JDBC

**JDBC** - расшифровывается как Java DataBase Connectivity. Говоря дословно – соединения с «хранилищами электронных структурированных материалов» в Джаве. Это – платформенно-независимый стандарт. Используется для того, чтобы обеспечивать взаимодействие Java-контента со всевозможными СУБД.

java.sql

java.sql – это библиотека с реализацией jdbc.

Интерфейсы:

**Array** - отображение на языке программирования Java для типа SQL ARRAY.

**Blob** - представление (отображение) на языке программирования Java™ SQL- BLOB значение.

**CallableStatement** - интерфейс, используемый для выполнения хранимых процедур SQL.

**Clob** - отображение на языке программирования Java™ для SQL CLOB Тип.

**Connection** - соединение (сеанс) с определенной базой данных.

**DatabaseMetaData** - исчерпывающая информация о базе данных в целом.

**Driver** - интерфейс, который должен реализовывать каждый класс драйвера.

**DriverAction** - интерфейс, который должен быть реализован при Драйвер хочет получать уведомления от DriverManager.

**NClob** - отображение на языке программирования Java™ для типа SQL NCLOB.

**ParameterMetaData** - объект, который может использоваться для получения информации о типах и свойствах каждого параметра-маркера в PreparedStatement объект.

**PreparedStatement** - объект, представляющий предварительно скомпилированную инструкцию SQL.

**Ref** - отображение на языке программирования Java SQL- REF значение, которое является ссылкой на значение структурированного типа SQL в базе данных.

**ResultSet** - таблица данных, представляющая результирующий набор базы данных, который обычно генерируется путем выполнения инструкции, которая запрашивает базу данных.

**ResultSetMetaData** - объект, который может использоваться для получения информации о типах и свойствах столбцов в ResultSet объект.

**RowId** - представление (сопоставление) значения ROWID SQL на языке программирования Java.

**Savepoint** - представление точки сохранения, которая является точкой в текущей транзакции, на которую можно ссылаться из Connection.rollback метод.

**SQLData** - интерфейс, используемый для пользовательского сопоставления пользовательского типа SQL (UDT) с классом на языке программирования Java.

**SQLInput** - входной поток, который содержит поток значений, представляющих экземпляр структурированного типа SQL или отдельного типа SQL.

**SQLOutput** - поток вывода для записи атрибутов пользовательского типа обратно в базу данных.

**SQLType** - объект, который используется для идентификации общего типа SQL, называемого типом JDBC или типом данных конкретного поставщика.

**SQLXML** - отображение на языке программирования JavaTM для типа SQL XML.

**Statement** - объект, используемый для выполнения статической инструкции SQL и возврата результатов, которые она выдает.

**Struct** - стандартное отображение в языке программирования Java для структурированного типа SQL.

**Wrapper** - интерфейс для классов JDBC, которые предоставляют возможность извлекать экземпляр делегата, когда рассматриваемый экземпляр фактически является прокси-классом.

Driver

**Driver** – это интерфейс, который должен реализовывать каждый класс драйвера.

**Методы:**

**acceptsURL(String url)** - извлекает, считает ли драйвер, что он может открыть соединение с заданным URL.

**connect(String url, Properties info)** - пытается установить соединение с базой данных по указанному URL.

**getMajorVersion()** - извлекает основной номер версии драйвера.

**getMinorVersion()** - возвращает младший номер версии драйвера.

**getParentLogger()** - возвращает родительский регистратор всех регистраторов, используемых этим драйвером.

**getPropertyInfo(String url, Properties info**) - получает информацию о возможных свойствах для этого драйвера.

**jdbcCompliant()** - сообщает, является ли этот драйвер подлинным драйвером, совместимым с JDBC ™.

Connection

**Connection** – это интерфейс который отвечает за соединение (сеанс) с определенной базой данных. Выполняются инструкции SQL, и результаты возвращаются в контексте подключения.

**Поля:**

**TRANSACTION\_NONE** - константа, указывающая на то, что транзакции не поддерживаются.

**TRANSACTION\_READ\_COMMITTED** - константа, указывающая на то, что грязные чтения предотвращены; могут возникать неповторяемые чтения и фантомные чтения.

**TRANSACTION\_READ\_UNCOMMITTED** - константа, указывающая на то, что могут выполняться грязные чтения, неповторяемые чтения и фантомные чтения.

**TRANSACTION\_REPEATABLE\_READ** - константа, указывающая на то, что грязные чтения и неповторяемые чтения запрещены; могут возникать фантомные чтения.

**TRANSACTION\_SERIALIZABLE** - константа, указывающая на то, что грязные чтения, неповторяемые чтения и фантомные чтения предотвращены.

**Методы:**

**abort(Executor executor)** - завершает открытое соединение.

**clearWarnings() -** удаляет все предупреждения, сообщенные для этого Connection объекта.

**close()** - немедленно освобождает базу данных этого Connection объекта и ресурсы JDBC вместо того, чтобы ждать их автоматического освобождения.

**commit()** - делает все изменения, внесенные с момента предыдущей фиксации / отката, постоянными и снимает все блокировки базы данных, которые в настоящее время удерживаются этим Connection объектом.

**createArrayOf(String typeName, Object[] elements)** - заводской метод для создания объектов массива.

**createBlob()** - создает объект, который реализует Blob интерфейс.

**createClob()** - создает объект, который реализует Clob интерфейс.

**createNClob()** - создает объект, который реализует NClob интерфейс.

**createSQLXML()** - создает объект, который реализует SQLXML интерфейс.

**createStatement()** - создает Statement объект для отправки инструкций SQL в базу данных.

**createStatement(int resultSetType, int resultSetConcurrency)** - создает Statement объект, который будет генерировать ResultSet объекты с заданным типом и параллелизмом.

**createStatement(int resultSetType, int resultSetConcurrency, int resultSetHoldability)** - создает Statement объект, который будет генерировать ResultSet объекты с заданным типом, параллелизмом и возможностью хранения.

**createStruct(String typeName, Object[] attributes)** - заводской метод для создания объектов Struct.

**getAutoCommit()** - извлекает текущий режим автоматической фиксации для этого Connection объекта.

**getCatalog()** - извлекает текущее имя каталога этого Connection объекта.

**getClientInfo()** - возвращает список, содержащий имя и текущее значение каждого свойства client info, поддерживаемого драйвером.

**getClientInfo(String name)** - возвращает значение свойства client info, указанного по имени.

**getHoldability()** - извлекает текущую сохраняемость ResultSet объектов, созданных с использованием этого Connection объекта.

**getMetaData()** - извлекает DatabaseMetaData объект, содержащий метаданные о базе данных, к которой этот Connection объект представляет соединение.

**getNetworkTimeout()** - извлекает количество миллисекунд, в течение которых драйвер будет ожидать завершения запроса к базе данных.

**getSchema()** - извлекает текущее имя схемы этого Connection объекта.

**getTransactionIsolation()** - извлекает текущий уровень изоляции транзакции этого Connection объекта.

**getTypeMap()** - извлекает Map объект, связанный с этим Connection объектом.

**getWarnings()** - извлекает первое предупреждение, о котором сообщают вызовы для этого Connection объекта.

**isClosed()** - извлекает, был ли этот Connection объект закрыт.

**isReadOnly()** - извлекает, находится ли этот Connection объект в режиме только для чтения.

**isValid(int timeout)** - возвращает true, если соединение не было закрыто и все еще действует.

**nativeSQL(String sql)** - преобразует данный оператор SQL в собственную грамматику SQL системы.

**prepareCall(String sql)** - создает CallableStatement объект для вызова хранимых процедур базы данных.

**prepareCall(String sql, int resultSetType, int resultSetConcurrency**) - создает CallableStatement объект, который будет генерировать ResultSet объекты с заданным типом и параллелизмом.

**prepareCall(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability) -** создает CallableStatement объект, который будет генерировать ResultSet объекты с заданным типом и параллелизмом.

**prepareStatement(String sql)** - создает PreparedStatement объект для отправки параметризованных инструкций SQL в базу данных.

**prepareStatement(String sql, int autoGeneratedKeys)** - создает PreparedStatement объект по умолчанию, который имеет возможность извлекать автоматически сгенерированные ключи.

**prepareStatement(String sql, int[] columnIndexes)** - создает PreparedStatement объект по умолчанию, способный возвращать автоматически сгенерированные ключи, обозначенные данным массивом.

**prepareStatement(String sql, int resultSetType, int resultSetConcurrency)** - создает PreparedStatement объект, который будет генерировать ResultSet объекты с заданным типом и параллелизмом.

**prepareStatement(String sql, int resultSetType, int resultSetConcurrency, int resultSetHoldability)** - PreparedStatement объект, который будет генерировать ResultSet объекты с заданным типом, параллелизмом и возможностью хранения.

**prepareStatement(String sql, String[] columnNames)** - создает PreparedStatement объект по умолчанию, способный возвращать автоматически сгенерированные ключи, обозначенные данным массивом.

**releaseSavepoint(Savepoint savepoint)** - удаляет указанный Savepoint и последующие Savepoint объекты из текущей транзакции.

**rollback()** - отменяет все изменения, внесенные в текущую транзакцию, и освобождает все блокировки базы данных, которые в настоящее время удерживаются этим Connection объектом.

**rollback(Savepoint savepoint)** - отменяет все изменения, внесенные после установки данного Savepoint объекта.

**setAutoCommit(boolean autoCommit)** - переводит режим автоматической фиксации этого соединения в заданное состояние.

**setCatalog(String catalog)** - задает заданное имя каталога, чтобы выбрать подпространство базы данных этого Connection объекта, в котором нужно работать.

**setClientInfo(Properties properties)** - задает значение свойств клиентской информации для подключения.

**setClientInfo(String name, String value)** - устанавливает значение свойства client info, указанного в name, в значение, указанное в value.

**setHoldability(int holdability)** - изменяет сохраняемость по умолчанию ResultSet объектов, созданных с использованием этого Connection объекта, на заданную сохраняемость.

**setNetworkTimeout(Executor executor, int milliseconds)** - устанавливает максимальный период, в течение которого Connection или объекты, созданные из Connection, будут ожидать ответа базы данных на любой запрос.

**setReadOnly(boolean readOnly)** - переводит это соединение в режим только для чтения в качестве подсказки драйверу для включения оптимизации базы данных.

**setSavepoint()** - создает неназванную точку сохранения в текущей транзакции и возвращает новый Savepoint объект, который ее представляет.

**setSavepoint(String name)** - создает точку сохранения с заданным именем в текущей транзакции и возвращает новый Savepoint объект, который ее представляет.

**setSchema(String schema)** - задает для заданного имени схемы значение access.

**setTransactionIsolation(int level)** - пытается изменить уровень изоляции транзакции для этого Connection объекта на заданный.

**setTypeMap(Map<String,Class<?>> map)** - устанавливает данный TypeMap объект в качестве сопоставления типов для этого Connection объекта.

Statement

**Statement** – это интерфейс используемый для выполнения статической инструкции SQL и возврата результатов, которые она выдает. Задействован для доступа к БД при решении общих вопросов.

**Поля:**

**CLOSE\_ALL\_RESULTS -** константа, указывающая, что все ResultSet объекты, которые ранее оставались открытыми, должны быть закрыты при вызове getMoreResults.

**CLOSE\_CURRENT\_RESULT -** константа, указывающая, что текущий ResultSet объект должен быть закрыт при вызове getMoreResults.

**EXECUTE\_FAILED -** константа, указывающая на то, что произошла ошибка при выполнении пакетной инструкции.

**KEEP\_CURRENT\_RESULT -** константа, указывающая, что текущий ResultSet объект не должен быть закрыт при вызове getMoreResults.

**NO\_GENERATED\_KEYS -** константа, указывающая, что сгенерированные ключи не должны быть доступны для извлечения.

**RETURN\_GENERATED\_KEYS -** константа, указывающая, что сгенерированные ключи должны быть доступны для извлечения.

**SUCCESS\_NO\_INFO -** константа, указывающая на то, что пакетная инструкция выполнена успешно, но количество строк, на которые она повлияла, не подсчитывается.

**Методы:**

**addBatch(String sql)** - добавляет данную команду SQL в текущий список команд для этого Statement объекта.

**cancel()** - отменяет этот Statement объект, если и СУБД, и драйвер поддерживают прерывание инструкции SQL.

**clearBatch()** - очищает текущий список SQL-команд этого Statement объекта.

**clearWarnings()** - удаляет все предупреждения, сообщенные об этом Statement объекте.

**close()** - немедленно освобождает базу данных этого Statement объекта и ресурсы JDBC вместо того, чтобы ждать, пока это произойдет, когда он будет автоматически закрыт.

**closeOnCompletion()** - указывает, что это Statement будет закрыто, когда все его зависимые результирующие наборы будут закрыты.

**enquoteIdentifier(String identifier, boolean alwaysQuote)** - возвращает идентификатор SQL.

**enquoteLiteral(String val)** – возвращает String, заключенное в одинарные кавычки.

**enquoteNCharLiteral(String val)** - возвращает String, представляющий литерал национального набора символов, заключенный в одинарные кавычки и предваряемый заглавной буквой N.

**execute(String sql)** - выполняет данную инструкцию SQL, которая может возвращать несколько результатов.

**execute(String sql, int autoGeneratedKeys)** - выполняет данную инструкцию SQL, которая может возвращать несколько результатов, и сигнализирует драйверу, что любые автоматически сгенерированные ключи должны быть доступны для извлечения.

**execute(String sql, int[] columnIndexes)** - выполняет данную инструкцию SQL, которая может возвращать несколько результатов, и сигнализирует драйверу, что автоматически сгенерированные ключи, указанные в данном массиве, должны быть доступны для извлечения.

**execute(String sql, String[] columnNames)** - выполняет данную инструкцию SQL, которая может возвращать несколько результатов, и сигнализирует драйверу, что автоматически сгенерированные ключи, указанные в данном массиве, должны быть доступны для извлечения.

**executeBatch()** - отправляет пакет команд в базу данных для выполнения, и если все команды выполняются успешно, возвращает массив подсчетов обновлений.

**executeLargeBatch()** - отправляет пакет команд в базу данных для выполнения, и если все команды выполняются успешно, возвращает массив подсчетов обновлений.

**executeLargeUpdate(String sql)** - выполняет данную инструкцию SQL, которая может быть инструкцией INSERT, UPDATE или DELETE или инструкцией SQL, которая ничего не возвращает, такой как инструкция SQL DDL.

**executeLargeUpdate(String sql, int autoGeneratedKeys)** - выполняет данную инструкцию SQL и сигнализирует драйверу с заданным флагом о том, должны ли автоматически сгенерированные ключи, созданные этим Statement объектом, быть доступны для извлечения.

**executeLargeUpdate(String sql, int[] columnIndexes)** - выполняет данную инструкцию SQL и сигнализирует драйверу, что автоматически сгенерированные ключи, указанные в данном массиве, должны быть доступны для извлечения.

**executeLargeUpdate(String sql, String[] columnNames)** - выполняет данную инструкцию SQL и сигнализирует драйверу, что автоматически сгенерированные ключи, указанные в данном массиве, должны быть доступны для извлечения.

**executeQuery(String sql)** - выполняет данную инструкцию SQL, которая возвращает один ResultSet объект.

**executeUpdate(String sql)** - выполняет данную инструкцию SQL, которая может быть инструкцией INSERT, UPDATE или DELETE или инструкцией SQL, которая ничего не возвращает, такой как инструкция SQL DDL.

**executeUpdate(String sql, int autoGeneratedKeys)** - выполняет данную инструкцию SQL и сигнализирует драйверу с заданным флагом о том, должны ли автоматически сгенерированные ключи, созданные этим Statement объектом, быть доступны для извлечения.

**executeUpdate(String sql, int[] columnIndexes)** - выполняет данную инструкцию SQL и сигнализирует драйверу, что автоматически сгенерированные ключи, указанные в данном массиве, должны быть доступны для извлечения.

**executeUpdate(String sql, String[] columnNames)** - выполняет данную инструкцию SQL и сигнализирует драйверу, что автоматически сгенерированные ключи, указанные в данном массиве, должны быть доступны для извлечения.

**getConnection()** - извлекает Connection объект, который создал этот Statement объект.

**getFetchDirection()** - извлекает направление для выборки строк из таблиц базы данных, которое используется по умолчанию для наборов результатов, сгенерированных из этого Statement объекта.

**getFetchSize()** - извлекает количество строк результирующего набора, которое является размером выборки по умолчанию для ResultSet объектов, сгенерированных из этого Statement объекта.

**getGeneratedKeys()** - извлекает любые автоматически сгенерированные ключи, созданные в результате выполнения этого Statement объекта.

**getLargeMaxRows()** - извлекает максимальное количество строк, которое может содержать ResultSet объект, созданный этим Statement объектом.

**getLargeUpdateCount()** - извлекает текущий результат в виде количества обновлений; если результатом является ResultSet объект или результатов больше нет, возвращается -1.

**getMaxFieldSize()** - извлекает максимальное количество байтов, которое может быть возвращено для значений символов и двоичных столбцов в ResultSet объекте, созданном этим Statement объектом.

**getMaxRows()** - извлекает максимальное количество строк, которое может содержать ResultSet объект, созданный этим Statement объектом.

**getMoreResults()** - переходит к следующему результату этого Statement объекта, возвращает, true если это ResultSet объект, и неявно закрывает все текущие ResultSet объекты, полученные с помощью метода getResultSet.

**getMoreResults(int current)** - переходит к следующему результату этого Statement объекта, обрабатывает любые текущие ResultSet объекты в соответствии с инструкциями, указанными данным флагом, и возвращает, true если следующим результатом является ResultSet объект.

**getQueryTimeout()** - возвращает количество секунд, в течение которых драйвер будет ожидать выполнения Statement объекта.

**getResultSet()** - извлекает текущий результат в виде ResultSet объекта.

**getResultSetConcurrency()** - извлекает параллелизм результирующего набора для ResultSet объектов, сгенерированных этим Statement объектом.

**getResultSetHoldability()** - извлекает возможность хранения результирующего набора для ResultSet объектов, сгенерированных этим Statement объектом.

**getResultSetType()** - извлекает тип результирующего набора для ResultSet объектов, сгенерированных этим Statement объектом.

**getUpdateCount()** - извлекает текущий результат в виде количества обновлений; если результатом является ResultSet объект или результатов больше нет, возвращается -1.

**getWarnings()** - извлекает первое предупреждение, сообщенное вызовами для этого Statement объекта.

**isClosed()** - извлекает, был ли этот Statement объект закрыт.

**isCloseOnCompletion()** - возвращает значение, указывающее, будет ли это Statement закрыто, когда все его зависимые результирующие наборы будут закрыты.

**isPoolable()** - возвращает значение, указывающее, является Statement ли оно объединяемым или нет.

**isSimpleIdentifier(String identifier)** - извлекает, является ли identifier простым идентификатором SQL.

**setCursorName(String name)** - присваивает заданному имени курсора SQL String, которое будет использоваться последующими Statement объектными execute методами.

**setEscapeProcessing(boolean enable)** - включает или выключает обработку escape.

**setFetchDirection(int direction)** - дает драйверу подсказку относительно направления, в котором будут обрабатываться строки в ResultSet объектах, созданных с использованием этого Statement объекта.

**setFetchSize(int rows)** - дает драйверу JDBC подсказку относительно количества строк, которые следует извлечь из базы данных, когда для ResultSet объектов, сгенерированных этим Statement, требуется больше строк.

**setLargeMaxRows(long max)** - устанавливает ограничение на максимальное количество строк, которое может содержать любой ResultSet объект, сгенерированный этим Statement объектом, с заданным числом.

**setMaxFieldSize(int max)** - устанавливает ограничение на максимальное количество байтов, которое может быть возвращено для значений символов и двоичных столбцов в ResultSet объекте, созданном этим Statement объектом.

**setMaxRows(int max)** - устанавливает ограничение на максимальное количество строк, которое может содержать любой ResultSet объект, сгенерированный этим Statement объектом, с заданным числом.

**setPoolable(boolean poolable)** - запрашивает, чтобы Statement был объединен или не был объединен.

**setQueryTimeout(int seconds)** - устанавливает количество секунд, в течение которых драйвер будет ожидать выполнения Statement объекта, равным заданному количеству секунд.

PreparedStatement

**PreparedStatement** – это интерфейс представляющий предварительно скомпилированную инструкцию SQL. Способен принимать различные параметры.

**Методы:**

**addBatch()** - добавляет набор параметров к пакету команд этого PreparedStatement объекта.

**clearParameters()** - немедленно очищает текущие значения параметров.

**execute()** - выполняет инструкцию SQL в этом PreparedStatement объекте, который может быть любой инструкцией SQL.

**executeLargeUpdate()** - выполняет инструкцию SQL в этом PreparedStatement объекте, который должен быть инструкцией языка обработки данных SQL (DML), такой как INSERT, UPDATE или DELETE; или инструкция SQL, которая ничего не возвращает, такая как инструкция DDL.

**executeQuery()** - выполняет SQL-запрос в этом PreparedStatement объекте и возвращает ResultSet объект, сгенерированный запросом.

**executeUpdate()** - выполняет инструкцию SQL в этом PreparedStatement объекте, который должен быть инструкцией языка обработки данных SQL (DML), такой как INSERT, UPDATE или DELETE; или инструкция SQL, которая ничего не возвращает, такая как инструкция DDL.

**getMetaData()** - извлекает ResultSetMetaData объект, содержащий информацию о столбцах ResultSet объекта, которая будет возвращена при выполнении этого PreparedStatement объекта.

**getParameterMetaData()** - извлекает количество, типы и свойства параметров этого PreparedStatement объекта.

**setArray(int parameterIndex, Array x)** - устанавливает назначенный параметр для данного java.sql.Array объекта.

**setAsciiStream(int parameterIndex, InputStream x)** - устанавливает назначенный параметр для данного входного потока.

**setAsciiStream(int parameterIndex, InputStream x, int length)** - устанавливает назначенный параметр для данного входного потока, который будет иметь указанное количество байтов.

**setAsciiStream(int parameterIndex, InputStream x, long length)** - устанавливает назначенный параметр для данного входного потока, который будет иметь указанное количество байтов.

**setBigDecimal(int parameterIndex, BigDecimal x)** - присваивает указанному параметру заданное java.math.BigDecimal значение.

**setBinaryStream(int parameterIndex, InputStream x)** - устанавливает назначенный параметр для данного входного потока.

**setBinaryStream(int parameterIndex, InputStream x, int length)** - устанавливает назначенный параметр для данного входного потока, который будет иметь указанное количество байтов.

**setBinaryStream(int parameterIndex, InputStream x, long length)** - устанавливает назначенный параметр для данного входного потока, который будет иметь указанное количество байтов.

**setBlob(int parameterIndex, InputStream inputStream)** - устанавливает указанный параметр в InputStream объект.

**setBlob(int parameterIndex, InputStream inputStream, long length)** - устанавливает указанный параметр в InputStream объект.

**setBlob(int parameterIndex, Blob x)** - устанавливает назначенный параметр для данного java.sql.Blob объекта.

**setBoolean(int parameterIndex, boolean x)** - присваивает указанному параметру заданное Java boolean значение.

**setByte(int parameterIndex, byte x)** - присваивает указанному параметру заданное Java byte значение.

**setBytes(int parameterIndex, byte[] x)** - устанавливает назначенный параметр в заданный Java-массив байтов.

**setCharacterStream(int parameterIndex, Reader reader)** - устанавливает назначенный параметр для данного Reader объекта.

**setCharacterStream(int parameterIndex, Reader reader, int length)** - устанавливает назначенный параметр для данного Reader объекта, который представляет собой заданное количество символов длиной.

**setCharacterStream(int parameterIndex, Reader reader, long length)** - устанавливает назначенный параметр для данного Reader объекта, который представляет собой заданное количество символов длиной.

**setClob(int parameterIndex, Reader reader**) - устанавливает указанный параметр в Reader объект.

**setClob(int parameterIndex, Reader reader, long length)** - устанавливает указанный параметр в Reader объект.

**setClob(int parameterIndex, Clob x)** - устанавливает назначенный параметр для данного java.sql.Clob объекта.

**setDate(int parameterIndex, Date x)** - присваивает указанному параметру заданное java.sql.Date значение, используя часовой пояс по умолчанию виртуальной машины, на которой запущено приложение.

**setDate(int parameterIndex, Date x, Calendar cal)** - присваивает указанному параметру заданное java.sql.Date значение, используя заданный Calendar объект.

**setDouble(int parameterIndex, double x)** - присваивает указанному параметру заданное Java double значение.

**setFloat(int parameterIndex, float x)** - присваивает указанному параметру заданное Java float значение.

**setInt(int parameterIndex, int x)** - присваивает указанному параметру заданное Java int значение.

**setLong(int parameterIndex, long x)** - присваивает указанному параметру заданное Java long значение.

**setNCharacterStream(int parameterIndex, Reader value)** - устанавливает указанный параметр в Reader объект.

**setNCharacterStream(int parameterIndex, Reader value, long length)** - устанавливает указанный параметр в Reader объект.

**setNClob(int parameterIndex, Reader reader)** - устанавливает указанный параметр в Reader объект.

**setNClob(int parameterIndex, Reader reader, long length)** - устанавливает указанный параметр в Reader объект.

**setNClob(int parameterIndex, NClob value)** - устанавливает указанный параметр в java.sql.NClob объект.

**setNString(int parameterIndex, String value)** - устанавливает назначенный параметр для данного String объекта.

**setNull(int parameterIndex, int sqlType)** - устанавливает указанный параметр в SQL NULL.

**setNull(int parameterIndex, int sqlType, String typeName)** - устанавливает указанный параметр в SQL NULL.

**setObject(int parameterIndex, Object x)** - устанавливает значение назначенного параметра с использованием данного объекта.

**setObject(int parameterIndex, Object x, int targetSqlType)** - устанавливает значение назначенного параметра для данного объекта.

**setObject(int parameterIndex, Object x, int targetSqlType, int scaleOrLength)** - устанавливает значение назначенного параметра для данного объекта.

**setObject(int parameterIndex, Object x, SQLType targetSqlType)** - устанавливает значение назначенного параметра для данного объекта.

**setObject(int parameterIndex, Object x, SQLType targetSqlType, int scaleOrLength)** - устанавливает значение назначенного параметра для данного объекта.

**setRef(int parameterIndex, Ref x)** - присваивает указанному параметру заданное REF(<structured-type>) значение.

**setRowId(int parameterIndex, RowId x)** - устанавливает назначенный параметр для данного java.sql.RowId объекта.

**setShort(int parameterIndex, short x)** - присваивает указанному параметру заданное Java short значение.

**setSQLXML(int parameterIndex, SQLXML xmlObject)** - устанавливает назначенный параметр для данного java.sql.SQLXML объекта.

**setString(int parameterIndex, String x)** - присваивает указанному параметру заданное Java String значение.

**setTime(int parameterIndex, Time x)** - присваивает указанному параметру заданное java.sql.Time значение.

**setTime(int parameterIndex, Time x, Calendar cal)** - присваивает указанному параметру заданное java.sql.Time значение, используя заданный Calendar объект.

**setTimestamp(int parameterIndex, Timestamp x)** - присваивает указанному параметру заданное java.sql.Timestamp значение.

**setTimestamp(int parameterIndex, Timestamp x, Calendar cal)** - присваивает указанному параметру заданное java.sql.Timestamp значение, используя заданный Calendar объект.

**setURL(int parameterIndex, URL x)** - присваивает указанному параметру заданное java.net.URL значение.

CallableStatement

**CallableStatement** – это интерфейс используемый для выполнения хранимых процедур SQL. Помогает заполучить доступ к разнообразным процедурам «хранилищ структурированных данных».

**Методы:**

**getArray(int parameterIndex)** - извлекает значение указанного ARRAY параметра JDBC как Array объект на языке программирования Java.

**getArray(String parameterName)** - извлекает значение ARRAY параметра JDBC как Array объект на языке программирования Java.

**getBigDecimal(int parameterIndex)** - извлекает значение указанного NUMERIC параметра JDBC в виде java.math.BigDecimal объекта с таким количеством цифр справа от десятичной точки, сколько содержит значение.

**getBigDecimal(String parameterName)** - извлекает значение NUMERIC параметра JDBC в виде java.math.BigDecimal объекта с таким количеством цифр справа от десятичной точки, сколько содержит значение.

**getBlob(int parameterIndex)** - извлекает значение указанного BLOB параметра JDBC в виде Blob объекта на языке программирования Java.

**getBlob(String parameterName)** - извлекает значение BLOB параметра JDBC в виде Blob объекта на языке программирования Java.

**getBoolean(int parameterIndex)** - извлекает значение указанного JDBC BIT или BOOLEAN параметра в качестве boolean на языке программирования Java.

**getBoolean(String parameterName)** - извлекает значение JDBC BIT или BOOLEAN параметра как boolean на языке программирования Java.

**getByte(int parameterIndex)** - извлекает значение указанного TINYINT параметра JDBC в качестве byte на языке программирования Java.

**getByte(String parameterName)** - извлекает значение TINYINT параметра JDBC в качестве byte на языке программирования Java.

**getBytes(int parameterIndex)** - извлекает значение указанного JDBC BINARY или VARBINARY параметра в виде массива byte значений на языке программирования Java.

**getBytes(String parameterName)** - извлекает значение JDBC BINARY или VARBINARY параметра в виде массива byte значений на языке программирования Java.

**getCharacterStream(int parameterIndex)** - извлекает значение указанного параметра в виде java.io.Reader объекта на языке программирования Java.

**getCharacterStream(String parameterName)** - извлекает значение указанного параметра в виде java.io.Reader объекта на языке программирования Java.

**getClob(int parameterIndex)** - извлекает значение указанного CLOB параметра JDBC в виде java.sql.Clob объекта на языке программирования Java.

**getClob(String parameterName)** - извлекает значение CLOB параметра JDBC в виде java.sql.Clob объекта на языке программирования Java.

**getDate(int parameterIndex)** - извлекает значение указанного DATE параметра JDBC в виде java.sql.Date объекта.

**getDate(int parameterIndex, Calendar cal)** - извлекает значение указанного DATE параметра JDBC в виде java.sql.Date объекта, используя данный Calendar объект для построения даты.

**getDate(String parameterName)** - извлекает значение DATE параметра JDBC в виде java.sql.Date объекта.

**getDate(String parameterName, Calendar cal)** - извлекает значение DATE параметра JDBC в виде java.sql.Date объекта, используя данный Calendar объект для построения даты.

**getDouble(int parameterIndex)** - извлекает значение указанного DOUBLE параметра JDBC в качестве double на языке программирования Java.

**getDouble(String parameterName)** - извлекает значение DOUBLE параметра JDBC в качестве double на языке программирования Java.

**getFloat(int parameterIndex)** - извлекает значение указанного FLOAT параметра JDBC в качестве float на языке программирования Java.

**getFloat(String parameterName)** - извлекает значение FLOAT параметра JDBC в качестве float на языке программирования Java.

**getInt(int parameterIndex)** - извлекает значение указанного INTEGER параметра JDBC в качестве int на языке программирования Java.

**getInt(String parameterName)** - извлекает значение INTEGER параметра JDBC в качестве int на языке программирования Java.

**getLong(int parameterIndex)** - извлекает значение указанного BIGINT параметра JDBC в качестве long на языке программирования Java.

**getLong(String parameterName)** -

Извлекает значение BIGINT параметра JDBC в качестве long на языке программирования Java.

**getNCharacterStream(int parameterIndex)** - ивлекает значение указанного параметра в виде java.io.Reader объекта на языке программирования Java.

**getNCharacterStream(String parameterName)** - извлекает значение указанного параметра в виде java.io.Reader объекта на языке программирования Java.

**getNClob(int parameterIndex)** - извлекает значение указанного NCLOB параметра JDBC в виде java.sql.NClob объекта на языке программирования Java.

**getNClob(String parameterName)** - извлекает значение NCLOB параметра JDBC в виде java.sql.NClob объекта на языке программирования Java.

**getNString(int parameterIndex)** - извлекает значение назначенного NCHAR, NVARCHAR или LONGNVARCHAR параметра как String на языке программирования Java.

**getNString(String parameterName)** - извлекает значение назначенного NCHAR, NVARCHAR или LONGNVARCHAR параметра как String на языке программирования Java.

**getObject(int parameterIndex)** - извлекает значение указанного параметра в виде Object на языке программирования Java.

**getObject(int parameterIndex, Class<T> type)** - возвращает объект, представляющий значение параметра OUT parameterIndex, и преобразует из типа параметра SQL в запрошенный тип данных Java, если преобразование поддерживается.

**getObject(int parameterIndex, Map<String,Class<?>> map)** - возвращает объект, представляющий значение параметра OUT, parameterIndex и используется map для пользовательского сопоставления значения параметра.

**getObject(String parameterName)** - извлекает значение параметра в виде Object на языке программирования Java.

**getObject(String parameterName, Class<T> type)** - возвращает объект, представляющий значение параметра OUT parameterName, и преобразует из типа параметра SQL в запрошенный тип данных Java, если преобразование поддерживается.

**getObject(String parameterName, Map<String,Class<?>> map)** - возвращает объект, представляющий значение параметра OUT, parameterName и используется map для пользовательского сопоставления значения параметра.

**getRef(int parameterIndex)** - извлекает значение указанного REF(<structured-type>) параметра JDBC в виде Ref объекта на языке программирования Java.

**getRef(String parameterName)** - извлекает значение REF(<structured-type>) параметра JDBC в виде Ref объекта на языке программирования Java.

**getRowId(int parameterIndex)** - извлекает значение указанного ROWID параметра JDBC в виде java.sql.RowId объекта.

**getRowId(String parameterName)** - извлекает значение указанного ROWID параметра JDBC в виде java.sql.RowId объекта.

**getShort(int parameterIndex)** - извлекает значение указанного SMALLINT параметра JDBC в качестве short на языке программирования Java.

**getShort(String parameterName)** - извлекает значение SMALLINT параметра JDBC в качестве short на языке программирования Java.

**getSQLXML(int parameterIndex)** - извлекает значение указанного SQL XML параметра в виде java.sql.SQLXML объекта на языке программирования Java.

**getSQLXML(String parameterName)** - извлекает значение указанного SQL XML параметра в виде java.sql.SQLXML объекта на языке программирования Java.

**getString(int parameterIndex)** - извлекает значение указанного JDBC CHAR, VARCHAR или LONGVARCHAR параметра как String на языке программирования Java.

**getString(String parameterName)** - извлекает значение JDBC CHAR, VARCHAR или LONGVARCHAR параметра как String на языке программирования Java.

**getTime(int parameterIndex)** - извлекает значение указанного TIME параметра JDBC в виде java.sql.Time объекта.

**getTime(int parameterIndex, Calendar cal)** - извлекает значение указанного TIME параметра JDBC в виде java.sql.Time объекта, используя данный Calendar объект для построения времени.

**getTime(String parameterName)** - извлекает значение TIME параметра JDBC в виде java.sql.Time объекта.

**getTime(String parameterName, Calendar cal)** - извлекает значение TIME параметра JDBC в виде java.sql.Time объекта, используя данный Calendar объект для построения времени.

**getTimestamp(int parameterIndex)** - извлекает значение указанного TIMESTAMP параметра JDBC в виде java.sql.Timestamp объекта.

**getTimestamp(int parameterIndex, Calendar cal)** - извлекает значение указанного TIMESTAMP параметра JDBC в виде java.sql.Timestamp объекта, используя данный Calendar объект для построения Timestamp объекта.

**getTimestamp(String parameterName)** - извлекает значение TIMESTAMP параметра JDBC в виде java.sql.Timestamp объекта.

**getTimestamp(String parameterName, Calendar cal)** - извлекает значение TIMESTAMP параметра JDBC в виде java.sql.Timestamp объекта, используя данный Calendar объект для построения Timestamp объекта.

**getURL(int parameterIndex)** - извлекает значение указанного DATALINK параметра JDBC в виде java.net.URL объекта.

**getURL(String parameterName)** - извлекает значение DATALINK параметра JDBC в виде java.net.URL объекта.

**registerOutParameter(int parameterIndex, int sqlType)** - регистрирует параметр OUT в порядковой позиции parameterIndex в соответствии с типом JDBC sqlType.

**registerOutParameter(int parameterIndex, int sqlType, int scale)** - регистрирует параметр в порядковой позицииparameterIndex, чтобы он имел тип JDBC sqlType.

**registerOutParameter(int parameterIndex, int sqlType, String typeName)** - регистрирует назначенный выходной параметр.

**registerOutParameter(int parameterIndex, SQLType sqlType)** - регистрирует параметр OUT в порядковой позиции parameterIndex в соответствии с типом JDBC sqlType.

**registerOutParameter(int parameterIndex, SQLType sqlType, int scale)** - регистрирует параметр в порядковой позицииparameterIndex, чтобы он имел тип JDBC sqlType.

**registerOutParameter(int parameterIndex, SQLType sqlType, String typeName)** - регистрирует назначенный выходной параметр.

**registerOutParameter(String parameterName, int sqlType)** - регистрирует параметр OUT с именем, parameterName соответствующим типу JDBC sqlType.

**registerOutParameter(String parameterName, int sqlType, int scale)** - регистрирует параметр с именем, parameterName который должен иметь тип JDBC sqlType.

**registerOutParameter(String parameterName, int sqlType, String typeName)** - регистрирует назначенный выходной параметр.

**registerOutParameter(String parameterName, SQLType sqlType)** - регистрирует параметр OUT с именем, parameterName соответствующим типу JDBC sqlType.

**registerOutParameter(String parameterName, SQLType sqlType, int scale)** - регистрирует параметр с именем, parameterName который должен иметь тип JDBC sqlType.

**registerOutParameter(String parameterName, SQLType sqlType, String typeName)** -регистрирует назначенный выходной параметр.

**setAsciiStream(String parameterName, InputStream x)** - устанавливает назначенный параметр для данного входного потока.

**setAsciiStream(String parameterName, InputStream x, int length)** - устанавливает назначенный параметр для данного входного потока, который будет иметь указанное количество байтов.

**setAsciiStream(String parameterName, InputStream x, long length)** - устанавливает назначенный параметр для данного входного потока, который будет иметь указанное количество байтов.

**setBigDecimal(String parameterName, BigDecimal x)** - присваивает указанному параметру заданное java.math.BigDecimal значение.

**setBinaryStream(String parameterName, InputStream x)** - устанавливает назначенный параметр для данного входного потока.

**setBinaryStream(String parameterName, InputStream x, int length)** - устанавливает назначенный параметр для данного входного потока, который будет иметь указанное количество байтов.

**setBinaryStream(String parameterName, InputStream x, long length)** - устанавливает назначенный параметр для данного входного потока, который будет иметь указанное количество байтов.

**setBlob(String parameterName, InputStream inputStream**) - устанавливает назначенный параметр для InputStream объекта.

**setBlob(String parameterName, InputStream inputStream, long length)** - устанавливает назначенный параметр для InputStream объекта.

**setBlob(String parameterName, Blob x)** - устанавливает назначенный параметр для данного java.sql.Blob объекта.

**setBoolean(String parameterName, boolean x)** - присваивает указанному параметру заданное Java boolean значение.

**setByte(String parameterName, byte x)** - присваивает указанному параметру заданное Java byte значение.

**setBytes(String parameterName, byte[] x)** - устанавливает назначенный параметр в заданный Java-массив байтов.

**setCharacterStream(String parameterName, Reader reader)** - устанавливает назначенный параметр для данного Reader объекта.

**setCharacterStream(String parameterName, Reader reader, int length)** - устанавливает назначенный параметр для данного Reader объекта, который представляет собой заданное количество символов длиной.

**setCharacterStream(String parameterName, Reader reader, long length)** - устанавливает назначенный параметр для данного Reader объекта, который представляет собой заданное количество символов длиной.

**setClob(String parameterName, Reader reader)** - устанавливает указанный параметр в Reader объект.

**setClob(String parameterName, Reader reader, long length)** - устанавливает указанный параметр в Reader объект.

**setClob(String parameterName, Clob x)** - устанавливает назначенный параметр для данного java.sql.Clob объекта.

**setDate(String parameterName, Date x)** - присваивает указанному параметру заданное java.sql.Date значение, используя часовой пояс по умолчанию виртуальной машины, на которой запущено приложение.

**setDate(String parameterName, Date x, Calendar cal)** - присваивает указанному параметру заданное java.sql.Date значение, используя заданный Calendar объект.

**setDouble(String parameterName, double x)** - присваивает указанному параметру заданное Java double значение.

**setFloat(String parameterName, float x)** - присваивает указанному параметру заданное Java float значение.

**setInt(String parameterName, int x)** - присваивает указанному параметру заданное Java int значение.

**setLong(String parameterName, long x)** - присваивает указанному параметру заданное Java long значение.

**setNCharacterStream(String parameterName, Reader value)** - устанавливает указанный параметр в Reader объект.

**setNCharacterStream(String parameterName, Reader value, long length)** - устанавливает указанный параметр в Reader объект.

**setNClob(String parameterName, Reader reader)** - устанавливает указанный параметр в Reader объект.

**setNClob(String parameterName, Reader reader, long length)** - устанавливает указанный параметр в Reader объект.

**setNClob(String parameterName, NClob value)** - устанавливает указанный параметр в java.sql.NClob объект.

**setNString(String parameterName, String value)** - устанавливает назначенный параметр для данного String объекта.

**setNull(String parameterName, int sqlType)** - устанавливает указанный параметр в SQL NULL.

**setNull(String parameterName, int sqlType, String typeName)** - устанавливает указанный параметр в SQL NULL.

**setObject(String parameterName, Object x)** - устанавливает значение назначенного параметра для данного объекта.

**setObject(String parameterName, Object x, int targetSqlType)** - устанавливает значение назначенного параметра для данного объекта.

**setObject(String parameterName, Object x, int targetSqlType, int scale) -** устанавливает значение назначенного параметра для данного объекта.

**setObject(String parameterName, Object x, SQLType targetSqlType) -** устанавливает значение назначенного параметра для данного объекта.

**setObject(String parameterName, Object x, SQLType targetSqlType, int scaleOrLength)** - устанавливает значение назначенного параметра для данного объекта.

**setRowId(String parameterName, RowId x)** - устанавливает назначенный параметр для данного java.sql.RowId объекта.

**setShort(String parameterName, short x)** - присваивает указанному параметру заданное Java short значение.

**setSQLXML(String parameterName, SQLXML xmlObject)** - устанавливает назначенный параметр для данного java.sql.SQLXML объекта.

**setString(String parameterName, String x)** - присваивает указанному параметру заданное Java String значение.

**setTime(String parameterName, Time x)** - присваивает указанному параметру заданное java.sql.Time значение.

**setTime(String parameterName, Time x, Calendar cal)** - присваивает указанному параметру заданное java.sql.Time значение, используя заданный Calendar объект.

**setTimestamp(String parameterName, Timestamp x) -** присваивает указанному параметру заданное java.sql.Timestamp значение.

**setTimestamp(String parameterName, Timestamp x, Calendar cal)** - присваивает указанному параметру заданное java.sql.Timestamp значение, используя заданный Calendar объект.

**setURL(String parameterName, URL val)** - устанавливает назначенный параметр для данного java.net.URL объекта.

**wasNull()** - извлекает, имел ли последний прочитанный параметр OUT значение SQL NULL.

Классы:

**Date -** тонкая оболочка вокруг значения в миллисекунду, которая позволяет JDBC идентифицировать это как значение SQL DATE.

**DriverManager** - базовая служба для управления набором драйверов JDBC. ПРИМЕЧАНИЕ: DataSource Интерфейс, новый в API JDBC 2.0, предоставляет другой способ подключения к источнику данных.

**DriverPropertyInfo** - свойства драйвера для установления соединения.

**SQLPermission** - разрешение, для которого SecurityManager будет проверяться, когда код, выполняющий приложение с SecurityManager включенным, вызывает DriverManager.deregisterDriver метод, DriverManager.setLogWriter метод, DriverManager.setLogStream (устаревший) метод, SyncFactory.setJNDIContext метод, SyncFactory.setLogger метод, Connection.setNetworktimeout метод или Connection.abort метод.

**Time** - тонкая оболочка вокруг java.util.Date класса, которая позволяет JDBC API идентифицировать это как SQL TIME значение.

**Timestamp** - тонкая оболочка вокруг, java.util.Date которая позволяет JDBC API идентифицировать это как SQL TIMESTAMP значение.

**Types** - класс, определяющий константы, которые используются для идентификации универсальных типов SQL, называемых типами JDBC.

DriverManager

**DriverManager** – это класс который выступает в роли Базовой службы для управления набором драйверов JDBC.

**Методы:**

**deregisterDriver(Driver driver)** - удаляет указанный драйвер из DriverManager списка зарегистрированных драйверов.

**getConnection(String url**) - пытается установить соединение с указанным URL базы данных.

**getConnection(String url, Properties info)** - пытается установить соединение с указанным URL базы данных.

**getConnection(String url, String user, String password)** - пытается установить соединение с указанным URL базы данных.

**getDriver(String url)** - пытается найти драйвер, который понимает указанный URL.

**getDrivers()** - извлекает перечисление со всеми загруженными в данный момент драйверами JDBC, к которым текущий вызывающий имеет доступ.

**getLoginTimeout()** - получает максимальное время в секундах, которое драйвер может ожидать при попытке входа в базу данных.

**getLogWriter()** - извлекает средство записи журнала.

**println(String message)** - выводит сообщение в текущий поток журнала JDBC.

**registerDriver(Driver driver)** - регистрирует данный драйвер с помощью DriverManager.

**registerDriver(Driver driver, DriverAction da)** - регистрирует данный драйвер с помощью DriverManager.

**setLoginTimeout(int seconds)** - задает максимальное время в секундах, которое драйвер будет ждать при попытке подключения к базе данных после идентификации драйвера.

**setLogWriter(PrintWriter out)** - задает PrintWriter объект ведения журнала / трассировки, который используется DriverManager и всеми драйверами.