**Основные положения**

Библиотека JasperReports интегрируется в ядро системы с целью выполнения следующих задач:

1. Формирование онлайн отчетов.
2. Формирование офлайн отчетов.
3. Формирование периодических отчетов.

### Способ интеграции

Библиотеки JaserReports включают в состав итогового ear системы. Создается EJB ReportService для управления подсистемой отчетов. Отчеты могут формироваться в 3 режимах:

1. Онлайн – формирование отчета начинается непосредственно после вызова соответствующей функции EJB. Отчет возвращается как результат выполнения данной функции. Возвращаемое значение функции класс ReportResult, одним из полей которого сам отчет как массив байт.
2. Офлайн – формирование отчета асинхронно. Запуск формирования производится соответствующей функцией EJB. Для запуска используется технология Asynchronous EJB call. После окончания формирования отчета формируется уведомление в системе уведомлений. В процессе формирования отчета должна быть возможность сформировать на вызывающей стороне progressBar о процессе формирования.
3. Периодические отчеты. В системе создается специальный тип доменных объектов – scheduled\_task описывающий запуск задания по расписанию. Каждое из таких заданий имеет параметр тип задания, и соответствующее этому типу заданий набор параметров. Одним из таких заданий является задача формирования отчета. Как параметры к такому заданию является имя отчета и набор параметров отчета. Параметры могут быть как жестко заданные так и вычисляемые (пример вычисляемого параметра отчета – период за который формируется отчет, подразделение пользователя, сформировавшего задание на отчет) Периодические отчеты формируют сами пользователи, то есть в интерфейсе должен быть блок, позволяющий сформировать параметры периодических отчетов. Периодические отчеты формируются системой и отправляют уведомления о формирование по электронной почте. Если пользователь в данный момент подключен в системе на экране отображается уведомление о сформированном отчете. Описание подсистемы периодических заданий выходит за рамки данного документа.

Идентификацией отчета в системе является его имя. Оно уникально. Информация об отчете хранится в специальных доменных объектах report\_template. В report\_template хранится информация об имени отчета и привязаны вложения с шаблонами отчета и xml файлом описания отчета. Шаблонов может быть много (в случае с подотчетами, но только одно из вложений может быть основным шаблоном отчета). В xml файле описания отчета хранится метаинформация о параметрах отчета, позволяющая отобразить на интерфейсе форму ввода параметров отчета. В качестве формы ввода параметров отчета указывается форма из сконфигурированных форм основной конфигурации ядра, плюс имеется конфигурация для начальной инициализации полей форм. Так же в файле описания отчета хранится имя основного шаблона отчета, имя класса источника данных, доступные форматы, в которых формируется отчет, флаг времени хранения отчета, флаг возможности изменить время хранения отчета по умолчанию.

## JDBC драйвер.

Доступ к данным осуществляется с помощью JDBC драйвера, специально разработанного для платформы. Данный драйвер обращается к данным через ядро системы с учетом прав доступа. Драйвер имеет 2 режима работы с помощью локального интерфейса EJB и с помощью удаленного интерфейса EJB. Выбор способа работы драйвера определяется разработчиком использующего данный драйвер и указывается в строке подключения драйвера. В случае с работой через Remote интерфейсы так же в строке указывается имя сервера, логин и пароль. Драйвер разрабатывается таким образом, чтобы через него можно было бы работать с редактором шаблонов отчета iReport. Так же драйвер может использоваться для работы внешних систем построения отчетов, например CrystalReport Server или JasperReport Server. Драйвер должен работать с сервисом коллекций с учетом страничной подкачки данных, для исключения ошибки типа OutOfMemoryException.

## Описание сущьностей

Тип доменного объекта report\_template – Шаблон отчета.

|  |  |
| --- | --- |
| Поле | Описание |
| name | Строка. Имя отчета. Уникально в пределах системы. Используется для однозначной идентификации шаблона отчета в системе. |
| description | Строка. Описание отчета. Используется интерфейсом для формирования элементов управления, запускающих формирование отчета |

К экземпляру данного типа прикрепляются вложения: шаблоны отчета, шаблоны подотчетов, xml файл с метаинформацией, класс скриптлета (не обязательно), класс источника данных (не обязательно).

Тип доменного объекта report\_result – Сформированный отчет

|  |  |
| --- | --- |
| Поле | Описание |
| name | Имя отчета. Формируется исходя из имени шаблона отчета и времени формирования отчета |
| template\_id | Reference на report\_template. Идентификатор шаблона отчета. |
| owner | Reference на Person. Идентификатор персоны сформировавшей отчет или сформировавший задание на отчет. |
| keep\_to | Дата окончания хранения отчета. После этой даты отчет может быть удален специальным санитарным джобом. |

К экземпляру данного объекта прикрепляется сформированный отчет и файл xml с параметрами отчета.

Интерфейс EJB ReportService.

|  |  |  |
| --- | --- | --- |
| Метод | Параметр | Описание |
| generate | Имя шаблона отчета, формат отчета, параметры отчета | Производится онлайн формирование отчета. Результатом работы функции является экземпляр класса ReportResult. |
| generateAsync | Имя шаблона отчета, формат отчета, параметры отчета, права на сформированный отчет. | Офлайн (асинхронное) формирование отчета. Функция возвращает управление немедленно. Затем в асинхронном режиме формирует отчет и отправляет результат в подсистему уведомлений (подсистема уведомлений выходит за рамки данного документа. В простейшем случае подсистема уведомлений отправляет email с отчетом пользователю запустившем формирование задания) Асинхронный вызов должен поддерживать механизм отслеживания статуса формирования отчета, для формирования progressBar на клиентских местах. |

Интерфейс ReportServiceAdmin

|  |  |  |
| --- | --- | --- |
| Метод | Параметр | Описание |
| deploy | Передается структура описывающая шаблон отчета. В ту структуру входят имя отчета, все шаблоны отчетов в формате jrxml и xml файл с метоинформацией отчета, файл класса скриптлета, файл класса источника данных. | При вызове метода происходит компиляция шаблонов и классов скриптлета и источника данных, и формирование или обновление доменного объекта report\_template. В случае неудачи компиляции происходит формирование Exception. |
| undeploy | Имя отчета | Удаляется информация о шаблоне отчета. Удаляется соответствующий report\_template и прикрепленные к нему вложения. |

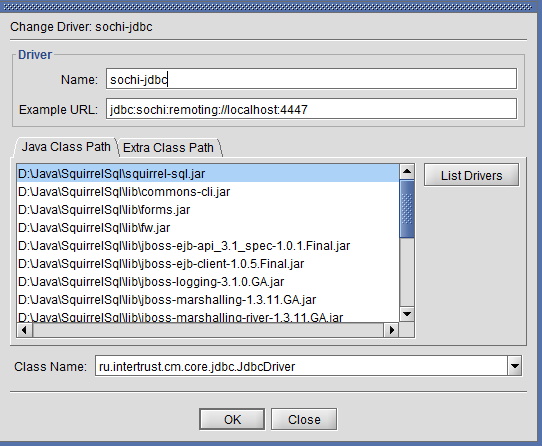
XML файл метаинформации.

|  |  |  |
| --- | --- | --- |
| XML элемент | Тип | Описание |
| Template | тэг | Описание шаблона отчета |
| name | атрибут | Имя шаблона |
| description | атрибут | Описание отчета |
| template | атрибут | Имя файла основного шаблона отчета без расширения |
| dataSourceClass | Атрибут | Имя класса источника данных (не обязательный) |
| Parametrs | Тэг | Паракметры отчета |
| Parameter | Тэг | В данном документе указан условно. Может быть StringParameter, NumberParameter, DateParameter, IntervalParameter, ListParameter и т.д. Содержит одинаковые для всех параметров атрибуты так и специфичные для параметра определенного типа (например источник записей для списка) |
| name | Атрибут | Имя параметра. Совпадает с именем параметра в шаблоне отчета именем источника данных для виджета на форме ввода параметров отчета. |
| settings | Тэг | Настройки конкретного элемента, здесь указан условно. Данный тэг зависит от виджета на форме ввода параметров отчета. Должен формировать правила заполнения виджета по умолчанию. |

## Настройка подключения программ используя JDBC драйвер

### SQuirreL SQL Client

1. Скопировать следующие библиотеки в директорию SquirrelSql\lib:
   1. service-api-1.0-SNAPSHOT.jar
   2. model-1.0-SNAPSHOT.jar
   3. xnio-nio-3.0.7.GA.jar
   4. jboss-remoting-3.2.17.GA.jar
   5. jboss-remote-naming-1.0.2.Final.jar
   6. xnio-api-3.0.7.GA.jar
   7. jboss-ejb-client-1.0.5.Final.jar
   8. jboss-ejb-api\_3.1\_spec-1.0.1.Final.jar
   9. jboss-transaction-api\_1.1\_spec-1.0.0.Final.jar
   10. jboss-logging-3.1.0.GA.jar
   11. jboss-marshalling-1.3.11.GA.jar
   12. jboss-marshalling-river-1.3.11.GA.jar
   13. log4j-1.2.16.jar
   14. jdbc-1.0-SNAPSHOT.jar
2. Удалить имеющийся в директории SquirrelSql\lib файл log4j.jar
3. Запустить приложение
4. В окне доступных драйверов Drivers создать драйвер sochi-jdbc как показано ка картинке

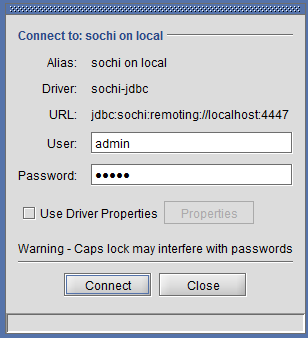


В поле Name ввести произвольное имя

В поле Example URL ввести jdbc:sochi:remoting://localhost:4447. Вместо localhost должно быть корректное имя сервера, вместо 4447 нужно указать корректный порт для remoting соединений.

В поле Class Name ввести имя класса драйвера ru.intertrust.cm.core.jdbc.JdbcDriver

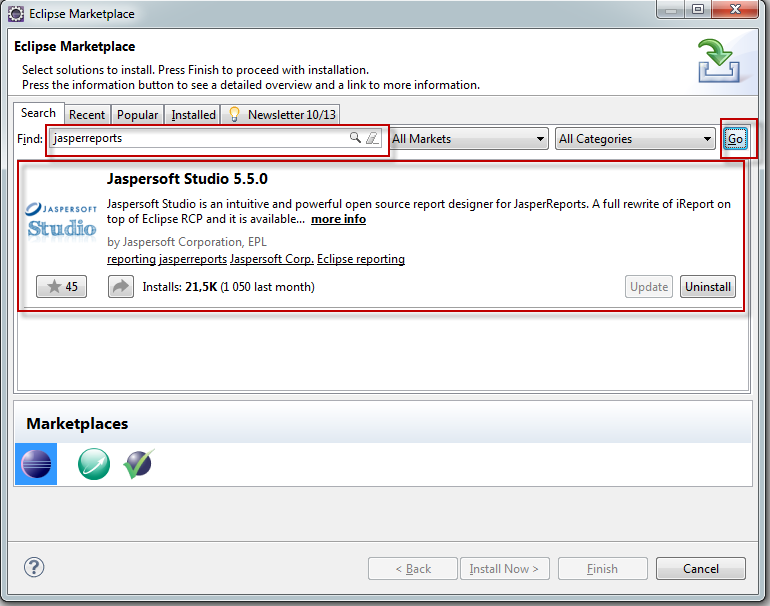
1. В окне подключений Aliases необходимо создать новое подключение как показано на рисунке



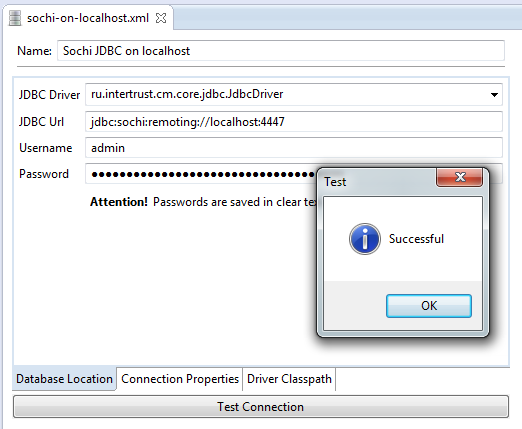
1. Нажать кнопку Connect, произойдет подключение и отобразится поле для ввода запроса. В данном поле можно писать запрос на внутреннем DSQL языке и получать результат.

### JasperReport Studio

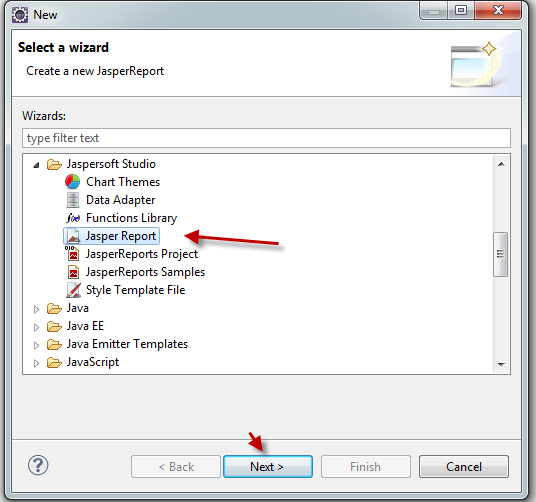
1. Установить плагин JasperReport Studio используя Eclipse Marketplace



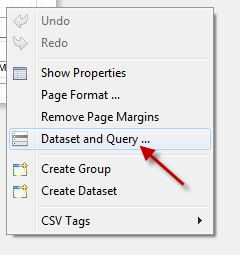
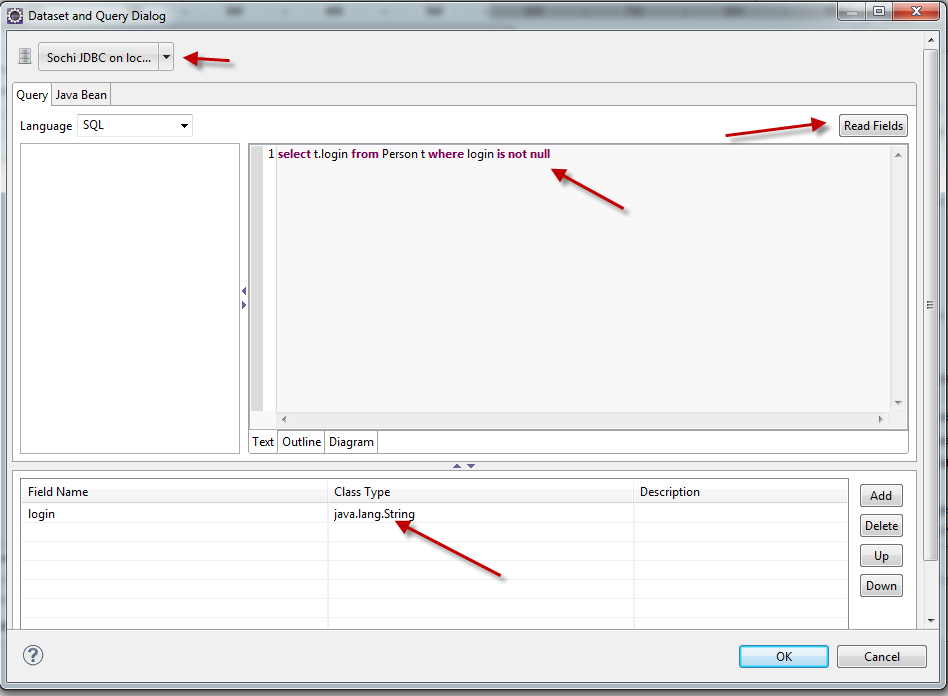
1. Создать workspace директорию отдельно от директорий проекта. Важно wokspace должен быть отдельным, и не являтся workspace проекта sochi.
2. Импортировать maven проект CM5\Client\reports
3. Отредактировать файл sochi-on-localhost.xml, указать корректные логин пароль и имя сервера.
4. Проверить соединение нажав кнопку test на форме проверки соединения. Тест должен пройти.



1. Создать новый шаблон отчета. Для каждого шаблона отчета нужна своя отдельная директория.



1. В шаблоне отчета из контекстного меню на свободном месте отчета вызываем конфигуратор источника данных. В нем указываем соединение и пишем запрос. Далее нажимаем кнопку Read Fields. Должны сформироваться поля.

1. Далее форматируем отчет и наполняем его содержимым.