *configid*. Данный идентификатор определяет значение атрибута *name* секции HttpOptions в текущем конфигурационном файле. Этот аргумент используется только в секции IndexedArea.
Режим обновления индекса	
При первом индексировании все документы считаются новыми. Рассмотрим повторное индексирование с использованием существующего индекса. Имеющиеся в нем документы будут называться старыми, остальные индексируемые документы — новыми. Старые документы можно разделить на три группы — изменившиеся, неизменившиеся и недоступные. Изменившимся считается документ, текущее время модификации которого больше, чем время модификации во время предыдущего индексирования. Недоступными считаются документы, если попытка получить их содержимое по URL, известному от предыдущего индексирования, заканчивается неудачей. Остальные документы считаются неизменившимися. Старые документы можно удалять из индекса, переиндексировать или оставлять в индексе без переиндексирования.	
<режим обновления>	Использовать один из указанных ниже в таблице режимов обновления.
Для удобства наиболее часто встречающиеся режимы обновления индекса можно задать с помощью аргументов:	
Update	Убирать из индекса данные о недоступных документах и индексировать заново новые и изменившиеся документы, не индексировать неизменившиеся документы. Эквивалентен заданию *indnew*, *indmod*, *skipold*, *remmiss*. Используется по умолчанию.
Reindex	Убирать из индекса недоступные документы и индексировать заново все существующие, независимо от того, изменились ли они со времени предыдущего индексирования. Эквивалентен заданию *indnew*, *indmod*, *indold*, *remmiss*.
Noremove	Индексировать документы в данной области индексирования, но не убирать из индекса недоступные документы. Этот флаг полезен при индексировании временно недоступных документов. Эквивалентен заданию *indnew*, *indmod*, *skipold*, *skipmiss*.
Addonly	Убирать из индекса удаленные документы и индексировать заново только новые документы, проиндексированные ранее документы не переиндексировать, даже если время их изменения увеличилось. Эквивалентен заданию *indnew*, *skipmod*, *skipold*, *remmiss*.

Аргумент	Описание
Noindex	Не индексировать документы из данной области индексирования, убирать из индекса все ранее проиндексированные документы из этой области. Эквивалентен заданию *skipnew*, *remmod*, *remold*, *remmiss*.
Skip	Не индексировать документы из данной области индексирования, но сохранить в индексе ранее проиндексированные документы из этой области. Эквивалентен заданию *skipnew*, *skipmod*, *skipold*, *skipmiss*.
При получении содержимого документов через HTTP-соединение можно использовать следующие аргументы:	
SkipDisconnected	Не удалять из индекса документы, принадлежащие веб-серверу, с которым не удалось установить HTTP-соединение. Это более слабый вариант *Noremove*, действующий только для недоступных веб-серверов.
RemoveDisconnected	Удалять из индекса документы, принадлежащие веб-серверу, с которым не удалось установить HTTP-соединение.
Reconnect	В случае обрыва HTTP-соединения с веб-сервером пытаться установить его для каждого последующего документа.
Кодировка символов, используемая в документах	
recognize	Всегда распознавать кодировку символов автоматически. Используется по умолчанию.
use_content_type	В случае документов, получаемых по протоколу HTTP, считать кодировкой документа значение, указанное в заголовке Content-Type. Если заголовок отсутствует или в нем не указана кодировка, распознавать кодировку с помощью анализа текста документа.
<значение кодировки>	Использовать одно из указанных ниже в таблице кодировок значений.
Обнаружение границ предложений и абзацев на основе пунктуации	
AllowPunctBreaks	Разрешить распознавание границ предложений и абзацев по знакам пунктуации — точкам, пробелам, переводам строк и т.п. Используется по умолчанию.
IgnorePunctBreaks	Границами предложений и абзацев считать только теги, разбивающие абзац в языке разметки или заданные в конфигурации парсера. Никакие естественные границы (например, точка+пробел+Большая_буква или два перевода строки и абзацный отступ внутри тега *<pre>* в HTML) не разбивают предложений и абзацев. Однако следует учитывать, что максимальная длина предложения ограничена, поэтому слишком длинные предложения все равно будут разбиты на несколько частей.
Набор атрибутов документа	
Set <имя>=<значение>	Включить область индексирования в раздел.
Unset <имя>=<значение>	Исключить область индексирования из раздела.
Указанные аргументы позволяют задать поисковые документные атрибуты типа LITERAL, дополнительно к атрибутам, назначаемым парсером во время индексирования документа. Использование данных аргументов позволяет включить документы в определенные тематические разделы на основании структуры веб-страниц, в которых находятся документы. Альтернативно, во время индексирования документы могут получить поисковые документные атрибуты в соответствии с их содержанием. См. обсуждение в разделе Документы, зоны и атрибуты.	

Аргумент	Описание
Строка *имя=значение* не должна включать пробелы. Чтобы удалить для данной области индексирования все унаследованные атрибуты, используйте атрибут *inherited* секции IndexedArea. Пример: `<IndexedArea inherited="no">`	

Режимы обновления индекса

Следующая таблица представляет значения аргументов, задающие соответствующий режим обновления.

Тип документа	Индексировать	Не индексировать, оставить	Не индексировать, удалить
Новый	*indnew*	*skipnew*	
Изменившийся	*indmod*	*skipmod*	*remmod*
Неизменившийся	*indold*	*skipold*	*remold*
Недоступный		*skipmiss*	*remmiss*

Кодировка символов, используемая в документах

В таблице приведены обозначения, используемые при принудительном задании кодировки.

Кодировка	Обозначение
WinCyrillic	*windows-1251*, *cp1251*
MacCyrillic	*MacCyrillic*, *MacRussian*
DOSCyrillic	*IBM855* или *cp855*
DOSCyrillicRussian	*IBM866*, *cp866*
ISOLatinCyrillic	*ISO-8859-5*, *iso-ir-144*
WinLatin1	*windows-1252*, *cp1252*
WinLatin2	*windows-1250*, *cp1250*
KOI8R	*KOI8-R*, *csKOI8R*
ISO8859_2	*iso-2*, *iso_8859-2*
UTF8	*utf8*, *utf-8*

Конфигурация HTTP-запросов (Секция <HttpOptions>)

При обращении к веб-серверу индексатор передает ему HTTP-запрос, который по умолчанию выглядит следующим образом:

```
GET _относительный_урл_документа_ HTTP/1.1
Host: _имя_хоста_с_которого_запрашивается_документ_
Connection: Keep-Alive
From: N/A
User-Agent: Yandex.Server/_номер_версии_Yandex.Server_
Accept: text/html _или_ text/plain
Accept-Language: ru; q=1.0, *; q=0.01
If-Modified-Since: _дата_модификации_при_последнем_индексировании_
```

Последний заголовок посылается при повторном индексировании, если документ уже был проиндексирован ранее, а в опциях настройки источника указано "переиндексировать, только если документ изменился".

Иногда требуется модифицировать посылаемые по умолчанию заголовки или добавить новые заголовки. Так, например, для сайтов, требующих авторизации, HTTP-запрос должен содержать дополнительную информацию о полномочиях пользователя. Также бывает нужно использовать прокси-сервер или установить время задержки между запросами, чтобы не загружать веб-сервер.

Имеется гибкий механизм настройки HTTP-запросов. Эта настройка может быть различной для разных областей индексирования. Каждая настройка запоминается либо в секции **<HttpOptions>** конфигурационного файла источника и имеет идентифицирующий атрибут name, либо задается как подсекция <IndexedArea>. Значение атрибута *name* указывается для каждой области индексирования в директиве Options секций IndexedArea конфигурационного файла источника.

Секция **<HttpOptions>** состоит из необязательных директив **Timeout**, **Delay** и **ProxyUrl** и двух необязательных секций **Headers** и **Authorization**.

Директива	Описание	Значение
Директивы основной секции		
Timeout	Максимальное время ожидания ответа веб-сервера в секундах.	Значение по умолчанию: *150*.
Delay	Время задержки перед запросом следующего документа в микросекундах. Директива необходима для уменьшения нагрузки на сервер, время индексирования при этом, естественно, увеличивается.	Значение по умолчанию: *0*.
ProxyUrl	Позволяет использовать в HTTP-запросе указанный прокси-сервер. Директива должна определять полный URL прокси-сервера, начинающийся с http и содержащий номер порта, если он отличается от *80*.	Значение по умолчанию: *не задан*.

	Если указан прокси-сервер, запрос
	`GET _относительный_урл_документа_ HTTP/1.1`
	`Host: _имя_хоста_с_которого_запрашивается_документ_`
	будет заменен на
	`GET _абсолютный_урл_документа_ HTTP/1.1`
	и HTTP-соединение будет устанавливаться с прокси-сервером, а не хостом, на котором расположен документ. Прокси-серверы, требующие отдельной авторизации со стороны пользователя, не поддерживаются в данной версии.	

Секция Headers		
Позволяет задавать любые HTTP-заголовки в формате `Имя: Значение` В этом примере будет добавлен HTTP-заголовок с именем *Имя* и значением *Значение*. Правила хорошего тона при индексировании независимых ресурсов требуют задания HTTP-заголовков: *User-Agent*, который содержит идентифицирующую информацию о программе-клиенте, пославшей запрос, и *From*, который должен содержать электронный адрес администратора программы-клиента в формате, определенном в RFC 822. Если ключи *User-Agent* и/или *From* отсутствуют, будут посланы заголовки по умолчанию, указанные выше. Кроме того, всегда будут посланы заголовки *Connection* и, если необходимо, *If-Modified-Since*, поэтому их не следует указывать в данной секции.		
Секция Authorization		
Носит вспомогательный характер и позволяет удобным способом добавлять HTTP-заголовок *Authorization*, обеспечивающий проверку полномочий клиента на доступ к данным по схеме BASIC. В секцию входят следующие директивы:		
UserName	Имя пользователя.	
UserPassword	Пароль пользователя.	
Имя и пароль кодируются по base64. Например, комбинация следующих ключей в секции **Authorization**		

Директива	Описание	Значение

```
<Authorization>
    UserName : yandex
    UserPassword : asdf12345
</Authorization>
```

эквивалентна заданию директивы

```
<Headers>
    Authorization : Basic eXNpdGU6YXNkZjEyMzQ1
</Headers>
```

в секции **Headers** и приведет к включению соответствующего HTTP-заголовка.

Пример. Конфигурация HTTP-запросов

Ниже приведен отрывок из конфигурационного файла источника, задающий конфигурацию HTTP-запросов при индексировании хоста www.host.ru.

```
<IndexedArea>HttpPrefix : www.host.ru
    Options : GetHttp:myhttp
</IndexedArea>
<HttpOptions name="myhttp">
    Timeout : 150
    Delay : 0
    ProxyUrl : http://proxy.host.ru:8080
    <Authorization>UserName : yandex
        UserPassword : asdf12345
    </Authorization>
    <Headers>
       User-Agent : Yandex.Server.MyHost/3.0
       From : admin@host.ru
       Accept-Language : ru, *;q=0.1
       MyHeader : TestStroka
    </Headers>
</HttpOptions>
```

Примеры настройки источника

Настройка области индексирования

```
<Webds>
    ! Индексируемый каталог
    <IndexedArea>HttpPrefix : www.company.ru
    </IndexedArea>
</Webds>
```

Настройка при обходе сайта по ссылкам

```
<Webds>
    ! Начальная ссылка
    StartUrls : www.company.ru/
</Webds>
```

Настройка для получения документов с использованием файловой системы

```
<Webds>
    ! Индексируемый каталог
    <IndexedArea>HttpPrefix : www.company.ru
        FilePrefix : /path/to/www.company.ru/data
    </IndexedArea>
</Webds>
```

Правила индексирования, не описываемые в конфигурационном файле

Исключение частей HTML-файлов из индексирования

Часто встречаются ситуации, когда необходимо исключить из индексирования не весь документ целиком, а только его часть. Добиться этого можно, немного подправив HTML-код страницы. Весь текст, размещенный между тегами *<NOINDEX>* и *</NOINDEX>*, будет исключен из индексирования. Использование этих тегов никак не отразится на внешнем виде Web-страницы, т.к. они не являются стандартными для языка HTML и будут просто проигнорированы браузером.

Файл robots.txt

При индексировании документов по протоколу HTTP Яндекс.Сервер поддерживает стандарт исключений для роботов. В соответствии с этим стандартом, правила, управляющие поведением поискового робота, должны располагаться в файле /robots.txt, лежащем в корне Web-сервера.

Детальное описание спецификации файла можно прочитать, например, по адресу: http://www.citforum.ru/internet/search/rbtspec.shtml.

В простейшем виде (разрешено все, кроме каталога скриптов) файл robots.txt выглядит следующим образом:

```
User-Agent: *
Disallow: /cgi-bin/
```

Если нужно, чтобы Яндекс.Сервер при индексировании вашего сайта не учитывал общие правила для поисковых роботов, модифицируйте robots.txt, добавив специальное правило для *User-Agent*, заданного при конфигурировании HTTP-запросов.

Например, в следующем примере каталог скриптов закрывается от всех роботов, кроме робота *MyYandexServer*, которому открыто все.

```
User-Agent: *
Disallow: /cgi-bin/

User-Agent: MyYandexServer
Disallow:
```

При написании robots.txt обратите внимание на следующие часто встречающиеся ошибки.

Строка с полем *User-Agent* является обязательной и должна предшествовать строкам с полем *Disallow*. Так, приведенный ниже файл robots.txt не запрещает ничего:

```
Disallow: /cgi-bin
Disallow: /forum
```

Пустые строки в файле robots.txt являются значимыми, они разделяют записи, относящиеся к разным роботам. Например, в следующем фрагменте файла robots.txt строка "*Disallow: /forum*" игнорируется, поскольку перед ней нет строки с полем *User-Agent*.

```
User-Agent: *
Disallow: /cgi-bin

Disallow: /forum
```

Строка с полем *Disallow* может запретить индексирование документов только с одним префиксом. Для запрета нескольких префиксов нужно написать несколько строк.

Например, нижеприведенный файл запрещает индексирование документов, начинающихся с "/cgi-bin /forum", которых, скорее всего, не существует (а не документов с префиксами "/cgi-bin" и "/forum").

```
User-Agent: *
Disallow: /cgi-bin /forum
```

В строках с полем *Disallow* записываются не абсолютные, а относительные префиксы.

То есть файл:

```
User-Agent: *
Disalow: www.myhost.ru/cgi-bin
```

запрещает, например, индексирование документа http://www.myhost.ru/www.myhost.ru/cgi-bin/counter.cgi, но НЕ запрещает индексирование документа http://www.myhost.ru/cgi-bin/counter.cgi.

В строках с полем *Disallow* указываются именно префиксы, а не что-нибудь еще.

Так, файл:

```
User-Agent: *
Disallow: *
```

запрещает индексирование документов, начинающихся с символа * (которых в природе не существует), и сильно отличается от файла:

```
User-Agent: *
Disallow: /
```

который запрещает индексирование всего сайта.

Мета-тег *robots*

При индексировании html-документов Яндекс.Сервер учитывает содержимое мета-тега *robots*, что позволяет запретить роботу индексировать какую-то страницу или следовать по ссылкам, содержащимся на ней.

Значение этого тега может состоять из следующих директив, разделенных запятыми:

Директива	Назначение
index	Страница может быть проиндексирована.
noindex	Страница не должна индексироваться.
follow	Следовать по ссылкам, содержащимся на странице.
nofollow	Не следовать по ссылкам, содержащимся на странице.
all	**index**,**follow** (по умолчанию).
none	**noindex**,**nofollow**.

Пример 1. Не индексировать страницу, но собрать с нее все ссылки на другие страницы:

```
<meta name="robots" content="noindex,follow">
```

Пример 2. Проиндексировать страницу, но не следовать по ссылкам, расположенным на ней:

```
<meta name="robots" content="index,nofollow">
```

Пример 3. Не индексировать страницу и не следовать по ссылкам, расположенным на ней:

```
<meta name="robots" content="noindex,nofollow">
```

Мета-тег *robots* имеет более высокий приоритет, чем настройки индексатора или директивы управления, заданные в файле robots.txt.

Т.е., если например директивы управления в файле robots.txt разрешают индексировать все файлы в каталоге, то блокирующий мета-тег *<meta name="robots" content="noindex,nofollow">* может запретить индексирование страницы, находящейся в этом каталоге.

Примечание:
Нельзя указывать повторяющиеся или конфликтующие директивы, например:

```
<meta name="robots" content="index,noindex,nofollow,follow,follow">
```

Если вы не хотите учитывать мета-тег *robots*, задайте в настройках директиву Options со значением IgnoreMetaRobots.

Индексирование данных через интерфейс ODBC

- Конфигурационный файл индексатора.
- Директивы конфигурационного файла источника odbcds.
- Генерация имени и содержимого документа.
- Пример файла конфигурации для индексирования и поиска по базе данных (ODBC).

Для работы модуля, описываемого в этом разделе, под Unix на компьютере должен быть установлен ODBC-менеджер *unixODBC*. Также необходим ODBC-драйвер для нужной базы данных, например *MyODBC* для *MySQL* или *odbc-postgresql* для *PostgreSQL*.

Конфигурационный файл индексатора

Пример. Определение источника данных odbcds

a)

```
<DataSrc>Name : myname
    Module : libyodbc2.so
    Symbol : ODBC_DATASRC_LIB
    Config : sqldb.cfg
</DataSrc>
```

b)

```
<DataSrc>Name : myname
    Module : libyodbc2.so
    Symbol : ODBC_DATASRC_LIB
    <Odbcds>
        ...
    </Odbcds>
</DataSrc>
```

c)

```
<DataSrc id="odbcds">
    Name : myname
    Config : sqldb.cfg
</DataSrc>
```

d)

```
<DataSrc id="odbcds">
    Name : myname
    <Odbcds>
        ...
    </Odbcds>
</DataSrc>
```

Здесь *myname* — произвольный идентификатор, отличающий один источник данных от другого, а *sqldb.cfg* — произвольное имя конфигурационного файла для источника данных.

Ниже в этом разделе мы рассмотрим директивы этого конфигурационного файла.

Директивы конфигурационного файла источника odbcds

В разделе описаны директивы, относящиеся к процессу индексирования через стандартный источник **odbcds**. Если какая-либо директива отсутствует в конфигурационном файле, для соответствующих параметров будут использованы указанные значения по умолчанию.

Пример конфигурационного файла источника odbcds

```
<Odbcds>
    DataSourceName : test
    UserName : test
    Password : 1234
    DocQuery : SELECT * FROM table
    DocFilter : WHERE id=$1
    UrlQuery : SELECT id FROM table
    DelUrlQuery : SELECT * FROM table WHERE not_indexed=1
    MimeType : text/html
    Charset : utf-8
    Template : /absolute/path/to/doc.template
    TimeStamp : $2
</Odbcds>
```

Директива	Описание
DataSourceName	Задает идентификатор драйвера, который будет использоваться для доступа к данным. Должен совпадать с одним из заданных в локальном файле настроек ODBC ~/.odbc.ini либо в системной файле /etc/odbc.ini (возможно, в вашем случае этот файл должен быть задан переменной среды окружения *ODBCINI*).
Пример: `DataSourceName : PostgreSQL`	
UserName	Идентификатор пользователя с правами на чтение базы данных. Необязательная директива.
Пример: `UserName : mylogin`	
Password	Пароль пользователя. Необязательная директива.
Пример: `Password : mypasssword`	

Генерация имени и содержимого документа

Для всех индексируемых записей базы данных нужно уметь генерировать имена и содержимое документов.

Есть два способа получения имен документов:

- Через директиву UrlQuery.

  В этом случае именем документа будет URL, компонентами которого являются значение директивы Name секции DataSrc конфигурационного файла индексатора и значения полей, указанных в директиве UrlQuery, разделенные символом '/' (слэш).

  В самом начале работы модуля будет выполнен запрос к базе данных для получения списка имен всех индексируемых документов. Затем для получения содержимого документов будут последовательно выполнены запросы, сгенерированные из значений директив DocQuery и DocFilter и соответствующие полученным на предыдущем этапе именам документов.

- Через директиву DocQuery.

  Именем документа в этом случае будет URL, компонентами которого являются значение директивы Name секции DataSrc конфигурационного файла индексатора и значение первого поля из запроса, заданного директивой DocQuery.

В этом случае информация, необходимая для генерации всех документов, будет получена одним запросом к базе данных. Это позволит быстро индексировать базы данных с большим числом записей.

Для получения содержимого документа нужно задать директиву DocQuery и дополняющую ее директиву DocFilter, которая задает условие, позволяющее получить запись базы данных, соответствующую заданному имени документа.

Шаблон генерируемого документа определяется директивой Template.

Директивы, используемые для генерации имени и содержимого документа

Директива	Описание
DocQuery	SQL-запрос для получения информации о записях базы данных, необходимой для генерации документа. Обязательная директива.
Пример: `DocQuery : SELECT id,m_time,title,content FROM mydata` Здесь *id*, *m_time*, *title*, *content* — имена полей таблицы *mydata*.	
DocFilter	Дополнение к запросу, определенному директивой DocQuery, позволяющее получить одну запись БД, соответствующую заданному имени документа. Перед выполнением запроса на получение записи значение этой директивы будет добавлено в конец значения директивы DocQuery. Получившийся запрос должен быть оформлен таким образом, чтобы в качестве результата запроса возвращалась ровно одна запись, содержащая данные для формирования документа. В запросе следует употребить параметры с именами *$1*-*$N*, где *$k* — порядковый номер поля в имени документа. Обязательная директива.
Пример: `DocFilter : WHERE id=$1`	
UrlQuery	Необязательная директива, задающая SQL-запрос, при помощи которого можно получить список всех записей базы данных, подлежащих индексированию. Обычно это оператор *SELECT* по полю или нескольким полям с неповторяющимися значениями (по первичному ключу, если пользоваться терминологией баз данных).

Пример:
 	
`UrlQuery : SELECT id FROM mydata`
 	
Так как результатом этого запроса является совокупность всех записей таблицы по полю *id*, будут проиндексированы все данные, содержащиеся в таблице *mydata*. При этом важно, чтобы содержимое поля *id* было уникально.
 	
`UrlQuery : SELECT id FROM mydata WHERE id<1000`
 	
Будет проиндексирована лишь та часть таблицы, поле *id* которой меньше *1000*.	

DelUrlQuery	Необязательная директива, определяющая SQL-запрос, при помощи которого можно получить список имен документов, подлежащих удалению из индекса. Обычно это оператор *SELECT*, позволяющий сгенерировать такой же URL, который был приписан документу при его индексировании.

Директива	Описание
Пример: `UrlQuery : SELECT id FROM mydata WHERE not_indexed==1` Результатом этого запроса является совокупность всех записей таблицы по полю *id*, для которых значение поля *not_indexed* равно *1*.	
Template	Указывает путь к файлу с шаблоном документа. В шаблоне следует употребить параметры с именами *$1-$N*, где *$k* — порядковый номер поля в директиве DocQuery. В процессе индексирования будут подставлены значения для текущей записи.

Пример:
`DocQuery : SELECT id,m_time,title,content FROM mydata`
`DocFilter : WHERE id=$1`
`Template : doc.template`
Файл doc.template имеет следующий вид:
`<HTML>`
`  <HEAD>`
`  <META NAME="m_time" CONTENT="$2">`
`  <TITLE>$3</TITLE>`
`  </HEAD>`
`  <BODY>`
`  $4`
`  </BODY>`
`</HTML>`
В этом примере *$2* соответствует полю с именем *m_time*, *$3* — полю с именем *title*, *$4* — полю с именем *content*.
Чтобы задать в шаблоне '*$*', напишите его дважды.
Пример:
`<HTML>`
`  <BODY>`
`  $4`
`  $$ - это знак доллара`
`  </BODY>`
`</HTML>`	

TimeStamp	Имя поля или шаблон, задающий номер поля из набора записей, полученного с помощью DocQuery, в котором содержатся данные, указывающие время последнего обновления записи. Необязательная директива.

Примеры:
`TimeStamp : m_time`
`TimeStamp : $2`
Если директива DocQuery имеет значение
`DocQuery : SELECT id,m_time,title,content FROM mydata`
то шаблон *$2* определяет второе поле из запроса, т.е. поле с именем *m_time*.	

MimeType	Указывает формат документа, генерируемого с помощью шаблона Template. Формат документа должен совпадать с одним из форматов, определенных в секциях DocFormat. Необязательная директива. Значение по умолчанию: *text/html*.
Пример:	

Директива	Описание
`MimeType : text/xml`	
Charset	Задает кодировку документа. Может быть указан номер поля из набора записей, полученного с помощью DocQuery. Аргументы директивы описаны ниже. Необязательная директива.

Примеры:
 	
`DocQuery : SELECT id,m_time,title,content,mimetype,charset FROM mydata`
`DocFilter : WHERE id=$1`
`Charset : windows-1251`	

Redirect	Необязательная директива, задающая шаблон для генерации ссылки на найденный документ. Используется при поиске. При отображении ссылки на веб-странице будут подставлены нужные значения полей. В шаблоне можно использовать параметры с именами *$1-$N*, где *$k* — порядковый номер поля в директиве UrlQuery, если она использовалась для получения списка имен, либо *$1*, если для генерации имени документа использовалась директива DocQuery.
Пример: `Redirect : http://myserver.ru/script.asp?id=$1`	

Аргументы директивы Charset

Описание директивы приведено выше.

Аргумент	Описание
recognize	Распознавать кодировку символов автоматически, с помощью анализа текста документа.
<значение кодировки>	Использовать одно из указанных ниже в таблице кодировок значений.

Кодировка символов, используемая в документах

В таблице приведены обозначения, используемые при принудительном задании кодировки.

Кодировка	Обозначение
WinCyrillic	*windows-1251*, *cp1251*
MacCyrillic	*MacCyrillic*, *MacRussian*
DOSCyrillic	*IBM855* или *cp855*
DOSCyrillicRussian	*IBM866*, *cp866*
ISOLatinCyrillic	*ISO-8859-5*, *iso-ir-144*
WinLatin1	*windows-1252*, *cp1252*
WinLatin2	*windows-1250*, *cp1250*
KOI8R	*KOI8-R*, *csKOI8R*
ISO8859_2	*iso-2*, *iso_8859-2*
UTF8	*utf8*, *utf-8*

Пример файла конфигурации для индексирования и поиска по базе данных (ODBC)

В конфигурационном файле индексатора задаем секцию DataSrc следующего вида (см. раздел Модуль источника данных).

```
<DataSrc>Name  : odbc_datasrc
    Module : libydodbc2.so
    Config : odbc_datasrc.cfg
</DataSrc>
```

Здесь *odbc_datasrc* — произвольный идентификатор, определяющий имя источника данных, а *odbc_datasrc.cfg* — имя конфигурационного файла для источника данных.

Содержимое файла odbc_datasrc.cfg может быть таким:

```
DataSourceName : MySQL
BaseName : mydb
UserName : search_user
Password : search_user_pass
DocQuery : SELECT id,m_time,title,content FROM mydata
DocFilter : WHERE id=$1
TimeStamp : m_time
Template : doc.template
MimeType : text/html
```

Файл doc.template шаблона документа имеет следующий вид.

```
  <HTML>
    <HEAD>
      <META NAME="id" CONTENT="$1">
      <META NAME="m_time" CONTENT="$2">
      <TITLE>$3</TITLE>
    </HEAD>
    <BODY>
      $4
    </BODY>
  </HTML>
```

Индексирование баз данных MySQL (Unix)

- Конфигурационный файл индексатора.
- Директивы конфигурационного файла источника mysqlds.
- Генерация имени и содержимого документа.
- Пример файла конфигурации для индексирования и поиска по базе данных (MySQL).

Для работы модуля, описываемого в этом разделе, на компьютере с ОС Unix должна быть установлена разделяемая библиотека *libmysqlclient.so.15*. Последнюю версию *libmysqlclient.so* можно скачать с http://www.mysql.com/data/download.htm.

Конфигурационный файл индексатора

Пример. Определение источника данных mysqlds

a)

```
<DataSrc>Name : myname
    Module : libydmysql2.so
    Symbol : MYSQL_DATASRC_LIB
    Config : sqldb.cfg
</DataSrc>
```

b)

```
<DataSrc>Name : myname
    Module : libydmysql2.so
    Symbol : MYSQL_DATASRC_LIB
    <Mysqlds>
        ...
    </Mysqlds>
</DataSrc>
```

c)

```
<DataSrc id="mysqlds">
    Name : myname
    Config : sqldb.cfg
</DataSrc>
```

d)

```
<DataSrc id="mysqlds">
    Name    myname
    <Mysqlds>
        ...
    </Mysqlds>
</DataSrc>
```

Здесь *mysqlds* — произвольный идентификатор, отличающий один источник данных от другого, а *mysql_datasrc.cfg* — произвольное имя конфигурационного файла для источника данных.

Ниже в этом разделе мы рассмотрим директивы этого конфигурационного файла.

Директивы конфигурационного файла источника mysqlds

В этом разделе описаны директивы, относящиеся к процессу индексирования через стандартный источник **mysqlds**. Если какая-либо директива отсутствует в конфигурационном файле, для соответствующих параметров будут использованы указанные значения по умолчанию.

Пример конфигурационного файла источника mysqlds

```
<Mysqlds>
    HostName : myhost
    BaseName : test
    UserName : test
    Password : 1234
#    UnixSocketPort : 3306
    DocQuery : SELECT * FROM table
    DocFilter : WHERE id=$1
#    UrlQuery : SELECT id FROM table
    DelUrlQuery : SELECT * FROM table WHERE not_indexed=1
    MimeType : text/html
    Charset : utf8
    Template : /absolute/path/to/doc.template
    TimeStamp : $2
</Mysqlds>
```

Директива	Описание	Значение
HostName	Имя хоста или IP адрес SQL-сервера.	
BaseName	Имя SQL-базы.	
UserName	Имя пользователя.	Значение по умолчанию: *текущий логин*.
Password	Пароль для доступа к SQL-серверу. Если **Password** не задан, то будут проверены только те записи в таблице пользователей, которые не имеют пароля.	
UnixSocket	Номер unix-сокета.	Значение по умолчанию: *не задан*.
Port	Номер порта.	Значение по умолчанию: *не задан*.

Генерация имени и содержимого документа

Для всех индексируемых записей базы данных нужно уметь генерировать имена и содержимое документов.

Есть два способа получения имен документов:

- Через директиву UrlQuery.

  В этом случае именем документа будет URL, компонентами которого являются значение директивы Name секции DataSrc конфигурационного файла индексатора и значения полей, указанных в директиве UrlQuery, разделенные символом '/' (слэш).

  В самом начале работы модуля будет выполнен запрос к базе данных для получения списка имен всех индексируемых документов. Затем для получения содержимого документов будут последовательно выполнены запросы, сгенерированные из значений директив DocQuery и DocFilter и соответствующие полученным на предыдущем этапе именам документов.

- Через директиву DocQuery.

  Именем документа в этом случае будет URL, компонентами которого являются значение директивы Name секции DataSrc конфигурационного файла индексатора и значение первого поля из запроса, заданного директивой DocQuery.

  Информация, необходимая для генерации всех документов, будет получена одним запросом к базе данных в начале работы модуля, а полученный набор данных будет закеширован на стороне MySQL-сервера. Это позволит быстро индексировать базы данных с большим числом записей.

Для получения содержимого документа нужно задать директиву DocQuery и дополняющую ее директиву DocFilter, которая задает условие, позволяющее получить запись базы данных, соответствующую заданному имени документа.

Шаблон генерируемого документа определяется директивой Template.

Директивы, используемые для генерации имени и содержимого документа

Директива	Описание
DocQuery	SQL-запрос для получения всей необходимой для генерации документов информации о записях базы данных. Условие *WHERE*, позволяющее получить данные из базы, соответствующие конкретному документу, задается директивой DocFilter.
Пример: `DocQuery : SELECT id,m_time,title,content,mimetype FROM programs` здесь *id*, *m_time*, *title*, *content*, *mimetype* — имена полей таблицы programs.	
DocFilter	Дополнение к SQL-запросу, определенному директивой DocQuery, позволяющее получить одну запись базы данных, соответствующую заданному документу. Перед выполнением SQL-запроса на получение записи, значение этой директивы будет добавлено в конец значения директивы DocQuery. Получившийся запрос должен быть оформлен таким образом, чтобы в качестве результата возвращалась ровно одна запись, содержащая данные для формирования документа. В запросе следует употребить параметры с именами *$1*-*$N*, где *$k* — порядковый номер поля в директиве UrlQuery, если она использовалась для получения списка имен, либо *$1*, если для генерации имени документа использовалась директива DocQuery.

Директива	Описание

Примеры:
`UrlQuery : SELECT id FROM mydata`
`DocQuery : SELECT m_time,title,content FROM mydata`
`DocFilter : WHERE id=$1`
В этом примере для генерации документа с именем *'/mysql_datasrc/99' ('mysql_datasrc'* — значение директивы Name секции DataSrc конфигурационного файла индексатора) будет выполнен запрос к базе данных: *"SELECT m_time,title,content FROM mydata WHERE id=99"*
`DocQuery : SELECT id,m_time,title,content FROM mydata`
`DocFilter : WHERE id=$1`
В этом примере для документа *'/mysql_datasrc/99'* будет выполнен запрос *"SELECT id,m_time,title,content FROM mydata WHERE id=99"*.	

UrlQuery	Необязательная директива, определяющая SQL-запрос, при помощи которого можно получить список всех записей базы данных, подлежащих индексированию. Обычно это оператор *SELECT* по полю или нескольким полям с неповторяющимися значениями (по первичному ключу, если пользоваться терминологией баз данных).

Примеры.
`UrlQuery : SELECT id FROM mydata`
- так как результатом этого запроса является совокупность всех записей таблицы по полю id, то проиндексированы будут все данные, содержащиеся в *mydata*. При этом важно, чтобы содержимое поля *id* было уникально, т.е. зная конкретное значение *id*, мы должны быть уверены, что ему соответствует одна и только одна запись в данной таблице.
`UrlQuery : SELECT id,name FROM mydata`
- то же, что и в первом случае, однако, значения в поле *id* или в поле *name* могут повторяться. Но любые их комбинации должны быть уникальны.
`UrlQuery : SELECT id,name FROM mydata WHERE id<1000`
- проиндексирована будет лишь та часть таблицы, поле *id* которой меньше *1000*.	

DelUrlQuery	Необязательная директива, определяющая SQL-запрос, при помощи которого можно получить список имен документов, подлежащих удалению из индекса. Обычно это оператор *SELECT*, позволяющий сгенерировать такой же URL, который был приписан документу при его индексировании.
Пример. `UrlQuery : SELECT id FROM mydata WHERE not_indexed==1` Результатом этого запроса является совокупность всех записей таблицы по полю *id*, для которых значение поля *not_indexed* равно 1.	
Template	Указывает путь к файлу с шаблоном документа. В шаблоне следует употребить параметры с именами *$1-$N*, где *$k* — порядковый номер поля в директиве DocQuery. В процессе индексирования будут подставлены значения для текущей записи.

Пример:
`DocQuery : SELECT id,m_time,title,content,mimetype FROM mydata`
`DocFilter : WHERE id=$1`
`Template : doc_template.htm`
Файл doc_template.htm имеет следующий вид:	

Директива	Описание

```
<HTML>
  <HEAD>
    <META NAME="id" CONTENT="$1">
    <META NAME="m_time" CONTENT="$2">
    <TITLE>$3</TITLE>
  </HEAD>
  <BODY>
    $4
  </BODY>
</HTML>
```

TimeStamp	Задает либо имя, либо номер поля из набора записей, полученного с помощью DocQuery, в котором содержатся данные, указывающие время последнего обновления записи. Необязательная директива.

Примеры:

```
DocQuery : SELECT id,m_time,title,content,mimetype FROM mydata
DocFilter : WHERE id=$1

# Используем в качестве времени последнего обновления записи
# значение поля с именем m_time.
TimeStamp : m_time

# То же самое, но задается не имя поля, а его номер в директиве DocQuery
TimeStamp : $2
```

MimeType	Указывает формат документа, генерируемого с помощью шаблона Template. Формат документа должен совпадать с одним из форматов, определенных в секциях DocFormat. Необязательная директива. Значение по умолчанию: *text/html*.

Пример:

```
MimeType : text/xml
```

Charset	Задает кодировку документа. Может быть указан номер поля из набора записей, полученного с помощью DocQuery. Аргументы директивы описаны ниже. Необязательная директива.

Примеры:

```
DocQuery : SELECT id,m_time,title,content,mimetype,charset FROM mydata
DocFilter : WHERE id=$1

Charset : windows-1251
# или так:
Charset : $6
```

Redirect	Необязательная директива, задающая шаблон для генерации ссылки на найденный документ. Используется при поиске. При отображении ссылки на веб-странице будут подставлены нужные значения полей. В шаблоне можно использовать параметры с именами *$1-$N*, где *$k* — порядковый номер поля в директиве UrlQuery, если она использовалась для получения списка имен, либо *$1*, если для генерации имени документа использовалась директива DocQuery.

Директива	Описание

Примеры:
`Redirect : http://www.myhost.ru/cgi-bin/message?`
`Redirect : http://www.myhost.ru/cgi-bin/message?id=$1`	

Аргументы директивы Charset

Описание директивы приведено выше.

Аргумент	Описание
recognize	Распознавать кодировку символов автоматически, с помощью анализа текста документа.
<значение кодировки>	Использовать одно из указанных ниже в таблице кодировок значений.

Кодировка символов, используемая в документах

В таблице приведены обозначения, используемые при принудительном задании кодировки.

Кодировка	Обозначение
WinCyrillic	*windows-1251*, *cp1251*
MacCyrillic	*MacCyrillic*, *MacRussian*
DOSCyrillic	*IBM855* или *cp855*
DOSCyrillicRussian	*IBM866*, *cp866*
ISOLatinCyrillic	*ISO-8859-5*, *iso-ir-144*
WinLatin1	*windows-1252*, *cp1252*
WinLatin2	*windows-1250*, *cp1250*
KOI8R	*KOI8-R*, *csKOI8R*
ISO8859_2	*iso-2*, *iso_8859-2*
UTF8	*utf8*, *utf-8*

Пример файла конфигурации для индексирования и поиска по базе данных (MySQL)

В конфигурационном файле индексатора задаем секцию DataSrc следующего вида (см. раздел Модуль источника данных).

```
<DataSrc>Name : mysql_datasrc
    Module : libydmysql2.so
    Config : mysql_datasrc.cfg
</DataSrc>
```

Здесь *mysql_datasrc* — произвольный идентификатор, определяющий имя источника данных, а *mysql_datasrc.cfg* — имя конфигурационного файла для источника данных.

Содержимое файла mysql_datasrc.cfg может быть таким:

```
HostName : localhost
BaseName : mydb
UserName : search_user
Password : search_user_pass
DocQuery : SELECT id,m_time,title,content FROM mydata
DocFilter : WHERE id=$1
Template : doc.template
TimeStamp : m_time
MimeType : text/html
Charset : utf-8
```

Файл doc.template шаблона документа имеет следующий вид:

```
<HTML>
  <HEAD>
    <META NAME="id" CONTENT="$1">
    <META NAME="m_time" CONTENT="$2">
    <TITLE>$3</TITLE>
  </HEAD>
  <BODY>
    $4
  </BODY>
</HTML>
```

В данном примере для каждой записи MySQL-базы генерируется HTML-документ.

Парсеры (анализаторы содержимого документа)

Парсер представляет собой встроенный модуль, анализирующий содержимое индексируемых документов.

Основная задача парсера — выделить из документа нужный для индексирования текст. Текст, выделяемый парсером, может быть помечен как принадлежащий определенной зоне документа, или как имеющий определенные свойства (атрибуты). На основании элементов форматирования документа парсер может указать границы предложений и абзацев, а также вес данного отрывка текста.

В состав Яндекс.Сервера входят HTML- и XML-парсеры, каждый из которых предназначен для анализа документов соответствующего типа. Кроме того, данные парсеры распознают документы наиболее распространенных типов: **txt**, **pdf**, **rtf**, **doc**, **xls**, **ppt**, **odt**, **swf**, **mpeg** и т.д. Полный список поддерживаемых медиа-типов приведен в таблице Медиа-типы документов.

Яндекс.Сервер поставляется с уже настроенными парсерами. При необходимости, любой из парсеров парсер может быть настроен дополнительно.

Для дополнительной настройки парсера необходимо:

1. Определить коллекцию документов, для которой требуется изменить стандартные настройки парсера.
2. Определить тип парсера (HTML или XML), настройки которого требуется изменить.
3. В соответствующей секции Collection задать подсекцию DocFormat, в которой определить тип конфигурируемого парсера и путь к его конфигурационному файлу.
4. Разработать конфигурационный файл парсера.

Подробная информация о настройках парсеров размещена в разделах Конфигурация HTML-парсера и Конфигурация XML-парсера.

Медиа-типы документов

Тип/подтип (MimeType)	Возможность настройки
text/html	есть
text/plain	-
audio/mpeg	-
text/xml	есть
application/pdf	-
text/rtf	-
application/msword	-
application/x-shockwave-flash	-
application/vnd.ms-excel	-
application/vnd.ms-powerpoint	-
application/xhtml+xml	-

Тип/подтип (MimeType)	Возможность настройки
application/vnd.openxmlformats-officedocument.wordprocessingml.document	-
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	-
application/vnd.openxmlformats-officedocument.presentationml.presentation	-
application/vnd.oasis.opendocument.text	-
application/vnd.oasis.opendocument.presentation	-
application/vnd.oasis.opendocument.spreadsheet	-
application/vnd.oasis.opendocument.graphics	-

Конфигурация HTML-парсера

- Конфигурационный файл HTML-парсера.
- Проектирование конфигурации HTML-парсера.
- Синтаксис конфигурационного файла.
- Конфигурация по умолчанию.

Конфигурационный файл HTML-парсера

Пример

```
<HtmlParser>
    <Zones>
        zonename : zone1,zone2, ...
        ...
    </Zones>
    <Attributes>
        attrname : attr1,attr2, ...
        ...
    </Attributes>
</HtmlParser>
```

Проектирование конфигурации HTML-парсера

Конфигурация HTML-парсера проектируется в соответствии с типичными поисковыми задачами, которые могут возникнуть для данной коллекции документов. В процессе разработки конфигурации рекомендуется придерживаться следующих основных шагов.

1. Определить имена поисковых зон и поисковых атрибутов, которые будут участвовать в языке запросов.
   Пример: Для поиска по подзаголовкам документов введем поисковую зону *header*. Для поиска по тексту ссылок на другие документы введем поисковую зону *anchor*. Для поиска по адресам ссылок на другие документы введем поисковый атрибут *link*.
2. Для каждой поисковой зоны указать список имен HTML-тегов, содержимое которых должно принадлежать данной поисковой зоне.
   Пример: Определим, что поисковой зоне *header* принадлежит текст документа, находящийся внутри любого из тегов *<h1>...</h1>*, *<h2>...</h2>* или *<h3>...</h3>*.
3. Определить, будут ли некоторые поисковые зоны условными. *Условными* будут называться поисковые зоны, образуемые только при наличии заданного поискового атрибута. Для списка HTML-тегов, определяющего поисковую зону, может быть дополнительно указано имя некоторого поискового (не HTML!) атрибута. Если имя указано, и при попытке создать поисковую зону из содержимого данного HTML-тега эта зона не получает поискового атрибута с указанным именем (и любым значением), то поисковая зона не создается.

Пример: Допустим поисковой зоне *anchor* должен принадлежать текст документа, находящийся внутри тега *<a>...*, но только при условии, что данный текст ссылается на другой документ. Для этого данный тег должен иметь HTML-атрибут *href*. Поэтому определим поисковую зону *anchor* как условную, возникающую только при наличии поискового атрибута *link*, описанного в следующем примере.

4. Для каждого поискового атрибута выбрать его тип (из числа описанных в разделе Типы поисковых атрибутов) и список пар (имя HTML-тега, имя HTML-атрибута этого тега). Если у HTML-тега, имя которого совпадает с первым именем в паре имеется HTML-атрибут, имя которого совпадает со вторым именем в паре, то значение этого HTML-тега распознается как значение определяемого поискового атрибута по правилам, специфичным для указанного типа поискового атрибута.
   Пример: Определим, что поисковый атрибут *link* имеет тип *URL* и значениями этого атрибута будут значения HTML-атрибута *href* HTML-тега *<a>* и значения HTML-атрибута *src* HTML-тега *<frame>*.

Конфигурация парсера, спроектированная в примерах данного раздела, имеет следующий вид.

```
<HtmlParser>
    <Zones>
        header : h1,h2,h3
        anchor : a/link
    </Zones>
    <Attributes>
        link : URL,any/a.href,frame.src
    </Attributes>
</HtmlParser>
```

Формальные правила описания конфигурации приведены в следующем разделе.

Синтаксис конфигурационного файла

Описание дополнительных настроек HTML-парсера размещается в отдельном файле. При этом в секции DocFormat конфигурационного файла индексатора нужно задать путь к файлу конфигурации парсера (путь к файлу указывается через директиву Config). Вся конфигурация парсера должна быть размещена внутри секции **<HtmlParser>**. Директивы, определяющие поисковые зоны, располагаются внутри подсекции **<Zones>**, а директивы, определяющие поисковые атрибуты — внутри подсекции **<Attributes>**.

Далее приводятся формальные правила написания конфигураций, и даются конкретные примеры с подробными комментариями.

Конфигурирование поисковых зон

Пример. Формальные правила конфигурации поисковых зон (HTML)

Формальные правила описания зон можно представить следующим набором выражений:

```
<Zones>
    yxzone : htelem(,htelem)*
    yxzone : htelem(,htelem)*/yxattr
</Zones>
```

где:

- *yxzone* — имя поисковой зоны;
- *htelem* — имя HTML-тега;
- *yxattr* — имя поискового атрибута, определяющего условную поисковую зону;
- *(...)** — ноль, один или несколько элементов.

Имя поисковой зоны не может совпадать с одним из зарезервированных имен "doc", "empty", "any".

Вместо имени HTML-тега допустимо использовать символ "_" (подчеркивание). Он означает любой тег.

Пример: Текст внутри тега “title” принадлежит поисковой зоне “title”.

```
title : title
```

Пример: Текст внутри всех элементов “Hn”, а также заголовки таблиц принадлежат поисковой зоне “header” .

```
header : h1,h2,h3,h4,h5,h6,caption
```

Пример: Текст внутри тега “a” принадлежит поисковой зоне “anchor” только при условии, что имеется поисковый атрибут “link”.

```
anchor : a/link
```

Конфигурирование поисковых атрибутов

Пример. Формальные правила конфигурации поисковых атрибутов (HTML)

Формальные правила описания поисковых атрибутов можно представить следующим набором выражений:

```
<Attributes>
    yxattr : TYPE/htelem.htattr(,htelem.htattr)*
    yxattr : TYPE,yxzone/htelem.htattr(,htelem.htattr)*
    yxattr : TYPE,yxzone,function/htelem.htattr(,htelem.htattr)*
    yxattr : TYPE,yxzone,function,ignore/htelem.htattr(,htelem.htattr)*
    # только для атрибутов типа URL
    yxattr : TYPE,yxzone,function,ignore,ext( ext)*/htelem.htattr(,htelem.htattr)*
    yxattr : TYPE,yxzone,function,ignore,ext( ext)*,local/htelem.htattr(,htelem.htattr)*
</Attributes>
```

где:

- *yxzone* — имя поисковой зоны;
- *yxattr* — имя поискового атрибута;
- *htelem* — имя HTML-тега;
- *htattr* — имя HTML-атрибута;
- *(...)** — ноль, один или несколько элементов;
- *TYPE* — тип поискового атрибута, как описано в разделе Типы поисковых атрибутов;
- *function* — одно из строковых значений, указанных ниже, определяющих правила распознавания текста атрибута;
- *ignore* — флажок, указывающий, что документный атрибут надо распознавать, но не надо запоминать его в индексных файлах. Используется для создания группировочных атрибутов;
- *ext* — список расширений имен файлов разделенных пробелами;
- *local* — флажок для пометки только локальных (внутрисайтовых) ссылок.

Символ "_" (подчеркивание) вместо имени HTML-тега или HTML-атрибута обозначает любой элемент или атрибут. Если символ "_" задан вместо имени поискового атрибута, имя поискового атрибута будет совпадать с именем HTML-атрибута.

Пример: Многие HTML-теги могут иметь всплывающую подсказку, заданную через HTML-атрибут *title*. Чтобы проиндексировать все эти атрибуты, можно определить поисковый атрибут *tooltip* следующим образом:

```
tooltip : TEXT/_.title
```

Каждое из слов текста атрибута *title* любого HTML-тега войдет в индекс с учетом морфологии.

Особый случай представляют собой теги <*META*> и <*LINK*>. Для удобства их использования принято, что именем атрибута тега <*META*> является значение атрибутов *NAME* или *HTTP-EQUIV*, а значением — содержимое атрибута *CONTENT*. Для тега <*LINK*> именем атрибута считается значение атрибутов *REL/REV*, а значением — содержимое атрибута *HREF*.

Пример: Каждый документ в электронной библиотеке содержит meta-тег следующего вида:

```
<meta name="author" content="_имя_автора_">
```

Чтобы обеспечить поиск документов, принадлежащих конкретному автору, определим поисковый атрибут *author*:

```
author : TEXT/meta.author
```

Пример:

```
email : URL/link.made
_ : TEXT/meta._
_ : URL/link._
```

Это означает, что будут проиндексированы все встретившиеся в документах *META* и *LINK* теги. Причем, если в *LINK* значение одного из его атрибутов окажется равно *made*, то поисковый атрибут будет иметь имя *email*. Во всех остальных случаях (это касается и *META*) будут образовываться поисковые атрибуты с именами, равными соответствующим значениям атрибутов данных тегов. Это позволяет решить проблему постоянного добавления в конфигурацию записей при каждом появлении нового, еще не описанного *META* или *LINK* тега. В настройках подобного вида приоритет имеют явно заданные имена атрибутов.

Если название поисковой зоны после типа атрибута опущено, поисковый атрибут будет документным атрибутом. Если имя поисковой зоны указано, возможны следующие случаи:

Значение	Описание
doc	Документный атрибут, тот же самый результат получается, если имя зоны не указывать.
empty	Атрибут данной точки документа, то есть атрибут специальной зоны нулевой длины, как обсуждалось в Поисковые зоны и атрибуты.
any	Если для содержимого данного HTML-элемента сформирована поисковая зона, поисковый атрибут является атрибутом этой зоны. В противном случае это атрибут данной точки документа.
другое имя	Если из содержимого HTML-элемента не сформирована поисковая зона с данным именем, поисковый атрибут не создается.

Пример: Для поиска картинок по именам файлов определим поисковый атрибут *image*:

```
image : URL,empty/img.src
```

Аргумент *function* может принимать следующие значения:

Значение	Описание
parse_http_expires	Распознавать текст атрибута по правилам, применяемым для *meta.expires*.
parse_http_refresh	Распознавать текст атрибута по правилам, применяемым для *meta.refresh*.
parse_http_charset	Распознавать текст атрибута по правилам, применяемым для *meta.content-type*.
parse_meta_robots	Распознавать текст атрибута по правилам, применяемым для *meta.robots*.
parse_data_integer	Распознавать текст атрибута как целое число.

Аргумент *function* может быть пропущен. В этом случае применяются правила распознавания атрибута по умолчанию, в соответствии с его типом.

Имена поисковых атрибутов типа *URL* могут определяться не только наличием или отсутствием соответствующих тегов и их атрибутов, но и расширениями имен файлов, составляющих значение HTML-атрибута, а также фактом, является ли ссылка локальной для данного сайта или уходит на внешние ресурсы.

Пример: Создадим дополнительные поисковые атрибуты для случаев, когда HTML-атрибут представляет собой внутрисайтовую ссылку или ссылается на музыкальный файл:

```
# По умолчанию - расширения не даны
link : URL,anchor/a.href
link : URL,any/frame.src,iframe.src,area.href

# В случае музыкальных расширений или внутренних ссылок меняем имя атрибута
linkmp3 : URL,anchor,,,mp3 mpga mp2 ra/a.href
linkint : URL,anchor,,,,local/a.href
linkint : URL,any,,,,local/frame.src,iframe.src,area.href
```

Если в конфигурации зон сформирована условная зона *anchor* по правилу

```
anchor : a/link
```

то текст внутрисайтовых и музыкальных ссылок в эту зону не попадет.

Конфигурация по умолчанию

Ниже приведен пример конфигурационного файла для HTML-парсера. Данная настройка соответствует поведению парсера по умолчанию, то есть будет использоваться в случае, если дополнительная конфигурация парсера не указана.

Пример. Конфигурация HTML-парсера по умолчанию

```
<HtmlParser>
    <Zones>
        title : title
        address : address
        anchor : a/link
    </Zones>
    <Attributes>
        _ : LITERAL/meta._
        link : URL,anchor/a.href
        link : URL,any/frame.src,iframe.src,area.href
        link : URL/link._
        robots : LITERAL,doc,parse_meta_robots,ignore/meta.robots
        refresh : URL,doc,parse_http_refresh,ignore/meta.refresh
        style : URL/link.stylesheet
        profile : URL/head.profile
        script : URL,any/script.src
        image : URL,any/img.src
        applet : URL,any/applet.code,applet.object
        object : URL,any/object.data,object.classid
        abstract : TEXT/meta.description
        keywords : TEXT/meta.keywords
        hint : TEXT,any/img.alt,area.alt
        tooltip : TEXT,any/_.title
    </Attributes>
</HtmlParser>
```

Конфигурация XML-парсера

- Конфигурационный файл XML-парсера.
- Проектирование конфигурации XML-парсера.
- Синтаксис конфигурационного файла.
- Конфигурация по умолчанию.
- Пример конфигурации XML-парсера.

Конфигурационный файл XML-парсера

Пример

```
<XmlParser>
    <DOCTYPE>
        LocalDTD : ...
        <Zones>
            zonename : zone1,zone2, ...
            ...
        </Zones>
        <Attributes>
            attrname : attr1,attr2, ...
            ...
        </Attributes>
        <TextFlags>
            ...
        </TextFlags>
    </DOCTYPE>
</XmlParser>
```

Проектирование конфигурации XML-парсера

В процессе разработки конфигурации XML-парсера рекомендуется придерживаться тех же основных шагов, что подробно описаны в разделе Проектирование конфигурации HTML-парсера:

1. Определить имена поисковых зон и поисковых атрибутов, которые будут участвовать в языке запросов.
2. Для каждой поисковой зоны указать список имен XML-элементов, содержимое которых должно принадлежать данной поисковой зоне. Определить, будут ли некоторые поисковые зоны условными.
3. Для каждого поискового атрибута выбрать его тип и список пар (имя XML-элемента, имя XML-атрибута этого элемента), определяющих атрибут.
4. Дополнительно для каждого XML-элемента можно определить способ обработки текста — границы слов и абзацев, способ обработки пробелов и вес слов.

Синтаксис конфигурационного файла

Описание дополнительных настроек XML-парсера размещается в отдельном файле. При этом в секции DocFormat конфигурационного файла индексатора нужно задать путь к файлу конфигурации парсера (путь к файлу указывается через директиву Config). Вся конфигурация парсера должна быть размещена внутри секции **<XmlParser>**. Эта секция включает одну или несколько подсекций **<DOCTYPE>**, каждая из которых определяет конфигурацию, относящуюся к заданному типу XML-документов и включает, в свою очередь, подсекции **<Zones>**, **<Attributes>** и **<TextFlags>**.

Определение типов XML-документов

Правила интерпретации каждого типа XML-документов описываются в отдельной секции **<DOCTYPE>**. Каждая такая секция может иметь атрибуты *public*, *system* и *root*. Анализ значений этих атрибутов позволяет установить соответствие между данным XML-документом и нужными настройками парсера.

Сначала анализируется атрибут *public*, который, в случае своего наличия, содержит подстроку, содержащуюся в значении одноименного атрибута элемента **<DOCTYPE>** XML-документа. Если соответствие не найдено, аналогичный анализ проводится для атрибутов *system*. Если соответствие опять не найдено (это может случиться, в частности, если элемент **<DOCTYPE>** отсутствует в XML-документе), сравнивается значение атрибута *root* секции **<DOCTYPE>** конфигурационного файла и имени корневого элемента XML-документа.

Если ни одна из секций **<DOCTYPE>** конфигурационного файла, имеющая атрибуты, не соответствует XML-документу, будет использована конфигурация, описанная в секции без атрибутов.

Если секция **<DOCTYPE>** без атрибутов отсутствует, будет использована конфигурация, описанная в разделе Конфигурация по умолчанию.

Директивы секции <DOCTYPE>

LocalDTD — необязательная директива. Определяет локальный файл DTD, который будет использоваться парсером вместо внешнего, в случае, если он указан в элементе **<DOCTYPE>** XML-документа.

Конфигурирование поисковых зон

Пример. Формальные правила конфигурации поисковых зон (XML)

Формальные правила описания зон можно представить следующим набором выражений:

```
<Zones>
    yzone : xelem(,xelem)*
    yzone : xelem(,xelem)*/yattr
</Zones>
```

где:

- *yzone* — имя поисковой зоны;
- *xelem* — имя XML-элемента;
- *yattr* — имя поискового атрибута, определяющего условную поисковую зону;
- *(...)** — ноль, один или несколько элементов.

Имя поисковой зоны не может совпадать с одним из зарезервированных имен "doc", "empty", "any".

Вместо имени XML-элемента допустимо использовать символ "_" (подчеркивание), который означает любой XML-элемент. Если символ "_" употреблен вместо имени поисковой зоны, это значит, что имя поисковой зоны совпадает с именем XML-элемента.

Конфигурирование поисковых атрибутов

Пример. Формальные правила конфигурации поисковых атрибутов (XML)

Формальные правила описания поисковых атрибутов можно представить следующим набором выражений:

```
<Attributes>
    yattr : TYPE/xelem.xattr(,xelem.xattr)*
    yattr : TYPE,yzone/xelem.xattr(,xelem.xattr)*
    yattr : TYPE,yzone,function/xelem.xattr(,xelem.xattr)*
    yattr : TYPE,yzone,function,ignore/xelem.xattr(,xelem.xattr)*
</Attributes>
```

где:

- *yzone* — имя поисковой зоны, как описано в разделе Конфигурация HTML-парсера;
- *yattr* — имя поискового атрибута;
- *xelem* — имя XML-элемента;
- *xattr* — имя XML-атрибута;
- *(...)** — ноль, один или несколько элементов;
- *TYPE* — тип поискового атрибута, как описано в разделе Типы поисковых атрибутов;
- *function* — одно из строковых значений, описанных в разделе Конфигурация HTML-парсера и определяющих правила распознавания текста атрибута;
- *ignore* — флажок, указывающий, что документный атрибут надо распознавать, но не надо запоминать его в индексных файлах. Используется для создания группировочных атрибутов.

Символ "_" (подчеркивание) вместо имени XML-элемента или XML-атрибута обозначает любой элемент или атрибут. Если символ "_" задан вместо имени поискового атрибута, имя поискового атрибута будет совпадать с именем XML-атрибута.

Конфигурирование правил обработки текста

Пример. Формальные правила конфигурации обработки текста (XML)

Формальные правила обработки текста можно представить следующим набором выражений:

```
<TextFlags>
    ybreak : (xelem)*(, xelem.xattr)*(, xelem.xattr.xval)*
</TextFlags>
```

где:

- *ybreak* — один из флагов обработки текста, перечисленных ниже;
- *xelem* — имя XML-элемента;
- *xattr* — имя XML-атрибута;
- *xval* — значение XML-атрибута;
- *(...)** — ноль, один или несколько элементов.

Символ "_" (подчеркивание) вместо имени XML-элемента обозначает любой элемент.

Флажки обработки текста

Флаги	Описание	Значение
BREAK_NONE, *BREAK_WORD*, *BREAK_SENTENCE*, *BREAK_PARAGRAPH*	Определяет, будет ли текст внутри XML-элемента отделен границами слова, предложения или абзаца в дополнение к обычным пунктуационным правилам.	Значение по умолчанию: *BREAK_NONE*.
SPACE_DEFAULT, *SPACE_PRESERVE*	Определяет, значимы ли пробельные символы в тексте внутри XML-элемента.	Значение по умолчанию: *SPACE_DEFAULT*.
WEIGHT_ZERO, *WEIGHT_LOW*, *WEIGHT_NORMAL*, *WEIGHT_HIGH*, *WEIGHT_BEST*	Определяет относительный вес слов в тексте внутри XML-элемента. В случае значения *WEIGHT_ZERO* текст проиндексирован не будет.	Значение по умолчанию: *WEIGHT_NORMAL*.

Примечание:
Чтобы у найденного документа было определено свойство "заголовок документа", необходимо, чтобы в настройках парсера была определена зона *title* с флагом обработки текста *BREAK_PARAGRAPH* и документ содержал не менее одного предложения в этой зоне.

Конфигурация по умолчанию

Ниже приведен пример конфигурационного файла для XML-парсера. Данная настройка соответствует поведению по умолчанию — она будет использоваться в случае, если дополнительная конфигурация парсера не указана.

Пример

```
<XmlParser>
    <DOCTYPE>
        <Zones>
           # все XML-элементы образуют поисковые зоны с таким же именем
           _ : _
        </Zones>
        <Attributes>
           # для всех зон все XML-атрибуты соответствующих элементов образуют поисковые
зонные атрибуты
           # с таким же именем, как имя XML-атрибута и типом TEXT
           _ : TEXT,any/_._
        </Attributes>
        <TextFlags>
           # все XML-элементы независимо от XML-атрибутов и их значений разбивают текст
на абзацы
           BREAK_PARAGRAPH : _._
        </TextFlags>
    </DOCTYPE>
</XmlParser>
```

Пример конфигурации XML-парсера

Индексирование WAP-ресурсов

Ниже приведен пример секции **<DOCTYPE>**, которая может быть использована при индексировании WML-документов (см. http://www.wapforum.org/DTD/wml12.dtd).

Пример. Конфигурация WML-парсера

```
<DOCTYPE public="DTD WML 1.2">
    LocalDTD : wml12.dtd
    <Zones>
        anchor : anchor,a,go,card,option/link
    </Zones>
    <Attributes>
        link : URL,anchor/
_.href,card.ontimer,card.onenterforward,card.onenterbackward,option.onpick
        title : TEXT,doc/card.title
    </Attributes>
    <TextFlags>
        BREAK_PARAGRAPH : br,table,tr,td
    </TextFlags>
</DOCTYPE>
```

Настройка и использование поискового сервера

Настройка поискового модуля

- Конфигурационный файл поискового модуля.
- Директивы конфигурационного файла поискового модуля.

Конфигурационный файл поискового модуля

Пример

```
<Collection>IndexDir : workindex
    RequestThreads:
    RequestQueueSize:
    MorphFixList: ...
    SearchOver: ...
    DownloadMissingPassages: ...
    UserRelevanceLibrary: ...
    ...
    <QueryCache>Dir : cache
        LifeTime : ...
    </QueryCache>
    ...
    <SearchPageTemplate>Method : ...
        Module : ...
        Options : ...
    </SearchPageTemplate>
</Collection>
```

Директивы конфигурационного файла поискового модуля

Конфигурация поискового сервера состоит из директив и секций: QueryCache, SearchPageTemplate. Ни одна из секций не является обязательной.

Директива	Описание	Значение
IndexDir	Имя каталога с индексными файлами, подготовленными индексатором и используемыми для поиска.	Значение по умолчанию: *./workindex*.
RequestThreads	Определяет максимальное количество одновременно выполняемых поисковых запросов. Если уже выполняется определяемое данной директивой количество запросов, выполнение вновь поступивших запросов откладывается до тех пор, пока не будут выполнены текущие запросы.	Значение по умолчанию: *5*.
RequestQueueSize	Определяет максимальный размер очереди поисковых запросов, ожидающих начала выполнения. В случае нулевого значения директивы максимальный размер очереди запросов бесконечен.	Значение по умолчанию: *0*.

Директива	Описание	Значение
	Если начала выполнения уже ожидает определяемое данной директивой количество запросов, на вновь поступившие запросы сервер отвечает "HTTP/ 1.0 503 Service Unavailable" и не выполняет их. При максимальной загрузке системы уменьшение значения данной директивы приводит, с одной стороны, к сокращению времени выполнения запроса, за счет уменьшения времени ожидания в очереди, но с другой стороны, ведет к росту числа отказов.	
MorphFixList	Имя файла, содержащего правильное морфообразование отсутсвующих в словаре слов. Если этот файл задан, то слова, указанные в нем (и их формы) будут участвовать в поиске как словоформы друг друга.	Значение по умолчанию: *не задан*.
UserRelevanceLibrary	Указывает путь к библиотеке, в которой определена функция, позволящая выполнять произвольное ранжирование результатов поиска. Ранжирование осуществляется посредством вычисления оценки документа на основании группировочных атрибутов и текстовой релевантности. В поставку Я.Сервера включены примеры реализации функции в файле: .\data\sources_sample\include \userrelevancesample.cpp и скомпилированная библиотека: .\data \sources_sample \userrelevancesample.so для Linux или .\data\sources_sample \userrelevancesample.dll для Windows.	
DownloadMissingPassages	Указывает, следует ли для показа фрагментов во время поиска загружать документы, сохраненные тексты которых отсутствуют в архиве.	Значение по умолчанию: *по*.
SearchOver	Продолжение заголовка "Поиск по ..." страницы с результатами поиска во встроенном дизайне.	Значение по умолчанию: *серверу*.

Директивы секции SearchPageTemplate

Секция **SearchPageTemplate** предназначена для задания шаблона, в соответствии с которым сервер будет показывать страницу с результатами поиска. В случае отсутствия данной секции будет использован встроенный шаблон.

В секцию входят следующие директивы:

Директива	Описание	Значение
Method	Указывает язык программирования, на котором написан шаблон страницы. Аргументы директивы описаны ниже. Обязательная директива.	
Module	Указывает путь к файлу, содержащему шаблон страницы. Если указанный файл шаблона отсутствует или имеет неправильный формат, будет использован встроенный шаблон. Обязательная директива.	
Options	Используется для шаблонов на Perl и C++. Для интерпретатора Perl позволяет задать дополнительные параметры командной строки, например, подключить какие-либо специфические библиотеки. Для динамической библиотеки на C++ определяет строку инициализации, которую можно получить, вызвав *IInitContext::GetCommandLine()* в функции *UserInit*.	Значение по умолчанию: *не задан*

Аргументы директивы Method

Описание директивы приведено выше.

Аргумент	Описание
perl	Шаблон страницы написан на языке Perl. Для использования этого аргумента на компьютере должен быть установлен Perl 5.8, который используется для интерпретации шаблона. Пример такого шаблона приведен в файле report.phtml, включенного в поставку.
binary	Шаблон страницы представляет собой предварительно скомпилированную бинарную динамическую библиотеку, обычно написанную на C++. Пример исходных кодов библиотеки содержится в каталоге sources_sample/report из комплекта поставки.

Директивы секции QueryCache

Секция **QueryCache** предназначена для описания политики кеширования результатов выполнения поисковых запросов.

По умолчанию, если данная секция отсутствует, поисковые запросы не кешируются, то есть каждое обращение к поисковому серверу сопровождается выполнением поиска по индексным файлам.

Если секция **QueryCache** имеется, результаты выполненных запросов временно сохраняются в специальном каталоге, и время на повторную обработку недавно выполненного запроса не тратится.

Необходимость кеширования определяется размером индексных файлов и интенсивностью запросов. При малой нагрузке в кешировании нет необходимости, что упрощает администрирование поискового сервера.

В секцию входят следующие директивы:

Директива	Описание	Значение
Dir	Каталог для размещения кешированных поисковых запросов.	Значение по умолчанию: *системный каталог для временных файлов*.
LifeTime	Время в минутах, в течение которого выполненный поисковый запрос хранится в кеше. Если значение равно нулю, запросы будут храниться в кеше бесконечно долго.	Значение по умолчанию: *60*.

Метапоиск и его настройка

- Метапоисковые источники.
- Конфигурация метапоисковых источников.
- Директивы конфигурационного файла метапоиска.

Метапоисковые источники

Поисковый сервер может работать в так называемом *метапоисковом режиме*. В этом случае выполняются поиски по одному или нескольким поисковым источникам. Результаты поисков по каждому источнику объединяются в окончательный результат, форма представления которого задается точно также, как и в режиме обычного поиска (см. Формирование страниц с результатами поиска и Директивы секции SearchPageTemplate).

Дополнительные индексы для метапоиска и соответствующие им коллекции документов называются *метапоисковыми источниками* и описываются в секциях SearchSource, по одной секции на индекс. Эти дополнительные индексы созданы другими поисковыми модулями (или индексаторами, совместимыми с ним) на этом же или на другом компьютере.

Конфигурация метапоисковых источников

Пример конфигурационного файла метапоисковых источников

```
<Collection>IndexDir : mainindex
    ...
    <SearchSource>CgiSearchPrefix : http://searcher1:17000/source
    </SearchSource>
    <SearchSource>CgiSearchPrefix : http://searcher2:17000/source
    </SearchSource>
    ...
    MetaSearchOptions : ...
    <QueryCache>Dir : cache
        ...
    </QueryCache>
    <SearchPageTemplate>
        ...
    </SearchPageTemplate>
</Collection>
```

Директивы конфигурационного файла метапоиска

Секция **SearchSource** должна включать директиву CgiSearchPrefix. Поиски по удаленным источникам выполняются в своих собственных процессах, работающих на этом же или на других компьютерах. Данный метапоисковый процесс направляет удаленным источникам запросы и получает от них резуль-

таты поиска по протоколу HTTP в специальном формате, после чего объединяет полученные результаты в окончательный результат поиска.

Директива	Описание
IndexDir	Указывает путь к каталогу с индексом. Несмотря на то, что при метапоиске никакого индекса не создается, системе необходимо указать этот путь, предварительно поместив файлы, созданные при индексации любого массива.
Секция QueryCache	
В случае метапоиска, для повышения эффективности работы системы желательно наличие секции QueryCache в конфигурации поиска.	
Секция SearchSource	
CgiSearchPrefix	Указывает HTTP-адрес поисковой страницы на удаленном поисковом источнике. Например, удаленный поисковый источник является коллекцией документов поискового модуля, установленного на порту *17000* компьютера с интернет-адресом www.metasource.ru. Эта коллекция описывается в секции Collection конфигурационного файла этого удаленного поискового модуля, имеющей атрибут *id* со значением *name1*. Тогда значением данной директивы, описывающим этот удаленный источник, будет *http://www.metasource.ru:17000/name1/*.
MetaSearchOptions	Определяет метод получения цитат с найденными словами. Аргументы директивы описаны ниже. Необязательная директива.

Аргументы директивы MetaSearchOptions

Описание директивы приведено выше.

Аргумент	Описание
OneStepQuery	В случае удаленных источников получать всю информацию в одном запросе.
TwoStepQuery	В случае удаленных источников получать отрывки текста документа с найденными словами во втором запросе. Эта опция полезна для оптимизации времени отклика в случае большого числа однородных поисковых источников. Значение по умолчанию: *OneStepQuery*.

Формирование страниц с результатами поиска

- Механизм взаимодействия с поисковым сервером.
- Поля поисковой формы, используемые Яндекс.Сервером.
- Использование функций формирования страниц (C++ и Perl).

Механизм взаимодействия с поисковым сервером

Поисковый модуль ожидает поступления HTTP-запросов по указанному в конфигурации сервера порту. Сервер анализирует запрошенные URL, а именно путь (*abs_path* в терминах RFC 2616, часть URL до

знака вопроса) и запрос (*query*, часть URL после знака вопроса, содержащую значения полей поисковой формы), выполняя перечисленные ниже действия в указанном порядке.

- Анализируется путь запрошенного URL.
  - Если путь совпадает со строкой "/admin", сервер выдает административную страницу.
  - Если путь начинается с подстроки "/images/", сервер выдает статическую картинку.
  - Если путь начинается с подстроки "/hl", сервер показывает "подсвеченный" документ, указанный в поле запроса url, в котором выделены слова, релевантные запросу, содержащемуся в поле text.
  - Если предыдущие условия не выполняются, путь (без начального слеша) рассматривается как название поискового сервера, заданное в атрибуте id секции Collection конфигурационного файла сервера. В частном случае одной коллекции документов это имя может быть пустым.

    Если поисковый сервер с указанным именем найден, анализируется запрос. В противном случае показывается страница с сообщением об ошибке.
- Если запрос содержит поле where со значением "1", запрос перенаправляется на сервер www.yandex.ru, в качестве текста запроса используется значение поля text.
- Если запрос пустой, показывается поисковая форма с параметрами по умолчанию.
- Если запрос содержит поле q, в котором передается идентификатор предварительно выполненного запроса, то будет загружена следующая страница результатов этого запроса, указанная в поле p, при условии, что
  - В настройке поиска указано, что запросы надо кешировать (см. секцию QueryCache).
  - Выполненный ранее запрос не удален из кеша по истечении времени хранения.
  - Текущий запрос не содержит поля *nocache*.

  Если одно из этих условий не выполняется, значение поля q будет проигнорировано, и будет сделана попытка перевыполнить запрос, с вызовом функции UserRequest, как описано ниже.

  Чтобы эта попытка была успешной, запрос должен содержать все необходимые поля, как при первичном выполнении. Если запрос выполнен успешно, будет показана страница, указанная в поле p.
- Если не выполняется ничего из перечисленного выше, вызывается функция UserRequest, которая должна установить текст поискового запроса на языке запросов Яндекса.

  Действия, выполняемые в этой функции, обычно сводятся к преобразованию значений некоторых полей поисковой формы в текст поискового запроса.

  Реализация функции UserRequest по умолчанию использует в качестве поискового запроса значение поля text и может быть изменена в процессе настройки сервера.
- Если поисковый запрос успешно получен, выполняется собственно поиск, после чего показывается страница с результатами выполнения запроса.

  Если запрос содержит поле xml со значением "yes", результаты поиска будут представлены в формате XML. Схема результирующего XML приведена в файлах request.xs и yandex.xs, включенных в комплект поставки.

Внимание!
В текущей версии Яндекс.Сервера (2009.05) поисковые результаты в форматах XML и HTML формируются по-разному: при обработке запроса в HTML используется *НЕстрогое* соответствия, в XML – *строгое*. XML можно настроить, используя параметр мягкости в тексте запроса: добавьте значение мягости=50 в текст запроса («//50») и получится тот же результат, что и в HTML.
В следующей версии Яндекс.Сервера механизм формирования результатов XML и HTML будет идентичен.

Если поле xml со значением "yes" отсутствует, страница с результатами поиска будет сформирована в соответствии с шаблоном, указанным в секции SearchPageTemplate конфигурации поискового сервера, или в соответствии с встроенным шаблоном, если эта секция отсутствует.

- Если ни одно из предыдущих условий не выполнено, показывается страница с сообщением об ошибке.

Поля поисковой формы, используемые Яндекс.Сервером

Следующие поля задают сортировку, группировку или структурные параметры страницы с результатами поиска и могут самостоятельно устанавливаться пользователями или разработчиками сайта.

Поле	Описание
where	Значение "1" перенаправляет поисковый запрос на www.yandex.ru.
url	Сигнализирует, что должен быть показан "подсвеченный" документ.
text	В отсутствие параметра *url* сигнализирует, что должен быть выполнен новый запрос, по умолчанию содержит текст запроса
xml	Значение "yes" сигнализирует о том, что результаты должны быть представлены в XML-формате.
numdoc	Число документов на одной странице выдачи, если не указано поле *g*, по умолчанию "10".
t	Максимальное число фрагментов текста для документа при их использовании, по умолчанию "3".
ds	URL документа для поиска похожих документов.
how	Способ сортировки найденных документов, может принимать значения: *rlv* — сортировка по релевантности (степени соответствия запросу); *tm* — сортировка по дате последней модификации; *usrf* — сортировка с использованием заданной пользователем функции вычисления релевантности; *<имя группировочного атрибута>* — сортировка по значению указанного атрибута.
asc	Уточняет поле *how*, указывая сортировку в обратном порядке.
np	Уточняет поле *how*, указывая сквозную сортировку без учета групп приоритетности.
g	Задает группировку найденных документов, это поле подробно описано ниже. Полей *g* может быть несколько, по числу необходимых группировок.
	Поле *g* имеет структуру *mode.attr.ngrp.ndoc.cur*, где *mode* — тип группировки. "0" — пустая. "1" — плоская или глубокая, если атрибут иерархический. "2" — широкая, имеет смысл только для иерархического атрибута. *attr* — имя группировочного атрибута, по которому будет осуществляться группировка, в случае пустой группировки — пустая строка. *ngrp* — число групп на одной странице. *ndoc* — максимальное число документов в группе, документы сверх этого числа выкидываются на этапе выполнения группировки.

Поле	Описание
	cur — значение атрибута для того узла иерархии, группы документов которого надо показать. Группы, не имеющие этого узла среди "родителей", выкидываются на этапе выполнения группировки. Имеет смысл только в случае иерархического атрибута. Значение по умолчанию: "1" (показать всю иерархию). Заметим, что по умолчанию при выполнении запроса не делается никаких группировок. Для того чтобы получить ту или иную группировку, необходимо задать поле *g*. Группировка и сортировка происходят на этапе выполнения запроса, поэтому нельзя использовать функции работы с группами при построении выдачи, если до выполнения поиска не были заданы соответствующие поля.
fa	Задает фильтрацию найденных документов по атрибуту, это поле подробно описано ниже.

	Поле *fa* служит для фильтрации найденных документов по значениям указанных группировочных атрибутов. Такая фильтрация выполняется после выполнения поискового запроса, но до начала выполнения группировок.
	
	Использование поля *fa* для фильтрации результатов поиска по группировочным атрибутам предпочтительнее, чем использование поисковых атрибутов в поисковом запросе (если в индексе есть поисковые атрибуты с теми же названиями и числовыми значениями), так как работает быстрее.
	
	Поле *fa* может содержать несколько подвыражений, разделенных точкой с запятой. Каждое подвыражение состоит из названия атрибута, двоеточия и диапазона, в котором должен находиться указанный атрибут у прошедших фильтр документов. Диапазон указывается в виде двух чисел, разделенных дефисом. Если одно из задающих диапазон значений отсутствует, фильтр пройдут документы со значением атрибута большим или меньшим указанного.
	
	Пример:
	
	`fa=price:40-50`
	`fa=f:1-20;d:-30;t:40-`

Следующие поля являются служебными и участвуют в формировании адресов страниц с результатами поиска и подсвеченными документами.	
q	Идентификатор запроса для следующей страницы.
p	Номер страницы результатов поиска, начиная с нуля.
mime	Формат подсвеченного документа.
charset	Кодировка подсвеченного документа.
hldoclist	Адрес страницы с результатами поиска, на которой находится ссылка на подсвеченный документ.

Использование функций формирования страниц (C++ и Perl)

Функции, вызываемые поисковым сервером

В процессе *выполнения поискового запроса* поисковый сервер вызывает следующие функции, которые могут быть написаны разработчиком страницы с результатами поиска.

Если какая-либо из функций не предоставлена разработчиком поисковой страницы, будет использована реализация по умолчанию, такая же, как во встроенном дизайне.

Функции и их описание
int UserInit(IInitContext *_ictx_, void **_userHndl_) **sub UserInit** Вызывается только один раз при старте поискового сервера, позволяет в случае необходимости инициализировать глобальные переменные и объекты. Для языка Perl вызывается без параметров. Для C++ принимает два аргумента. Первый является указателем на интерфейс контекста инициализации, декларированный в файле initcb.h. Функция **IInitContext::GetCommandLine()** позволяет получить аргумент директивы Options секции SearchPageTemplate конфигурационного файла. Во втором аргументе передается адрес указателя на пользовательский контекст, который разработчик библиотеки может инициализировать для последующего использования в функции UserRequest.
int UserDone(void *_userHndl_) Вызывается только один раз при завершении поисковой сессии.

int UserSearch(void *_userHndl_, IRequestContext* _rc_)
 	
sub UserSearch
 	
Вызывается поисковым сервером один раз в начале выполнения запроса, чтобы дать возможность подменить логику обработки или выполнить несколько поисков при выполнении запроса.
 	
Если возвращает "0", выполнение запроса поисковым сервером прекращается, иначе сервер выполняет запрос, вызывая функции в той же последовательности, как и в следующем примере.
 	
Интерфейс **IRequestContext** описан в pagecb.h.
 	
`int UserSearch(void* /*userHandler*/, IRequestContext* rc)`
`  {`
`  ISearchContext* ysc = rc->CreateContext();`
`  UserHttpHeaders(rc, ysc);`
`  UserRequest(ysc);`
`  rc->Search(ysc);`
`  UserReport(ysc);`
`  rc->DestroyContext(ysc);`
`  return 0;`
`  }`

int UserHttpHeaders(ISearchContext* _ysc_) **sub UserHttpHeaders** Вызывается в самом начале формирования страницы с результатами поиска для установки дополнительных HTTP-заголовков.
void UserRequest(ISearchContext* _ysc_) **sub UserRequest** Вызывается до выполнения поискового запроса. Предназначена для преобразования полей поисковой формы в строку запроса на языке запросов Яндекса. Использование этой функции полезно, если пользователи системы не знакомы с языком запросов и используют переключатели формы для уточнения области поиска. В этой функции также можно установить поля запроса, влияющие на сортировку и группировку с помощью вызова FormFieldInsert. На C++ текст запроса следует установить с помощью функции void IReqResults::SetUserRequest(const char *_req_), на Perl запрос является возвращаемым значением.
int UserReport(ISearchContext* _ysc_) **sub UserReport** Вызывается после выполнения запроса для формирования страницы с результатами поиска. Вся структура и дизайн страницы результатов задается в этой функции.

Функции и их описание
В процессе *формирования документа с подсвеченными словами* поисковый сервер вызывает следующие функции, которые могут быть написаны разработчиком страницы с результатами поиска. Если какая-либо из функций не предоставлена разработчиком поисковой страницы, будет использована реализация по умолчанию, такая же, как во встроенном дизайне.
void HitStartBody(ISearchContext* *ysc*) **Hit::StartBody()** Верхняя вывеска-табличка в подсвеченном документе.
void HitEndBody(ISearchContext* *ysc*, int *DocChanged*, int *FoundInBody*, int *FoundInCData*) **Hit::EndBody(*DocChanged*, *FoundInBody*, *FoundInCData*)** Нижняя вывеска-табличка в подсвеченном документе. *DocChanged* — равен "0", если документ не был изменен с момента последнего индексирования, в противном случае равен "1". *FoundInBody* — найдено слов в теле документа (можно подсветить). *FoundInCData* — найдено слов в области, не разрешающей подсветку (заголовок документа, меню, многострочные области ввода).
void HitFirst(ISearchContext* *ysc*) **Hit::First()** Стрелочка-ссылка на первое найденное слово.
void HitLast(ISearchContext* *ysc*) **Hit::Last()** Стрелочка-ссылка на последнее найденное слово.
void HitLeft(ISearchContext* *ysc*, int *word_num*, int *word_prev_num*) **Hit::Left(*word_num*, *word_prev_num*)** Стрелочка-ссылка на предыдущее найденное слово. *word_num* — номер найденного слова. *word_prev_num* — номер предыдущего найденного слова.
void HitRight(ISearchContext* *ysc*, int *word_num*) **Hit::Right(*word_num*)** Стрелочка-ссылка на следующее найденное слово. *word_num* — номер найденного слова.

Функции, вызываемые разработчиком страницы результатов

В функциях, вызываемых поисковым сервером, разработчик страницы с результатами поиска или с подсвеченным документом может для получения дополнительной информации использовать следующие функции, имплементированные в поисковом сервере.

Функции и их описание
Свойства поисковой сессии, независимые от данного поиска
int IIndexProperty::AttrCount() **Yx::AttrCount()** Возвращает число атрибутов в базе группировочных атрибутов.
const char *IIndexProperty::Attr(int *attrnum*) **Yx::Attr(*attrnum*)**

Функции и их описание
Возвращает имя каждого группировочного атрибута. *attrnum* — номер атрибута по порядку, от нуля до значения, возвращаемого AttrCount().
const char *IIndexProperty::CategName(const char **attr*, long *cat*) **Yx::CategName(*attr*, *cat*)** Возвращает строковый идентификатор данного значения группировочного атрибута. *attr* — имя группировочного атрибута. *cat* — числовое значение атрибута. Функцию можно использовать, если в индексном каталоге есть файл `attr.c2n`, где "attr" — значение первого аргумента функции (подробности см. в разделе База группировочных атрибутов).
long IIndexProperty::CategParent(const char **attr*, long *cat*) **Yx::CategParent(*attr*, *cat*)** Возвращает родительское значение данного значения иерархического группировочного атрибута. *attr* — имя группировочного атрибута. *cat* — числовое значение атрибута. Функцию можно использовать, если в индексном каталоге есть файл `attr.c2p`, где "attr" — значение первого аргумента функции (подробности см. в разделе База группировочных атрибутов).
time_t IIndexProperty::IndexModTime() **Yx::IndexModTime()** Возвращает время создания индекса в секундах от 00:00:00 1 января 1970 до момента последней модификации файла `indexkey`.
const char *IIndexProperty::ImagesUrl() **Yx::ImagesUrl()** Возвращает веб-адрес встроенного каталога с картинками. Возвращаемое значение можно использовать, если в дизайне страниц результатов и подсветки используются изображения, выдаваемые Яндекс.Сервером. Фактически функция возвращает строку вида ***http://host:port/images/***, где "host" — значение директивы Host, а "port" — значение директивы Port секции Server конфигурационного файла поискового модуля.

const char *IIndexProperty::ConfigParam(const char **directive*)
 	
Yx::ConfigParam(*directive*)
 	
Возвращает аргумент указанной директивы конфигурационного файла.
 	
Директива должна находится в подсекции **UserParams** секции Collection конфигурационного файла.
 	
Подсекция **UserParams** может содержать любое число директив с произвольными ключами, значения аргументов которых могут быть использованы при формировании страницы с результатами поиска, например:
 	
`const char* MyTest = ysc->IndexProperty()->ConfigParam("MyTest");`
`if (MyTest && strcmp(MyTest, "yes") == 0)`
`  YxPrint(ysc, "Test!");`
 	
directive — ключ директивы, расположенной в подсекции **UserParams** секции Collection конфигурационного файла.

const char *IReqEnv::SearchUrl() **Yx::SearchUrl()** Возвращает префикс веб-страницы, принимающей поисковые запросы. Возвращаемое значение нужно использовать в качестве значения атрибута *action* html-тага *form*.

Функции и их описание
Фактически функция возвращает строку вида ***http://host:port/id***, где "host" и "port" — значения директив Host и Port секции Server конфигурационного файла поискового модуля, а ***id*** — идентификатор коллекции документов, то есть значение одноименного атрибута секции Collection конфигурационного файла поискового модуля.
Данные о поисковом запросе До и после выполнения поискового запроса можно получить или изменить данные о нем.
const char *IReqEnv::HeaderIn(const char* *key*) **Yx::HeaderIn(*key*)** Возвращает значение HTTP-заголовка *key*, переданного клиентом в запросе к поисковому веб-серверу. Если указанного заголовка нет, возвращает "0" (C++) или неопределенное ("undef") значение (Perl). *key* — название HTTP-заголовка в запросе
const char *IReqEnv::Environment(const char* *key*) **Yx::GetEnv(*key*)** Возвращает значение одной из переменных окружения, указанных ниже. Если указанной переменной нет, возвращает "0" (C++) или неопределенное ("undef") значение (Perl). В переменной *key* может передаваться одна из следующих строк: "QUERY_STRING" — часть URL поисковой страницы после знака вопроса, содержащая значения полей поисковой формы. "SERVER_NAME" — доменное имя поискового сервера. "SERVER_PORT" — порт, на котором работает поисковый сервер. "SCRIPT_NAME" — часть относительного URL поисковой страницы до знака вопроса, abs_path в терминах RFC 2616. "REMOTE_ADDR" — если в поисковом запросе есть HTTP-заголовок "X-Real-IP", будет возвращено значение этого заголовка, в противном случае — IP-адрес клиента, задавшего поисковый запрос.
int IReqEnv::FormFieldTest(const char* *key*, const char* *value*) **Yx::FormFieldTest(*key*, *value*)** Возвращает "1", если есть поле формы key со значением value, в противном случае — "0". *key* — имя поля формы. *value* — значение поля формы.
int IReqEnv::FormFieldCount(const char* *key*) **Yx::FormFieldCount(*key*)** Возвращает число одноименных полей. *key* — имя поля формы.
const char *IReqEnv::FormField(const char* *key*, int *num*, bool *encode* = false) **Yx::FormField(*key*, *num* = 0)** Возвращает значение поля формы *key*. Если запрашиваемого поля нет, возвращает "0" (C++) или неопределенное ("undef") значение (Perl). *key* - имя поля формы. *num* — номер значения, начиная с 0. *encode* — флаг, указывающий, нужно ли перекодировать запрещенные в HTML символы.
const char *IReqEnv::QueryString() **Yx::QueryString()** Возвращает все поля поисковой формы в формате адреса веб-страницы.

Функции и их описание
Возвращаемое значение может отличаться от значения Environment("QUERY_STRING"), так как учитывает результаты вызовов FormFieldInsert и FormFieldRemove из UserRequest.
void IReqEnv::FormFieldInsert(const char* *key*, const char* *value*) **Yx::FormFieldInsert(*key*, *value*)** Функция вставляет в список переданных полей поисковой формы новое значение. Эффект такой же, как если бы поле было передано из формы на веб-странице. *key* — имя поля формы. *value* — значение поля формы, которое нужно вставить.
void IReqEnv::FormFieldRemove(const char **key*, int *num*) **Yx::FormFieldRemove(*key*, *num*)** Функция удаляет данное поле из списка переданных полей. *key* - имя поля формы. *num* — номер значения, которое надо удалить, начиная с 0.
Статистика, относящаяся к запросу в целом
const char *IReqResults::ReqId() **Yx::ReqId()** Возвращает идентификатор запроса, назначенный поисковым сервером.
const char *IReqResults::Request() **Yx::Request()** Возвращает текст выполненного запроса, который может отличаться от введенного пользователем вследствие модификаций, выполненных в функции UserRequest.
int IReqResults::ErrorCode() **Yx::ErrorCode()** При неудачном поиске возвращает код ошибки.
const char *IReqResults::ErrorText(const char* *lang* = "ru") **Yx::ErrorText(*lang*)** При неудачном поиске возвращает описание ошибки. *lang* — строка-идентификатор языка.
const char *IReqResults::WordStat() **Yx::WordStat()** Возвращает строку со статистикой найденных слов в формате *слово:количество*.
int ICluster::TotalDocCount(int *prior*) **Yx::TotalDocCount(*prior* = 0)** Возвращает число найденных документов с данным приоритетом релевантности. *prior* — приоритет релевантности. Значение "2" соответствуют совпадению фразы, "1" — строгому соответствию запросу, "0" — нестрогому соответствию запросу. В числе, возвращаемом при меньшем значении параметра, суммируется количество документов с худшим приоритетом, так что при значении "0" функция возвращает полное число найденных документов.
Свойства группировки
long ICluster::CurCateg(const char **attr*, int *gMode*)

Функции и их описание
Yx::CurCateg(*attr, gMode*) Возвращает текущую группу (числовое значение атрибута), заданную при определении группировки. *attr* — имя группировочного атрибута. *gMode* — тип группировки, "1" — плоская или глубокая, если атрибут иерархический, "2" — широкая, имеет смысл только для иерархического атрибута.
const char *IReqEnv::SearchPageUrl(int *npage*, const char **attr*) **Yx::SearchPageUrl(*npage, attr*)** Возвращает веб-адрес одной из страниц с найденными документами данной группировки. Этот адрес всегда начинается со строки, возвращаемой функцией SearchUrl. *npage* — номер страницы по порядку, начиная с нуля. *attr* — имя группировочного атрибута.
int ICluster::Count(int *prior*, const char **attr*, int *gMode*) **Yx::Count(*prior, attr, gMode*)** Возвращает число найденных групп в данной группировке, полученное в процессе выполнения запроса. *prior* — приоритет релевантности, как описано для функции TotalDocCount.
Свойства группы в данной группировке
long ICluster::GroupCateg(int *nGroup*, const char **attr*, int *gMode*) **Yx::GroupCateg(*nGroup, attr, gMode*)** Возвращает числовое значение группировочного атрибута, соответствующее данной группе. *nGroup* — номер группы по порядку, начиная с нуля на первой странице, и с произведения *(номер страницы)*(количество групп на странице)* на последующих. *attr* — имя группировочного атрибута. *gMode* — тип группировки, "1" — плоская или глубокая, если атрибут иерархический, "2" — широкая, имеет смысл только для иерархического атрибута.
int ICluster::GroupDocCount(int *prior*, int *nGroup*, const char **attr*, int *gMode*) **Yx::GroupDocCount(*prior, nGroup, attr, gMode*)** Возвращает число документов, попавших в данную группу в процессе группировки. *prior* — приоритет релевантности, как описано для функции TotalDocCount. Полному числу документов в группе соответствует значение "0".
int ICluster::GroupRelevance(int *nGroup*, const char **attr*, int *gMode*) **Yx::GroupRelevance(*nGroup, attr, gMode*)** Возвращает релевантность группы в виде положительного целого числа.
int ICluster::GroupPriority(int *nGroup*, const char **attr*, int *gMode*) **Yx::GroupPriority(*nGroup, attr, gMode*)** Возвращает приоритет релевантности группы. Значения "5", "6", "7" соответствуют совпадению фразы, "2", "3", "4" — строгому соответствию запросу, "0", "1" — нестрогому соответствию запросу.
Свойства документа в данной группе
long ICluster::DocId(int *nGroup*, int *nDoc*, const char **attr*, int *gMode*) **Yx::DocId(*nGroup, nDoc, attr, gMode*)** Возвращает числовой идентификатор документа, назначенный при индексировании.

Функции и их описание
nGroup — номер группы по порядку, начиная с нуля на первой странице, и с произведения (номер страницы)*(количество групп на странице) на последующих. *nDoc* — номер документа в группе по порядку, от нуля до значения "GroupDocCount(...)-1" включительно. *attr* — имя группировочного атрибута. *gMode* — тип группировки, "1" — плоская или глубокая, если атрибут иерархический, "2" — широкая, имеет смысл только для иерархического атрибута.
int ICluster::DocPriority(int *nGroup*, int *nDoc*, const char **attr*, int *gMode*) **Yx::DocPriority(*nGroup*, *nDoc*, *attr*, *gMode*)** Возвращает приоритет релевантности документа, как описано для функции GroupPriority.
long ICluster::DocRelevance(int *nGroup*, int *nDoc*, const char **attr*, int *gMode*) **Yx::DocRelevance(*nGroup*, *nDoc*, *attr*, *gMode*)** Возвращает релевантность документа в виде положительного целого числа.
const char *IReqEnv::HighlightedDocUrl(int *nGroup*, int *nDoc*, const char **attr*, int *gMode*) **Yx::HighlightedDocUrl(*nGroup*, *nDoc*, *attr*, *gMode*)** Возвращает веб-адрес страницы с подсвеченным документом. Этот адрес всегда начинается со строки, возвращаемой функцией SearchUrl.
int ICluster::DocPropertyCount(int *nGroup*, int *nDoc*, const char **attr*, int *gMode*, const char* **property*) **Yx::DocPropertyCount(*nGroup*, *nDoc*, *attr*, *gMode*, *property*)** Возвращает число различных значений свойства документа, указанного в аргументе *property*. *property* — название свойства в виде строки. В качестве свойства можно передавать следующие значения. • Название группировочного атрибута, которое возвращается функцией Attr(attrnum). В этом случае функция вернет "0", если у данного документа указанный группировочный атрибут отсутствует, "1", если атрибут имеется, или значение, большее 1, если имеется несколько значений иерархического атрибута. • Один из аргументов директивы DocProperty конфигурации индексатора. В этом случае функция вернет "0", если у данного документа указанный поисковый атрибут отсутствует, и "1", если атрибут имеется. • Одно из следующих значений. "_Url" — URL документа. "_ModTime" — время в секундах, прошедшее с 00:00:00 1 января 1970 года до момента последнего изменения файла с документом. "_ModTimeStr" — то же, что и "_ModTime", но в виде строки формата *18 Aug 2000, 08:52*. "_Size" — размер файла с документом в байтах. "_Charset" — кодировка документа. "_MimeType" — медиа-тип документа. "_Title" — заголовок документа. "_HeadLine" — краткое содержание документа или первые 200 символов текста. "_Passage" — отрывок текста документа, содержащий искомые слова. Функция вернет "1", кроме свойства "_Passage", значений которого может быть несколько, но не больше, чем указано в параметре поисковой формы *t*.
const char *ICluster::DocProperty(int *nGroup*, int *nDoc*, const char **attr*, int *gMode*, const char* **property*, unsigned *index*) **Yx::DocProperty(*nGroup*, *nDoc*, *attr*, *gMode*, *property*, *index*)**

Функции и их описание
Возвращает значение указанного свойства документа. *property* — название свойства в виде строки, возможные значения перечислены в описании функции DocPropertyCount. *index* — номер значения свойства от нуля до значения, возвращаемого функцией DocPropertyCount. Для свойств "_Title", "_HeadLine" и "_Passage" функция возвращает строку, размеченную специальным символом BEL (символ с кодом "7") следующим образом. • "BEL{" и "BEL}" окружают слова с приоритетом "совпадение фразы". • "BEL[" и "BEL]" окружают слова с приоритетом "строгое соответствие". • "BEL(" и "BEL)" окружают слова с приоритетом "нестрогое соответствие". Размеченные таким образом строки могут быть преобразованы с помощью функции ConvertArchiveText.
const char* IReqEnv::ConvertArchiveText(const char* *rawtext*, unsigned *maxLen*, int *mode*, const char **Open1*, const char **Open2*, const char **Open3*, const char **Close1*, const char **Close2*, const char **Close3*) **Yx::ConvertArchiveText(*rawtext, maxLen, mode, Open1, Open2, Open3, Close1, Close2, Close3*)** Функция преобразует заголовок, аннотацию и фрагменты текста документа, возвращаемые функцией DocProperty. Возвращает текст, в котором специальные символы заменены на указанные теги и который обрезан до нужной длины, по возможности, по словам. *rawtext* — размеченный текст, выданный функцией DocProperty, который надо обработать. *maxLen* — длина, до которой надо укоротить *rawtext*, если этот параметр равен "0", то текст не укорачивается. *mode* — вид обработки. "0" — запрещенные в HTML символы заменяются на соответствующие последовательности, а специальная разметка слов — на указанные теги. "1" — запрещенные в HTML символы заменяются на соответсвующие последовательности, а специальная разметка слов выбрасывается. "2" — запрещенные в HTML символы не заменяются, специальная разметка слов выбрасывается. *Open1*, *Close1* — Открывающий и закрывающий теги для слов, найденных с нестрогим соответствием. *Open2*, *Close2* — Открывающий и закрывающий теги для слов, найденных со строгим соответствием. *Open3*, *Close3* — Открывающий и закрывающий теги для слов, найденных с совпадением фразы.
Функции формирования HTTP-заголовков на Perl
Yx::HeaderOut(*header, value*) Функция печатает HTTP-заголовок *header* со значением "value". *header* — HTTP-заголовок. *value* — значение HTTP-заголовка.
Yx::ContentType(*value*) *value* — значение заголовка. Функция печатает HTTP-заголовок *Content-Type* со значением "value".
Yx::SetLastModified() Функция печатает HTTP-заголовок *Last-Modified* со временем изменения индексных файлов в качестве значения заголовка.

Примеры использования C++

В данном разделе приведены простейшие примеры использования описанных выше функций на языке C++.

```
int UserHttpHeaders(ISearchContext* ysc)
```

```
int UserHttpHeaders(ISearchContext* ysc)
{
    YxPrint(ysc, "Content-Type: text/html\n");
    YxPrint(ysc, "Cache-Control: max-age=3600\n");
    YxPrint(ysc, "Last-Modified: ");
    YxPrintHttpTime(ysc, ysc->IndexProperty()->IndexModTime());
    YxPrint(ysc, "\n");
    return 0;
}
```

void UserRequest(ISearchContext* *ysc*)

Текст запроса на языке запросов Яндекса следует установить с помощью функции **SetUserRequest**.

```
void UserRequest(ISearchContext* ysc)
{
    const char* p = ysc->ReqEnv()->FormField("text", 0);
    if (!p || !strlen(p)) {
        ysc->ReqResults()->SetUserRequest("");
        return;
    }
    char Query[1024];
    strncpy(Query, p, 200);

    // поиск документов, измененных за последние N дней,
    // где N определяется cgi-параметром within
    const char* within = ysc->ReqEnv()->FormField("within", 0);
    if (within && within[0] != '0') {
        char Temp[512];
        int days = atol(within);
        time_t beg_time = time(0) - days * 3600 * 24;
        strftime(Temp, 1024, "&&#date>=\"%Y%m%d\"", localtime(&beg_time));
        strcat(Query, Temp);
    }
    ysc->ReqResults()->SetUserRequest(Query);
}
```

int UserReport(ISearchContext* *ysc*)

```
int UserReport(ISearchContext* ysc)
{
    YxPrint(ysc, "<html><head><title>Список найденных документов</title></head>");
    YxPrint(ysc, "<body><p>");
    const char *stat = ysc->ReqResults()->WordStat();
    if(stat && *stat) {
        YxPrint(ysc, "Результат поиска: ");
        YxPrint(ysc, stat);
        YxPrint(ysc, "<br>\n");
        YxPrint(ysc, "Найдено документов: <b>");
        YxPrint(ysc, ysc->Cluster()->TotalDocCount(0));
        YxPrint(ysc, "</b>\n");
    }
    YxPrint(ysc, "</body></html>");
    return 0;
}
```

Примеры использования Perl

В данном разделе приведены простейшие примеры использования описанных выше функций на языке Perl. Реальные работающие примеры можно найти в файле report.phtml, включенном в комплект поставки программы.

sub UserInit

```
sub UserInit {
  # Каталог, содержащий картинки
  $::PICTURE_DIR  = Yx::ImagesUrl();
  # количество документов в списке найденного
  $::DEF_NUM_DOC  = 10;
  # количество отображаемых для каждого документа контекстов
  $::DEF_NUM_PASSAGES = 3;
  # Включить в форму поиск похожих документов
  $::USE_SIMILAR_DOCS_SEARCH = 1;
}
```

sub UserHttpHeaders

```
sub UserHttpHeaders {
  Yx::ContentType ("text/html");
  Yx::HeaderOut("Cache-Control", "max-age=3600");
  Yx::SetLastModified();
}
```

sub UserRequest

```
sub UserRequest {
    my $text = Yx::FormField("text");
    return "" if $text eq "";

    my $req = $text;

    # поиск документов, измененных за последние N дней,
    # где N определяется cgi-параметром within
    my $within = Yx::FormField("within");
    if ($within != 0) {
        my $beg_time = time() - ($within * 3600 * 24);
        my ($sec,$min,$hour,$mday,$mon,$year) = localtime($beg_time);
        $req .= "&&#date>=".sprintf("\"%04d%02d%02d\"", $year+1900, $mon+1, $mday);
    }
    return $req;
}
```

sub UserReport

Перловый модуль, используемый для настройки поисковой выдачи, представляет собой HTML-файл с участками кода на Perl 5, заключенными между разделителями <% и %>:

```
<%
sub UserReport {
%>
<html><head><title>Список найденных документов</title></head>
<body><p>
<%
    my $stat = Yx::WordStat();
    if ($stat ne "") {
        print "Результат поиска: $stat<br>\n";
        print "Найдено документов: <b>".Yx::TotalDocCount()."</b>\n";
    }
%>
</body></html>
<%
return 0;
}
%>
```

Язык запросов

Предварительные сведения

- Документы и словопозиции.
- Документные атрибуты.
- Зоны и зонные атрибуты.
- Язык запросов и алгоритмы фильтрации.

Для полного понимания всех деталей языка запросов необходимы некоторые знания структуры поискового индекса и механизмов выполнения запроса, а также понимание терминологии зон и атрибутов. Эти вопросы рассматриваются в данном разделе.

Документы и словопозиции

В дальнейшем термин **документ** используется для обозначения предмета **поиска**. Документ содержит сведения, нужные пользователю. Документы входят в **коллекцию документов**, которая подвергается процедуре **индексирования**, для того чтобы стал возможен поиск по этой коллекции.

Все слова, встречающиеся в коллекции документов, приводятся к своей канонической словарной форме (**лемматизируются**). Полученные **леммы** запоминаются в **индексе** и в дальнейшем называются **ключами**. Для каждого ключа запоминается список **словопозиций**, или просто позиций.

Каждая словопозиция ключа-леммы включает:

- уникальный **идентификатор документа**, присвоенный ему во время индексирования;
- номер предложения, в котором встретился ключ;
- номер слова в предложении;
- вес, назначенный данной позиции во время индексирования;
- номер словоформы в списке данного ключа.

Эти данные позволяют указывать в языке запросов требуемое расстояние между словами и предложениями.

Типичный пример представления леммы в индексе (слова с заглавной буквы считаются отдельными формами):

```
признавать(признав,признавал,признавать,признает,Признан,признать)
[2.20.1.1.4]
[4.132.25.1.5]
[4.153.26.1.1]
[4.171.9.1.0]
[4.171.40.1.2]
[4.269.28.1.3]
[13.90.3.1.5]
```

В дополнение к словам, встречающимся в документе, ключами индекса (и, соответственно, частью языка запросов) служат названия поисковых зон и атрибутов документа (подробнее см. в разделе Документные атрибуты). Словопозиции некоторых атрибутов вместо номеров предложений и слов содержат другую информацию (подробности можно найти в следующих разделах).

Поисковый запрос, сформулированный на **языке запросов Яндекса**, указывает, какие ключи надо найти в индексе, и какие операции (объединение, пересечение, взятие диапазона и т. п.) надо произвести над соответствующими списками словопозиций. Результатом выполнения запроса является список документов из итогового множества словопозиций.

Документные атрибуты

Каждый документ может иметь некоторый набор свойств, таких как заголовок, дата создания, размер, принадлежность к определенному разделу сайта или каталога. Такие свойства документа будем называть **документными атрибутами**, или просто **атрибутами**. Атрибуты могут быть непосредственно не связаны с текстом документа, видимым при его просмотре.

Каждый атрибут документа получает идентификатор в виде текстовой строки, который в дальнейшем будем называть **именем атрибута**. Само свойство документа будем называть **значением атрибута**. Документ в общем случае может иметь произвольное число атрибутов. Каждый атрибут документа может иметь несколько различных значений. С другой стороны, не все атрибуты, определенные в коллекции документов, должны быть определены для данного документа.

Рассмотрим коллекцию из двух документов, включающих литературные произведения. Каждый из них имеет документный атрибут *datecreated*, его значением является дата написания произведения его автором. Первый документ имеет атрибут *author* со значением "Pushkin", а у второго документа этот атрибут имеет два значения — "Sheckley" и "Zelazny", так как произведение написано в соавторстве. Наконец, у первого документа имеется атрибут *inmeter* со значением "iambus", а у второго документа этот атрибут отсутствует.

В дальнейшем будут рассматриваться только **поисковые атрибуты**, имена которых используются в языке запросов. (О **группировочных** и **архивных** атрибутах см. в разделе Документы, зоны и атрибуты Поисковые документные атрибуты являются частным случаем **зонных** атрибутов, рассмотренных ниже. Словопозиции, соответствующие документным атрибутам, не содержат номеров слов и предложений.

Зоны и зонные атрибуты

Большинство документов имеет внутреннюю структуру, которая позволяет выделить различные части документов — поисковые **зоны**.

Примеры поисковых зон: параграфы, заголовки различных уровней, ссылки, картинки. Формат электронного письма подразумевает наличие в нем полей from, to, служебных областей, поля сообщения и т. п.

Каждая зона может иметь одно или несколько свойств, не связанных непосредственно с ее текстом. Такие свойства зон будем называть **зонными атрибутами**. Зона в общем случае может иметь произвольное число атрибутов. Каждый атрибут может иметь несколько различных значений. Разумеется, не все атрибуты, определенные в данном массиве документов, должны быть определены для данной зоны.

Зона «to» в электронном письме могла бы иметь свойство «адреса получателей» или «количество адресов рассылки», а зона «ссылка» на другой документ в HTML-документе имеет только свойство «URL документа».

Документ размечается на зоны во время индексирования. Каждая зона получает уникальное **имя зоны** (текстовую строку), используемое в языке запросов, и может быть предметом поиска независимо от других зон. Мы будем называть эти размеченные области документа поисковыми зонами.

Каждая поисковая зона имеет точки начала и конца в теле документа, которые всегда приходятся на границы слов. Таким образом, каждой зоне в индексе соответствуют два ключа, словопозиции которых содержат номера слов и предложений, в которых зона начинается и заканчивается.

Зонные атрибуты также могут быть помечены в качестве независимых объектов поиска. Такие атрибуты будем называть зонными поисковыми атрибутами, или просто атрибутами. Каждый атрибут имеет уникальное имя (текстовую строку) и значение, которое может быть различного типа, в зависимости от способа его обработки при индексировании. Каждой паре (имя, значение) атрибута соответствует один ключ индекса, словопозиции которого содержат номера слов и предложений, в которых начинается соответствующая зона.

Важным частным случаем зоны является зона нулевой длины. Она используется для того, чтобы иметь возможность назначить атрибуты определенной точке внутри документа. Это может быть полезным в случае, когда документ имеет "вложенные потоки", отличающиеся форматом или медиа-типом.

Картинка из HTML-документа расположена в отдельном файле с заданным именем. Это имя — свойство (атрибут) точки документа, в которой расположена картинка.

Язык запросов Яндекса позволяет искать в нужной зоне с нужным значением атрибута.

Язык запросов и алгоритмы фильтрации

Язык запросов Яндекса представляет собой контекстно-свободную грамматику, терминалами которой служат слова естественного языка, а также имена зон и атрибутов.

В результате анализа запроса формируется набор ключей, по которым из индекса загружаются соответствующие списки словопозиций — **листовые ноды** или просто **листы**. Полученные списки позиций затем комбинируются и фильтруются в соответствии с операторами языка в два этапа.

Сначала формируются несколько списков **эффективных позиций**. Каждая эффективная позиция является либо **элементарной позицией**, соответствующей листовой ноде, либо **скобкой**. Позиции-скобки получаются комбинированием операций пересечения (строгого многоместного *AND*) и объединения *OR* и, в отличие от элементарных позиций, имеют длину, — то есть позиции начала и конца. Разбиение строки запроса на скобки осуществляется как явным заданием скобок, так и при помощи использования приоритета операций.

На заключительном этапе выполняется **многоместный оператор AND**, который может быть **мягким**.

К листу, скобке как к целому, а также к многоместному *AND* верхнего уровня могут быть последовательно применены один или несколько **фильтров**. Фильтры могут быть **простыми** и **ранжирующими**. Простые фильтры отфильтровывают ненужные позиции, а ранжирующие не изменяют множество позиций, но устанавливают свойство, что в итоговом результате документы, в которых есть только позиции, не прошедшие фильтр, будут гарантированно ниже остальных документов.

В дальнейшем мы будем обозначать латинскими буквами *a*, *b*, *c*, . . . лист или скобку.

Основные операторы

- Объединение OR.
- Многоместный AND.
- Задание расстояния.
- Префиксы и кавычки.
- Простые фильтры.
- Ранжирующие фильтры.

Большинство поисковых запросов задается пользователями на «естественном языке», то есть без использования операторов языка. Поэтому для уменьшения числа «ложных срабатываний» для всех бинарных операторов &, &&, ~, ~~, |, <<, <–, описанных ниже, требуется наличие пробелов с каждой стороны. Если пробел отсутствует, операторы будут проигнорированы.

Объединение OR

Представляет собой цепочку листов или скобок, разделенных операторами |. В результате выполнения операции множества позиций объединяются. Семантика операции объединения — задание терминов-синонимов (или выражений-синонимов).

Многоместный AND

Представляет собой цепочку листов или скобок, разделенных операторами & и && с возможным указанием расстояния. Одинарный оператор & соответствует пересечению внутри предложения, а двойной && — пересечению внутри документа. Если оператор не указан, то есть листы или скобки разделены пробелом, применяется двойной оператор, то есть дополнительная фильтрация отсутствует.

Задание расстояния

Если слова в тексте перенумеровать по порядку их следования, то расстояние между словами *a* и *b* — это разница между номерами слов *a* и *b*. Таким образом, расстояние между соседними словами равно 1, а не 0.

При задании расстояния существенен порядок следования слов. Высказывание *на расстоянии в –2 слова* означает *на расстоянии в 2 слова слева*, а высказывание *на расстоянии в +2 слова* означает *на расстоянии в 2 слова справа*.

Аналогично определяется расстояние между предложениями (если заменить термин *слово* на термин *предложение*).

Синтаксис задания расстояния: /(*mn*), при этом число *m* должно быть не больше, чем число *n*.

Применяются различные сокращенные формы:

- /*n* равнозначно /(–*n* +*n*)
- /+*n* равнозначно /(+*n* +*n*)
- /–*n* равнозначно /(–*n* –n)

Если задана область действия оператора, но не задан сам оператор, то применяется оператор &, то есть *a* /2 *b* эквивалентно запросу *a* &/2 *b*.

a & *b* — *a* и *b* должны находиться в одном предложении.

a /+1 *b* — *b* должно следовать за *a*.

a /2 *b* — расстояние в словах независимо от порядка следования (то же, что и *a* &/2 *b*).

a /(–2 4) *b* — *b* должно находиться от *a* в интервале расстояний от 2 слов слева до 4 слов справа.

a /0 *b* — расстояние в словах, позиции должны совпадать (то же, что и *a* &/0 *b*).

a && *b* — *a* и *b* должны находиться в одном документе.

a &&/1 *b* — *b* должно находиться в том же предложении, что и *a*, либо в соседнем.

a &&/2 *b* — расстояние в предложениях.

a &&/0 *b* — расстояние в предложениях (то же, что и *a* & *b*).

Префиксы и кавычки

Перед листом могут быть указаны некоторые префиксные операторы. Префикс может быть и у скобки, в этом случае он распространяется на все ее листы.

! — поиск точной формы (отключение морфологии)

По запросу `!лужков` будут найдены словоформа *лужков*, но не *лужкову*, *лужки*.

В общем случае слова с большой и маленькой буквы считаются разными формами одного слова, поэтому все равно, какой регистр использовать в запросе. Исключением является оператор точной формы. По запросу `!лужков` будут найдены все документы, содержащие эту словоформу в любом регистре, а по запросу `!Лужков` — только документы, в которых имеется форма *Лужков* с большой буквы.

!! — поиск точной леммы
Указанный ключ считается леммой. Будут найдены все формы этой леммы, но не формы других лемм, отличных по написанию от указанной, но таких, которым соответствует данный ключ, рассматриваемый как форма.

Если указанный ключ является словарной леммой, поиск ограничивается этим ключом, иначе запрос дополняется операторами точной формы.

По запросу `!!лужков` найдутся формы леммы *лужков*, но не леммы *лужок*, т. е. найдутся формы *лужкову*, *лужкова*, но не *лужки*.

Выражение в кавычках применяется для поиска точной фразы и почти эквивалентно многоместному *AND* на уровне предложения с расстояниями в +1 между соответствующими точными формами. Небольшое отличие в том, что для выражения в кавычках регистр точной формы игнорируется.

Запрос *"рисовать квадрат и круг"* эквивалентен запросу !*рисовать* /+1 !*квадрат* /+1 !*и* /+1 !*круг*.

Оператор расстояния /+*n* в кавычках не распознается. Вместо него поддерживается конструкция *"a * * b"*, которая эквивалентна запросу !*a* /+3 !*b*. Символы звездочки должны быть отделены пробелами.

*"a * b|"*,

*"раз * три * * шесть"*,

*"a * * * e f * h"*

Простые фильтры

- **Исключение***a* ~ b или a ~~ b. Возможно указание расстояния точно таким же образом, как в многоместном *AND*. В результате выполнения операции из левого множества позиций удаляются элементы, которые ближе, чем указанное расстояние, к любому из элементов правого множества словопозиций. Одинарный оператор ~ соответствует уровню предложения, а двойной ~~ — уровню документа.

> *a* ~/(–4 +8) *b* — *b* не должно находиться рядом с *a*, причем «рядом» здесь означает «в интервале расстояний от 4 слов слева до 8 слов справа».

- **Поиск в зонах** *zone:a*. В результате выполнения операции из множества *a* удаляются позиции вне указанной зоны. Более подробно зонно-атрибутивный поиск описан в разделе Поиск в зонах и атрибутах.
- **Ограничение по документам***a* << *b*. В результате выполнения операции из левого множества позиций удаляются позиции с документами, отсутствующими в правом множестве. Оператор похож на документный *AND* (&&), с той лишь разницей, что правый операнд влияет на возможность документов попасть в результаты поиска, но не влияет на ранжирование. Этот оператор следует использовать при ограничении результатов поиска атрибутивным выражением.
- **Интервальное ограничение по позициям***a inpos:lo..hi*. Назовем **величиной** словопозиции число, которое получается из словопозиции после обнуления идентификатора документа. В результате выполнения данной операции из множества словопозиций *a* удаляются позиции, величина которых лежит вне указанного интервала.

> При запросе *date:20060514 inpos:60..120* будут найдены документы, созданные 14 мая 2006 года с часу до двух ночи.

Ранжирующие фильтры

- **Уточнение***a* <– *b*. Эквивалентен операции *OR*, но правые позиции должны ранжироваться выше.
- **Ранжирующий поиск в зонах***zone–>a*. Позиции вне указанной зоны ранжируются ниже.

Поиск в зонах и атрибутах

- Синтаксические правила зонно-атрибутивного поиска.
- Типизация поисковых атрибутов.
- Конфигурирование.
- Зоны и атрибуты по умолчанию.
- Сравнение с синтаксисом версии 3.10.8.

В операторах зонно-атрибутивного поиска используются имена поисковых зон и атрибутов, заданные во время индексирования и описанные в специальном конфигурационном файле (см. раздел Конфигурирование). Если имя зоны или атрибута не описано в конфигурационном файле или значение атрибута не соответствует его типу, как описано в разделе Типизация поисковых атрибутов, вся конструкция преобразуется в набор слов. Также во всех конструкциях должны отсутствовать пробелы с обеих сторон двоеточия.

Ниже в качестве имени зоны мы будем использовать *zone*, а в качестве имени атрибута — *attr*. Вместо *zone* и *attr* следует подставить одно из действительных значений.

> В веб-поиске Яндекса (www.yandex.ru) можно использовать *title* или *anchor* вместо *zone* и *link*, *image* или *date* вместо *attr*.

Операторы зонно-атрибутивного поиска могут составлять подзапрос более сложного запроса. В этом случае они комбинируются с другими частями запроса с помощью операторов языка запросов, описанных в разделе Основные операторы.

Синтаксические правила зонно-атрибутивного поиска

- **zone:слово** — ищет *слово* в зонах с именем *zone*. Токен (слово) не может содержать пробелы.

Найти документы, в тексте которых встречается слово *ошибка*, но это слово не встречается в тексте ссылок на другие документы:
`ошибка ~~ anchor:ошибка`

- **zone:(текст-в-зоне)** — ищет *текст-в-зоне* в зонах с именем *zone*. В подзапросе *текст-в-зоне* могут употребляться любые языковые конструкции, описанные ранее, в частности, он может состоять из нескольких слов.
- **zone:"текст-в-зоне"**
- **zone:’текст-в-зоне’**
- **zone:‘текст-в-зоне‘** — ищет запрос в кавычках *"текст-в-зоне"* (то есть точные формы подряд идущих слов) в зонах с именем *zone*.
- **zone:((текст-в-зоне))** В отличие от варианта с одной парой скобок, ищет *текст-в-зоне* целиком в одной зоне с именем *zone*. Имеет смысл, когда в документе есть несколько разных одноименных зон, а *текст-в-зоне* состоит из нескольких слов.
- **attr:значение**
- **attr:"значение"**
- **attr:’значение’**
- **attr:‘значение‘**
- **attr:(значение)** — все формы записи эквивалентны, за исключением того, что в первом случае *значение* не может включать пробелы.
  Ищет документы, в которых существуют зоны (в том числе документ как целое и зона нулевой длины), имеющие атрибут с именем *attr* и значением *значение*. Возможный формат значения зависит от типа атрибута, подробности описаны в разделе Типизация поисковых атрибутов.

  Для всех типов атрибутов, кроме логических, возможно указывать операторы сравнения >, >=, <, <= после двоеточия: `date:<20090101` или `size:>=1024`

Найти документы, в которых есть картинки, расположенные в файле apple.gif:
`image:apple.gif`

В этом примере использован атрибут специальной зоны нулевой длины.

Найти документы, созданные 14 мая 1999 года:
`date:19990514`

В этом примере использован документный атрибут.

Найти все документы, которые ссылаются на сайт www.site.ru:
`link:www.site.ru/*`

В этом примере использован атрибут зоны anchor.

Таким образом, поиск документов с зонным атрибутом, встречающимся где угодно в тексте, ничем синтаксически и по смыслу не отличается от поиска по документным атрибутам.

Однако зонные атрибуты (в том числе и атрибуты пустой зоны) отличаются от документных тем, что имеют координатную привязку к тексту документа. То есть для поисковой системы это такие же объекты для поиска, как и слова, и они могут участвовать во всех допустимых операциях с заданным поисковым контекстом.

Найти все документы, в которых картинки из файла apple.gif находятся рядом со словом *яблоко*:

```
image:apple.gif &/1 яблоко
```

- **zone:(attr:значение текст)** — ищет *текст* в зонах с именем *zone*, имеющих атрибут с именем *attr* и значением *значение*. В подзапросе *текст* могут употребляться любые языковые конструкции, описанные ранее, в том числе и другие атрибуты. Отличие атрибута, стоящего на первом месте, в том, что он привязан к указанной зоне.

Найти документы, которые ссылаются на сайт www.site.ru словами *мягкая мебель*:
`anchor:(link:www.site.ru/* мягкая мебель)`

Типизация поисковых атрибутов

Поисковые атрибуты различаются по способам распознавания и обработки. В зависимости от типа возможны также некоторые вариации синтаксиса задания значения атрибута.

Встроенные зоны и атрибуты

Несколько зон и атрибутов изначально встроены в язык запросов и имеют зарезервированные имена.

- **intext** — специальное имя зоны для поиска только в текстах документов.
- **inlink** — специальное имя зоны для поиска только в ссылках на документы.
- **softness** — специальное имя атрибута для указания мягкости многоместного оператора AND верхнего уровня. Имеет синтаксис *softness:N*, где N — целое число от 0 до 100 включительно. Следует задавать не более одного softness-атрибута в запросе, так как будет использовано только последнее заданное значение.
- **prevreq** — специальное имя атрибута для указания идентификатора предварительно сохраненного запроса. Имеет синтаксис *prevreq:ID*, где ID может содержать буквы и цифры, а также может быть отрицательным целым числом.
- **inpos** — специальное имя атрибута для указания точного диапазона позиций, в которых должен находиться предыдущий лист или скобка. Имеет синтаксис *inpos:N1..N2*, где N1 и N2 — целые положительные числа.

Литеральные атрибуты

Значение атрибута распознается как неделимая последовательность символов, участвующая в индексировании и поиске как целое. Может быть документным или зонным.

Возможен поиск по префиксу, в этом случае значение атрибута (требуемый префикс) должно заканчиваться звездочкой (*).

Даты

Значение атрибута распознается как дата или время. Этот тип атрибута может быть только документным, при этом в ключах хранится дата с точностью до дня, аналогично литеральному атрибуту, а в словопозициях хранятся секунды от полуночи.

Поддерживаются несколько вариантов задания даты: *YYYYMMDD*, или *YYYY-MM-DD*, или *YYYY-MM-DDTHH:MM:SS*. В первом варианте можно использовать символ * для поиска диапазона, в последнем можно не задавать секунды или секунды и минуты, в этом случае поиск также будет выполнен по диапазону.

Поддерживается также интервальный поиск: `date:20080101..20080212`

`date:20080119` — найти документы, созданные 19 января 2008 года

`date:2008*` — найти документы, созданные в 2008 году

`date:2009-01-19T19:23:45` — найти документы, созданные 19 января 2008 года в 19 часов 23 минуты 45 секунд, тот же результат достигается запросом `date:20090119 inpos:69825..69826`

`date:2009-01-19T19:23` преобразуется в `date:2009-01-19 inpos:69780..69840`

`date:2009-01-19T19` преобразуется в `date:2009-01-19 inpos:68400..72000`

URL

Значение атрибута распознается как Uniform Resource Locator.

Значения атрибутов данного типа имеют дополнительную обработку: из них удаляется протокол — http:// или https://, а также все русские символы преобразуются в utf-8 и кодируются в соответствии с RFC1738.

`url:http://ru.wikipedia.org/wiki/Уткин_Иосиф_Павлович` работает как `url:ru.wikipedia.org/wiki/%D0%A3%D1%82%D0%BA%D0%B8%D0%BD_%D0%98%D0%BE%D1%81%D0%B8%D1%84_%D0%9F%D0%B0%D0%B2%D0%BB%D0%BE%D0%B2%D0%B8%D1%87` (Искусственный перевод строки в действительности отсутствует.)

`url:https://www.telebank.ru/web/front/login.x` работает как `url:www.telebank.ru/web/front/login.x`

Целочисленные атрибуты

Значение атрибута распознается как целое десятичное число.

Поддерживается также интервальный поиск: `cat:100004..1000010`

`cat:100004`, `size:200`

Запрос `size:0xFFFF` будет обработан как запрос `size 0xFFFF` из двух слов.

Логические атрибуты

В качестве значений атрибута могут использоваться:

1/0, *true/false*, *yes/no*, *on/off*

`new:yes`, `cyrillic:false`

Конфигурирование

Для каждой коллекции документов в конфигурационном файле может быть задан список зон и атрибутов. Секция Collection может содержать одну подсекцию **QueryLanguage**.

В секции **QueryLanguage** указываются зоны и атрибуты в формате *имя: тип*.

Имя может состоять из латинских букв в нижнем регистре и цифр, а также может содержать подчеркивания. Имя может начинаться только с буквы. Имена атрибутов не проверяются при загрузке конфигурации.

При неверном указании типа будет выдано сообщение об ошибке и данный атрибут будет пропущен. Поддерживаемые типы: *ZONE*, *ATTR_LITERAL*, *ATTR_INTEGER*, *ATTR_URL*, *ATTR_DATE*, *ATTR_BOOLEAN*. Названия типов могут быть заданы в любом регистре.

```
<Collection id="work">
  IndexDir /yandex/db/workindex/

  <QueryLanguage>
    title    : ZONE
    address  : ZONE
    anchor   : ZONE
    mime     : ATTR\_LITERAL
    cat      : ATTR\_INTEGER
    url      : ATTR\_URL
    date     : ATTR\_DATE
    new      : ATTR\_BOOLEAN
    cyrillic : ATTR\_BOOLEAN
  </QueryLanguage>

</Collection>
```

Зоны и атрибуты по умолчанию

Ниже приведены имена зон и атрибутов, которые будут использованы по умолчанию, в случае отсутствия конфигурационного файла. Эти имена и типы используются на веб-поиске Яндекса.

- **Зоны**: title, address, anchor, anchorint, anchormus, quote, del, ins
- **Литеральные**: charset, language, robots, style, profile, script, image, applet, object, action, domain, host, hostip, lang, mime, rhost, upath, whois
- **URL**: base, refresh, link, linkint, linkmus, url
- **Даты**: date, idate
- **Логические**: cyrillic, new
- **Целочисленные**: cat, size

Сравнение с синтаксисом версии 3.10.8

Версия 3.10.8	**Новая версия**
a|b&&c	a | b && c
(!a | !A) /+1 !b	"a * b"
title[слово]	title:слово
title[слово]$$$	title–>слово
keyword=(слово)	не поддерживается
date="20080101"	date:20080101
[[date="20060514",,60,,120]]	date:20060514 inpos:60..120
size<="1024"	size:<=1024
anchor#link="www.site.ru/*" [мебель]	anchor:(link:www.site.ru/* мягкая мебель)

SDK внешних модулей индексатора

Структуры данных, используемые в индексаторе

- Источник данных.
- Конфигурация индексатора.
- off-page информация о документе.
- Свойства документа.

Конфигурация индексатора, описанная ниже, инициализируется индексатором в начале своей работы на основании конфигурационного файла индексатора, описанного в разделе Директивы конфигурационного файла индексатора и передается в парсеры и источники данных (см. DATASRC_LIB::OpenIndexingSession).

Структуры, используемые индексатором, описаны в файле yandind.h.

Источник данных

Каждый источник данных в индексаторе описывается следующей структурой, которая соответствует одной секции DataSrc в конфигурационном файле индексатора.

```
typedef
struct DATASRC {
    const char *DsId;
    const char *Module;
    const char *Symbol;
    const char *Config;
} DATASRC;
```

Член класса	Описание
**DsId*	Идентификатор источника данных, соответствует директиве Name конфигурационного файла.
**Module*	Путь к разделяемой библиотеке, экспортирующей символ DATASRC_LIB, или NULL в случае, если источник данных является частью выполняемого модуля. Соответствует директиве Module конфигурационного файла.
**Symbol*	Имя символа типа DATASRC_LIB, который надо загрузить из разделяемой библиотеки, или адрес структуры DATASRC_LIB в выполняемом модуле. Соответствует директиве Symbol конфигурационного файла.
**Config*	Строка, передаваемая функциям OpenIndexingSession и OpenSearchSession во втором аргументе. Соответствует директиве Config конфигурационного файла.

Конфигурация индексатора

Опции индексирования, общие для всех документов, задаются в поле *IndexingFlag* структуры *INDEX_CONFIG*, соответствующим директиве GlobalOptions конфигурационного файла индексатора.

Во флажке могут быть установлены следующие биты.

```
typedef
enum INDEXING_FLAG {
  INDFL_DEFAULT           = 0,   /* StoreArchive IgnoreWordFreqs DiscardStopWords
StoreIndexingDate Update */
  INDFL_DISCARDARCHIVE      = 1,   /* см. DiscardArchive */
  INDFL_ANALYSEWORDFREQS    = 4,   /* см. AnalyseWordFreqs */
  INDFL_STORESTOPWORDS      = 8,   /* см. StoreStopWords */
  INDFL_DISCARDINDEXINGDATE = 16,  /* см. DiscardIndexingDate */
  INDFL_REINDEX             = 32,  /* см. Reindex */
  INDFL_ANALYSESEGMENTS     = 128, /* не предназначено для использования, не
документировано */
  INDFL_ONCEONLYINTID       = 256, /* не предназначено для использования, не
документировано */
} INDEXING_FLAG;
```

Индексатор, парсеры и источники данных вносят информацию о своей работе в лог с помощью функции, заданной в поле YxLogNotify структуры *INDEX_CONFIG*, и определенной как

```
typedef void (*YX_LOGNOTIFY)(YX_LOGOBJ Obj, LOG_LEVEL level, const char*
format, ...);
```

Уровень подробности сообщений задается перечислением *LOG_LEVEL*, определенным в файле yandind.h, и соответствующим директиве Level секции IndexLog конфигурационного файла.

Объект лога, передаваемый в первый параметр функции, задается в поле LogObj структуры *INDEX_CONFIG*. В данной версии этот объект создается индексатором на основании аргумента директивы FileName секции IndexLog конфигурационного файла.

Конфигурация индексатора определяется в следующей структуре.

```
typedef
struct INDEX_CONFIG {
  ui32  PortionDocCount;     /* см. директиву StoreStopWords */
  ui32  PortionMaxMemory;    /* max memory used by temporary indexer portion,
about 30 kb per doc by default */
  ui32       IndexingFlag;   /* см. описание выше*/
  const char *StopWordFile;  /* see the description of Yandex.Server config
directive 'StopWordFile' */
  const char *DocProperties; /* см. директиву DocProperty */
  const char *Groups;        /* см. директиву Groups */
  const char *newindexdir;   /* часть после последнего слеша является префиксом
файла, а не каталогом */
  const char *oldindexdir;   /* если присутствует, дескрипторы новых документов
берутся из свободных */
  const char *tempdir;       /* по умолчанию, каталог из newindexdir */
  YX_LOGNOTIFY YxLogNotify;  /* см. описание выше*/
  YX_LOGOBJ LogObj;
  DOCFORMAT *formats;        /* полный список поддерживаемых форматов см. в
разделе Парсеры (анализаторы содержимого документа)*/
  size_t formatSize;
} INDEX_CONFIG;
```

off-page информация о документе

Каждый документ может иметь документные атрибуты, как описано в разделе Документы, зоны и атрибуты. Значения атрибутов могут извлекаться из текста документа в соответствии с конфигурацией парсера документного формата (см., например, Конфигурация HTML-парсера), или задаваться источником данных в функции *OpenDoc* независимо от текста документа. Во втором случае каждому атрибуту соответствует следующая структура.

```
typedef
struct EXTATTR {
  const char *Name;
  const char *Value;
  ui32        Type;
} EXTATTR;
```

Член класса	Описание
**Name*	Название атрибута, используется в языке запросов, параметрах поисковой формы how, g и fa или в аргументе функции DocProperty.
**Value*	Значение атрибута, может иметь разный смысл в зависимости от значения параметра *Type*.
Type	Маска, задающая тип атрибута. Может быть комбинацией описанных ниже битовых полей, определяющих тип и подтип атрибута.

Атрибут может иметь один или несколько из перечисленных ниже основных типов.

Тип	Описание
EXTATTR_SEARCH	Поисковый атрибут для ограничения результатов поиска с помощью языка запросов. Может быть одного из подтипов *ATTR_TEXT*, *ATTR_LITERAL*, *ATTR_URL* или *ATTR_DATE* в соответствии с описанием в разделе Типы поисковых атрибутов.
EXTATTR_GROUP	Группировочный атрибут для использования при сортировке и группировке найденных документов. В зависимости от способа задания значения имеются следующие подтипы: GRATTR_CATEG_NAME, GRATTR_CATEG_INT, GRATTR_CATEG_INTBIN.
EXTATTR_PROPERTY	Архивное свойство документа, которое можно извлечь на странице результатов поиска с помощью функции DocProperty. Не имеет подтипов.
EXTATTR_AUX	Вспомогательное свойство документа для тонкой настройки индексатора. В настоящее время имеется только один подтип *AUXATTR_PARSER_PROPERTY*. Атрибуты такого типа устанавливаются как свойства парсера до начала чтения документа.

Атрибуты документа можно запаковать в маску Type и распаковать обратно с помощью следующих функций:

- **ui32 YxEaPack(ui32 *at*, GRATTR_TYPE *gt* = GRATTR_CATEG_NAME, ATTR_TYPE *st* = ATTR_TEXT)**.
- **ATTR_TYPE YxEaAttrType(ui32 *type*)**.
- **GRATTR_TYPE YxEaGrattrType(ui32 *type*)**.
- **ui32 YxEaExtType(ui32 *type*)**.

Подтипы группировочного атрибута определяют формат его значения в структуре *EXTATTR* следующим образом (перечислены в порядке снижения требований к вычислительным ресурсам).

Подтип	Описание
GRATTR_CATEG_NAME	Значение "EXTATTR::Value" указывает на литеральную строку, в которой записана строка-идентификатор числового значения атрибута. В этом случае при построении базы группировочных атрибутов будет автоматически создан файл имя_атрибута.c2n. **Внимание!** В данной версии при использовании этого значения не поддерживается переиндексирование.
GRATTR_CATEG_INT	Значение "EXTATTR::Value" указывает на литеральную строку, в которой записано числовое значение атрибута.
GRATTR_CATEG_INTBIN	Значение "EXTATTR::Value" указывает на область памяти размером 4 байта, в которой находится целое число, значение атрибута.

В настоящее время индексатор использует следующие свойства парсера, которые могут быть установлены с помощью атрибута типа EXTATTR_AUX.

Свойство	Описание
PP_BASE	Базовый адрес для относительных HTTP-ссылок.
PP_CHARSET	Эквивалентно заданию *DOCINPUT::Charset*.
PP_DEFCHARSET	Значение кодировки документа, которая будет использована, если распознать кодировку не удалось.
PP_ATTRCHARSET	Кодировка текстовых значений атрибутов документа, указанных в *DOCINPUT::ExtAttrs*. Если свойство не установлено, будет использована указанная или распознанная кодировка документа.
PP_LOCALFILE	Документ доступен как локальный файл с указанным именем. Оптимизация для форматов, требующих для своего анализа произвольного доступа.
PP_IGNORE_PUNCT_BREAKS	Игнорирование разбиения на предложения и параграфы от внутреннего лексического анализатора. Будет учитываться только внутренняя разметка документа.
PP_IGNORE_PASSAGES	Текст данного документа не будет архивироваться.

Свойства документа

```
typedef void *YX_DOCOBJ;

typedef
struct DOCINPUT {
  /* obligatory data */
  YX_DOCOBJ ReaderObj;  /* object with document content  */
  const char *UrlBase;  /* уникальный внешний идентификатор документа, */
                        /* источник данных однозначно определяет документ по этому
идентификатору. */
                        /* одновременно этот идентификатор является поисковым
атрибутом "#url" */
                        /* длина идентификатора не должна превышать 249 символов
*/
  const char *MimeType; /* document format identifier */
                        /* должен быть одним из идентификаторов, указанных в
```

```
INDEX_CONFIG::formats */
  /* optional data */
  const char *Charset;  /* encoding, will be recognized if not defined; */
  const char *UrlProp;  /* свойство "адрес документа" (DP_URL), видимое в
результатах поиска */
  time_t ModTime;
  /* off-page attributes */
  EXTATTR *ExtAttrs;
  size_t ExtAttrSize;
} DOCINPUT;
```

Модуль источника данных

- API источника данных.
- Работа источника данных при индексировании.
- Работа источника данных при поиске.
- Функции структуры DATASRC_LIB.

API источника данных

API источника данных содержится в файле yandds.h.

Пример реализации источника данных находится в каталоге sample/flist.

Интерфейс каждого источника данных заключен в структуру *DATASRC_LIB*, состоящую из 6 указателей на функции:

```
typedef
struct DATASRC_LIB {
    FUNC_DataSrc_OpenIndexingSession OpenIndexingSession;
    FUNC_DataSrc_OpenSearchSession   OpenSearchSession;
    FUNC_DataSrc_OpenDoc             OpenDoc;
    FUNC_DataSrc_ReadDocumentBytes   ReadDocumentBytes;
    FUNC_DataSrc_CloseDoc            CloseDoc;
    FUNC_DataSrc_CloseSession        CloseSession;
    FUNC_DataSrc_GetProperty         GetProperty;
} DATASRC_LIB;
```

Каждая из функций возвращает целое число, одно из следующих.

```
typedef enum YDS_STATUS {
    YDS_ERROR = -2,
    YDS_EOF   = -1,
    YDS_OK    =  0
} YDS_STATUS;
```

Каждая из функций в качестве первого аргумента принимает дескриптор объекта, инкапсулирующего источник данных и определенного как

```
typedef void *DATAOBJ;
```

и должна быть потоково безопасна.

Работа источника данных при индексировании

В начале каждого сеанса индексирования индексатор загружает структуру DATASRC_LIB из указанного в конфигурации места. Затем один раз вызывается OpenIndexingSession. Если OpenIndexingSession возвращает *YDS_OK*, происходит индексирование, в конце которого один раз вызывается CloseSession(). В противном случае индексирования и каких-либо последующих вызовов не происходит.

В течение индексирования индексатор последовательно вызывает OpenDoc()/ReadDocumentBytes()/CloseDoc(). Индексирование прекращается в случае, если OpenDoc() возвращает *YDS_EOF*. Если OpenDoc() вернула *YDS_OK*, в дальнейшем последует вызов CloseDoc(). Когда OpenDoc() возвращает

YDS_OK, она всегда должна установить поле *DOCINPUT::UrlBase*. Если при этом установлено поле *DOCINPUT::ReaderObj*, документ будет проиндексирован в первый раз или переиндексирован, в процессе этого между вызовами OpenDoc() и CloseDoc() будет один или несколько раз вызвана функция ReadDocumentBytes() для считывания содержимого документа. Если же *DOCINPUT::ReaderObj* нулевой, документ с идентификатором *DOCINPUT::UrlBase* будет удален из индекса, а вызовов ReadDocumentBytes() не будет.

Политика переиндексирования полностью определяется источником данных. Источник данных сам определяет, как формируются идентификаторы документов и где они хранятся, если их вообще нужно хранить между сеансами индексирования. Для каждого нового или изменившегося документа источник возвращает валидные *DOCINPUT::UrlBase* и *DOCINPUT::ReaderObj*. Источник может также установить и другие поля структуры *DOCINPUT*.

Работа источника данных при поиске

Во время поиска источник данных используется только для подсветки найденных слов в полном документе формата text/plain и text/html и для поиска похожих документов. Если такая функциональность не нужна, OpenSearchSession() должна просто возвращать *YDS_EOF*.

В начале каждого сеанса поиска загружается структура DATASRC_LIB из указанного в конфигурации места. Функция OpenSearchSession() вызывается один раз в каждом поисковом потоке, при первом запросе на подсветку или поиск похожего. Таким образом, во время поиска OpenSearchSession() может вызываться несколько раз из разных потоков. Если OpenSearchSession() возвращает *YDS_OK*, источник данных используется во время поиска, в конце которого один раз вызывается CloseSession().

Если поисковая сессия успешно открыта, при запросе на подсветку или поиск похожего документа будут вызываться функции OpenDoc()/ReadDocumentBytes()/CloseDoc(). Эта последовательность вызовов будет происходить в одном и том же поисковом потоке. Возможен одновременный параллельный вызов из разных потоков для разных документов.

Функции структуры DATASRC_LIB

Функции и их описание
typedef int (*FUNC_DataSrc_OpenIndexingSession)(DATAOBJ *_DataObj_, const char *_Config_, const INDEX_CONFIG *_Ic_) Возвращает *YDS_OK* или *YDS_ERROR*. *DataObj* — адрес объекта источника данных. Если открытие произошло успешно, переменной **DataObj* должен быть присвоен допустимый адрес. *Config* — строка конфигурации, указанная в директиве Config секции DataSrc конфигурации индексатора. *Ic* — конфигурационные данные индексатора. Например, INDEX_CONFIG*::tempdir* может использоваться для создания временных файлов, если надо, а INDEX_CONFIG*::newindexdir* может служить префиксом файла с данными о текущем сеансе индексирования, которые нужны во время следующего.
typedef int (*FUNC_DataSrc_OpenSearchSession)(DATAOBJ *_DataObj_, const char *_config_, const char *_indexprefix_, YX_LOGNOTIFY _YxLogNotify_, YX_LOGOBJ _LogObj_) Возвращает *YDS_OK* при успешном завершении, *YDS_ERROR* в случае ошибки и *YDS_EOF* в случае, если использовать источник данных во время поиска не нужно. *DataObj* — адрес объекта источника данных. Если открытие произошло успешно, переменной **DataObj* должен быть присвоен допустимый адрес. *Config* — строка конфигурации, указанная в директиве Config секции DataSrc конфигурации поискового сервиса. *indexprefix* — префикс индекса, по которому происходит поиск. *YxLogNotify*, *LogObj* — обеспечивают доступ к логу вызывающей программы для протоколирования или записи отладочной информации.
typedef int (*FUNC_DataSrc_OpenDoc)(DATAOBJ _DataObj_, DOCINPUT *_di_) Возвращает *YDS_OK* при успешном завершении, *YDS_ERROR* в случае ошибки и *YDS_EOF* в случае, если документов для индексирования больше нет. *DataObj* — допустимый объект источника данных, открытый в функции OpenIndexingSession или OpenSearchSession.

Функции и их описание

di — информация о документе, содержимое которого будет предоставлено во время последующего вызова ReadDocumentBytes().

Если поля структуры *di* нулевые, вызов произошел из индексатора. В этом случае надо заполнить поле *UrlBase* для удаляемого документа и, как минимум, поля *UrlBase*, *ReaderObj* и *MimeType* для документа, предназначенного для индексации. Если такой *UrlBase* уже есть в индексе, он будет переиндексирован. Если *ReaderObj* оставлен нулевым, документ будет удален из индекса. Если в поле *UrlProp* предоставлен URL с префиксом http:, он будет передан клиенту конечного пользователя поиска для открытия найденного документа.

В противном случае, во время поиска будет использован источник данных, и в структуре *di* при вызове данной функции будет заполнено поле *UrlProp*. В этом случае также необходимо заполнить поле *ReaderObj*.

typedef int (*FUNC_DataSrc_ReadDocumentBytes)(DATAOBJ *DataObj*, YX_DOCOBJ *ReaderObj*, void **toFill*, size_t *maxToRead*, size_t **Read*)

Возвращает *YDS_OK* при успешном завершении, *YDS_ERROR* в случае ошибки и *YDS_EOF* в случае, если документ полностью прочитан.

DataObj — допустимый объект источника данных, открытый в функции OpenIndexingSession или OpenSearchSession.

ReaderObj — дескриптор объекта для чтения документа, указанный источником в OpenDoc().

toFill — буфер, в который надо записать содержимое документа.

maxToRead — размер переданного буфера.

Read — адрес переменной, в которой источник должен вернуть фактическое число байт, записанных им в буфер *toFill* (и меньшее *maxToRead*).

При успешном заполнении буфера для чтения функция возвращает *YDS_OK*.

Если возвращается *YDS_EOF*, вызовы функции прекращаются, документ полностью прочитан.

Если в какой-то момент возвращается *YDS_ERROR*, документ в индекс не попадает. Фактически, источник может читать содержимое документа в момент вызова этой функции, или возвращать прочитанное ранее в OpenDoc().

typedef int (*FUNC_DataSrc_CloseDoc)(DATAOBJ *DataObj*, DOCINPUT **di*, IParser **pars*)

Возвращает *YDS_OK*, *YDS_EOF* или *YDS_ERROR*.

DataObj — допустимый объект источника данных, открытый в функции OpenIndexingSession или OpenSearchSession.

di — тот же объект, что был передан в OpenDoc().

pars — объект парсера, который был использован для интерпретации данного документа.

Освобождает ресурсы, возможно занятые в OpenDoc(). При необходимости, может вызвать *GetProperty* для получения свойств документа, определенных при индексировании, например, кодировки или списка ссылок. Свойство парсера *PP_INDEXRES* будет содержать строку "OK" или описание ошибки, возникшей при индексировании. Если данная функция возвращает *YDS_OK*, документ будет сохранен в индексе. В противном случае индекс не будет содержать документа с *di->UrlBase*. Это может быть использовано, если решение о сохранении документа в индексе нужно принять после анализа свойств документа, определенных при индексировании.

Пример реализации функции:

```
int ::CloseDoc(DATAOBJ DataObj, DOCINPUT *di, IParser* P)
{
    int ret = YDS_OK;
    if (di && di->ReaderObj != NULL) {
        // to do:
        // освобождаем ресурсы, выделенные для di->ReaderObj в функции OpenDoc()
        if (P) {
            char* prop = 0;
            ATTR_TYPE at;
            P->GetProperty(PP_INDEXRES, &prop, &at);
            if (prop && strcmp(indres, "OK") == 0) {
                // to do:
                // Записываем нужную информацию из di во внутренний лог источника данных
                // для последующего использования при переиндексировании или
                // в системе внешнего мониторинга.
                // Посылаем любые сообщения во внешний мир о том, что данный документ
проиндексирован.
            }
```

Функции и их описание

```
            if (P->GetProperty(PP_ROBOTS, &prop, &at) == 0) {
                if (prop && prop[0] == '0') // noindex
                    ret = YDS_EOF; // не включать в индекс
            }
        }
        // to do:
        // освобождаем ресурсы, выделенные для di в функции OpenDoc()
    }
    return ret;
}
```

typedef int (*FUNC_DataSrc_CloseSession)(DATAOBJ *DataObj*)

Возвращает *YDS_OK* или *YDS_ERROR*.

DataObj — допустимый объект источника данных, открытый в функции OpenIndexingSession или OpenSearchSession.

Освобождает ресурсы, занятые в объекте *DataObj*. После вызова этой функции дальнейшее использование *DataObj* недопустимо.

typedef int (*FUNC_DataSrc_GetProperty)(DATAOBJ *DataObj*, YDS_PROPERTY *PropName*, YDSPVALUE **pPropValue*)

Возвращает *YDS_OK*, если значение *pPropValue* установлено, или *YDS_EOF*, если нужно использовать поведение по умолчанию.

DataObj — допустимый объект источника данных, открытый в функции OpenIndexingSession или OpenSearchSession.

PropName — идентификатор параметра, значение которого запрашивается.

pPropValue — адрес указателя, в котором надо возвратить значение параметра.

Устанавливает значение одного из параметров, управляющих поведением индексатора.

Индексатор может запрашивать значение одного из следующих параметров:

YDSP_EXTENDID — нужно ли модифицировать идентификатор документа

YDSP_THREADMODEL — можно ли использовать источник данных в многопоточном режиме

YDSP_SAVEDFILES — список суффиксов файлов, в которых источник данных сохраняет информацию, нужную ему для следующего индексирования.

Вспомогательные утилиты

Программа printkeys

Индексные файлы

В результате индексирования каждому документу приписывается уникальное число — *внутренний идентификатор документа*, а содержимое и свойства документа запоминаются в *индексных файлах* indexkey и indexinv.

Все слова, встречающиеся в массиве индексируемых документов, а также поисковые атрибуты и зоны запоминаются в файле indexkey и в дальнейшем называются ключами.

Для каждого ключа в файле indexinv запоминается список словопозиций, или просто позиций. Каждая позиция включает внутренний идентификатор документа, номер предложения, в котором встретился ключ, номер слова в предложении, а также некоторый вес, назначенный данной позиции интерпретатором документного формата.

Служебная программа *printkeys* позволяет представить эту информацию в текстовом виде, пригодном для чтения или дальнейшей обработки программами-скриптами.

Параметры командной строки

Вызов программы *printkeys* осуществляется следующим образом:

```
printkeys [-0|1] [-k key] [-e] [-p] [-w|x|l|i|f]  [-s] [-m] [-c] [-t] [-o output]
indexkey indexinv
```

Здесь в квадратных скобках указаны необязательные параметры.

Параметр	Описание
Формат индекса	
-0	Индексный файл является порцией индекса, созданной во временном каталоге в процессе работы индексатора. В этом случае число позиций не выводится.
-1	Индексный файл является полным индексом, полученным после слияния порций и готовым для работы поискового сервера. Значение по умолчанию: *-1*.
Формат выводимых ключей	
-k keyprefix	Выводить только ключи, начинающиеся с указанной подстроки. По умолчанию выводятся все ключи.
-e	Выводить ключи в том виде, как они хранятся в индексе (все буквы строчные; если слово начинается с прописной буквы, в конце слова прибавляется символ со значением "1"). По умолчанию, ключи преобразуются к нужному регистру.
-p	Выводить смещение в файле indexinv до начала записи с упакованными позициями, соответствующими ключу.

Параметр	Описание
	По умолчанию смещение не выводится.
Формат выводимых позиций	
-w	Выводить позиции в формате "\t[D.S.W.R]\n". *D*, *S*, *W* и *R* — соответственно идентификатор документа, номер предложения, номер слова в предложении и вес позиции в десятичном формате.
-x	Выводить позиции в формате "\tP\n", где *P* — восьмибайтовое число в шестнадцатеричном формате. Идентификатор документа, номер предложения, номер слова в предложении и вес позиции занимают в этом числе определенное число бит, которое может зависеть от версии индексатора.
-l	Выводить позиции в формате "\tD\tL\n". *D* — идентификатор документа, *L* — число в десятичном формате, в котором запакованы номер предложения, номер слова в предложении и вес позиции.
-i	Выводить только внутренние идентификаторы документа в формате "\tD\n", где *D* — идентификатор документа в десятичном формате.
-f	Выводить слово позиции в формате "\t[D.S.W.R.F]\n". *D*, *S*, *W*, *R* и *F* — соответственно идентификатор документа, номер предложения, номер слова в предложении, вес позиции в десятичном формате и номер словоформы в данном ключе.
-s	Выводить статистику по словоформам.
-c	При выводе ключей показывать словоформы. Если опция отсутствует, показывать леммы.
Параметры ввода-вывода	
-o filename	Выводить информацию в filename. По умолчанию вывод осуществляется в *stdout*.
-t	Использовать символ табуляции в качестве разделителя полей.
-m	При чтении индекса использовать мапирование. По умолчанию используется последовательное чтение. Имеет смысл, только если указан один из параметров *-w*, *-i*, *-l*, *-x*.
Индекс	
indexkey	Путь к файлу с ключами. Обязательный параметр.
indexinv	Путь к файлу с позициями. Обязательный параметр.

Программа tarcview

Служебная программа *tarcview* предназначена для просмотра текста и свойств документа, сохраненных во время индексирования. Она принимает путь к архиву документов в качестве параметра командной

строки и выводит содержимое документов на стандартный вывод. Документный архив хранится в файлах файлах indexarc и indexdir, расположенных в индексном каталоге.

Параметры командной строки

Вызов программы tarcview осуществляется следующим образом:

```
tarcview [-ec] [-o output] archivepath [DocN]
```

Здесь в квадратных скобках указаны необязательные параметры.

Параметр	Описание
-e	Выводит текст и свойства документа, полученные при индексации.
-c	Выполняет проверку целостности индекса, ничего не выводит.
-m	Выводит документы вместе с зонами *print marks of markup zones*.
-w	Выводит документы и зоны, для которых определен вес.
-h	Выводит HTML-архив документов.
-o	Позволяет сохранять информацию в файл. *Output* – название файла для вывода информации.
archivename	Путь к файлу, содержащему документный архив, без окончания "arc". Например, если индексные файлы находятся в каталоге /yandex/workindex/, нужно указать /yandex/workindex/index.
DocN	Идентификатор документа, содержание которого надо вывести.

Программа atrview

Служебная программа *atrview* предназначена для преобразования базы группировочных атрибутов в набор текстовых файлов. Для каждого группировочного атрибута программа создает файл имя_атрибута.d2c. Каждая строка этого файла задает одно соответствие между внутренним идентификатором документа, назначенным индексатором, и целочисленным значением группировочного атрибута. Она имеет следующий формат:

```
(целое число, идентификатор документа)(символ табуляции)(целое число, значение атрибута)(символ перевода строки)
```

Параметры командной строки

Вызов программы atrview осуществляется следующим образом:

```
atrview [-a attrname] atrpath
```

Здесь в квадратных скобках указаны необязательные параметры.

Параметр	Описание
-a	Указывает, что нужно создать только один выходной файл, соответствующий данному имени атрибута. По умолчанию создается несколько файлов, по одному на каждый атрибут, имеющийся в базе атрибутов.

Параметр	Описание
atrpath	Путь к файлу, содержащему базу группировочных атрибутов, без окончания "atr". Например, если индексные файлы находятся в каталоге /yandex/workindex/, нужно указать /yandex/workindex/index.

Программа savequery

Служебная программа *savequery* предназначена для выполнения запроса на языке запросов Яндекса с сохранением результатов в заданный файл. Формат файла с сохраненными результатами такой же, как у временных файлов, создаваемых поисковым сервером в случае, если в его конфигурационном файле задана секция QueryCache.

Программа *savequery* может быть использована в постиндексирующей процедуре для создания предкомпилированных запросов, идентификаторы которых можно добавлять в пользовательские запросы с помощью оператора языка запросов *$* с целью применения фильтра. Файл сохраненного запроса, созданный с помощью *savequery*, может быть также использован как входные данные для программы hidedocs.

Параметры командной строки

Вызов программы *savequery* осуществляется следующим образом:

```
savequery -y indexprefix -q query -o hits [-r]
```

Здесь в квадратных скобках указаны необязательные параметры.

Параметр	Описание
-y indexprefix	Путь к файлам, содержащим индекс, по которым будет осуществлен поиск, без окончаний "inv" и "key". Например, если индексные файлы находятся в каталоге /yandex/workindex/, нужно указать /yandex/workindex/index.
-q query	Путь к файлу, содержащему запрос на языке запросов Яндекса. Файл может содержать переводы строки для удобства написания длинных запросов. В конце файла должен присутствовать один перевод строки.
-o hits	Путь к файлу, в который будет записан сохраненный запрос. Если имя файла начинается с символа "Z" и файл расположен в подчиненном каталоге hits индексного каталого (указанного в директиве IndexDir), то это имя можно использовать в операторе "$" поиска в найденном.
-r	Учитывать в сохраненном запросе текстовую релевантность найденных документов. По умолчанию сохраняются только внутренние идентификаторы документов.

Пример "семейного фильтра"

Допустим индекс расположен в каталоге /yandex/workindex.

Создадим каталог /yandex/workindex/hits и файл badwords следующего содержания:

```
"жрица любви" |
playboy
```

Создадим сохраненный запрос в файле Zexclude с помощью вызова:

```
savequery -y /yandex/workindex/index -q badwords -o /yandex/workindex/hits/
Zexclude
```

Теперь, если у пользователя тем или иным способом установлен "семейный фильтр", мы можем в функции UserRequest модифицировать запрос пользователя

```
userquery
```

к виду

```
(userquery) ~~$$Zexclude
```

с тем, чтобы исключить из результатов поиска документы с содержанием "только для взрослых".

Программа hidedocs

Служебная программа *hidedocs* предназначена для оперативного удаления из результатов поиска указанных документов. При этом информация о документах остается в индексных файлах и может быть окончательно удалена при последующем переиндексировании.

Программа *hidedocs* может быть полезна в условиях, когда индексные файлы очень большие, так что слияние результатов после переиндексирования занимает много времени. Программа помечает указанные документы как недоступные и может работать без остановки поискового процесса.

Параметры командной строки

Вызов программы *hidedocs* осуществляется следующим образом:

```
hidedocs [-n] hits indexprefix
```

Здесь в квадратных скобках указаны необязательные параметры.

Параметр	Описание
indexprefix	Каталог с индексными файлами, документы в которых надо пометить, дополненный префиксом имени файла. Например, если индексные файлы находятся в каталоге /yandex/workindex/, нужно указать /yandex/workindex/index.
hits	Путь к файлу с внутренними идентификаторами помечаемых документов. По умолчанию, этот файл имеет формат сохраненного запроса Яндекса, и может быть создан, например, программой savequery.
-n	Указывает, что файл hits с идентификаторами документов имеет текстовый формат. В этом случае файл hits должен содержать внутренние идентификаторы документов в десятичном формате, по одному на строку. Файл такого формата может быть получен с помощью вызова: `printkeys -w -k#url= indexkey indexinv` с последующим парсированием результата, например, с помощью такого скрипта на PERL:

Параметр	Описание

	`#!/usr/bin/perl`
	`use strict;`
	`while (<STDIN>) {`
	`  if (/^#url="(.+)\s\d+\s\d+/) {`
	`  # $1 содержит URL документа`
	`  } elsif (/\[(\d+)\.\d+\.\d+\.\d+\]/) {`
	`  print $1."\n";`
	`  }`
	`}`

Пример использования hidedoc

Допустим документы в индексе, расположенном в каталоге /yandex/workindex, имеют литеральный атрибут *myattr*, и имеется файл query следующего содержания:

```
#myattr="value1" |
#myattr="value2" |
#myattr="value3"
```

Тогда последовательность вызовов

```
savequery -y /yandex/workindex/index -q query -o /temp/hit
hidedocs /temp/hit /yandex/workindex/index
```

делает недоступными в результатах поиска все документы, имеющие атрибут *myattr* со значениями "value1", "value2" или "value3".

Релизы

В разделе содержится информация о выпущенных релизах документа. Для каждого релиза указан номер, дата выпуска, краткое содержание изменений и дополнений, вошедших в релиз.

Релиз 2010.09, сентябрь 2010 г.

- Новая функциональность Ссылочное ранжирование
  Добавлено описание аргумента StoreLinkIndex директивы GlobalOptions, включающего построение ссылочного индекса.
- Новая функциональность Навигационный источник.
  Добавлено описание директивы NavigationSource, определящий путь к файлу навигационного источника.

  Добавлен пример файла, содержащего навигационный источник.
- Обновлен список поддерживаемых медиа-типов (см. таблицу Медиа-типы документов)
- Прекращена поддержка поисковых запросов в кодировке, отличной от UTF-8.
  Удалено описание директивы **QueryCharset**, определяющей кодировку поисковых запросов.
- Прекращена поддержка внешних парсеров.
  Изменены и дополнены разделы:
  - Конфигурационный файл индексатора — Секция DocFormat.
  - Парсеры (анализаторы содержимого документа).
  - Конфигурация HTML-парсера.
  - Конфигурация XML-парсера.
  - Структуры данных, используемые в индексаторе.

  Удален раздел **Модуль парсера**.

Релиз 2009.05, сентябрь 2009 г.

Документ содержит справочную информацию о продукте Яндекс.Сервер, его структуре, установке и настройке компонент продукта и вспомогательных утилит.

Яндекс

Яндекс.Сервер 2010.09

Руководство по установке и эксплуатации

6.09.2010