

CloudaIDE

web framework for
database developers

CloudaIDE – web framework for database developers

- Database and User Interface – common denominators of web applications
- The idea of CloudaIDE – use the database development methods to design a web application

Goals

- easy to learn
- fast to develop
- ergonomony – mouseless data entry
- easy migration of Oracle Forms applications
- easy deployment
- scalability
- extensibility
- mobile apps

Reasons (among many others)

Removal of the Java plug-in (applets) support from browsers

Three-tier architecture

User
Interface

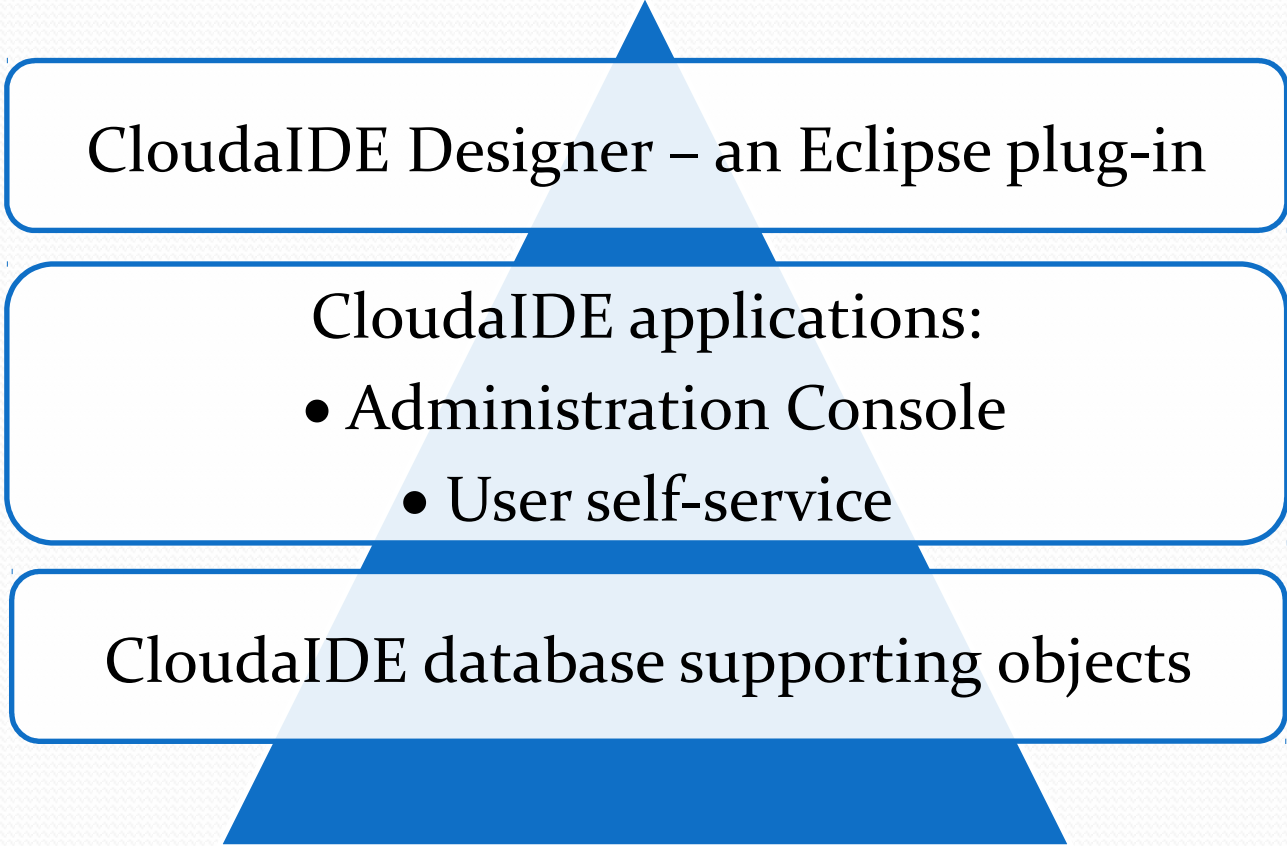


Application
server



Database

- Layout – Screen Editor
- User and Server Interaction– “triggers”. Programs in MT a simple language resembling PSQL
- Security tasks. Checking validity of client requests
- Translating request/response between client and server
- Application logic – gateway between MT and Java (in case of not sufficient power of database Stored Procedures)
- Data Storage
- Application logic implemented using Stored Procedures



CloudaIDE Designer – an Eclipse plug-in

CloudaIDE applications:

- Administration Console
 - User self-service

CloudaIDE database supporting objects

CloudaIDE Designer

Tool Palette

Data Source Explorer

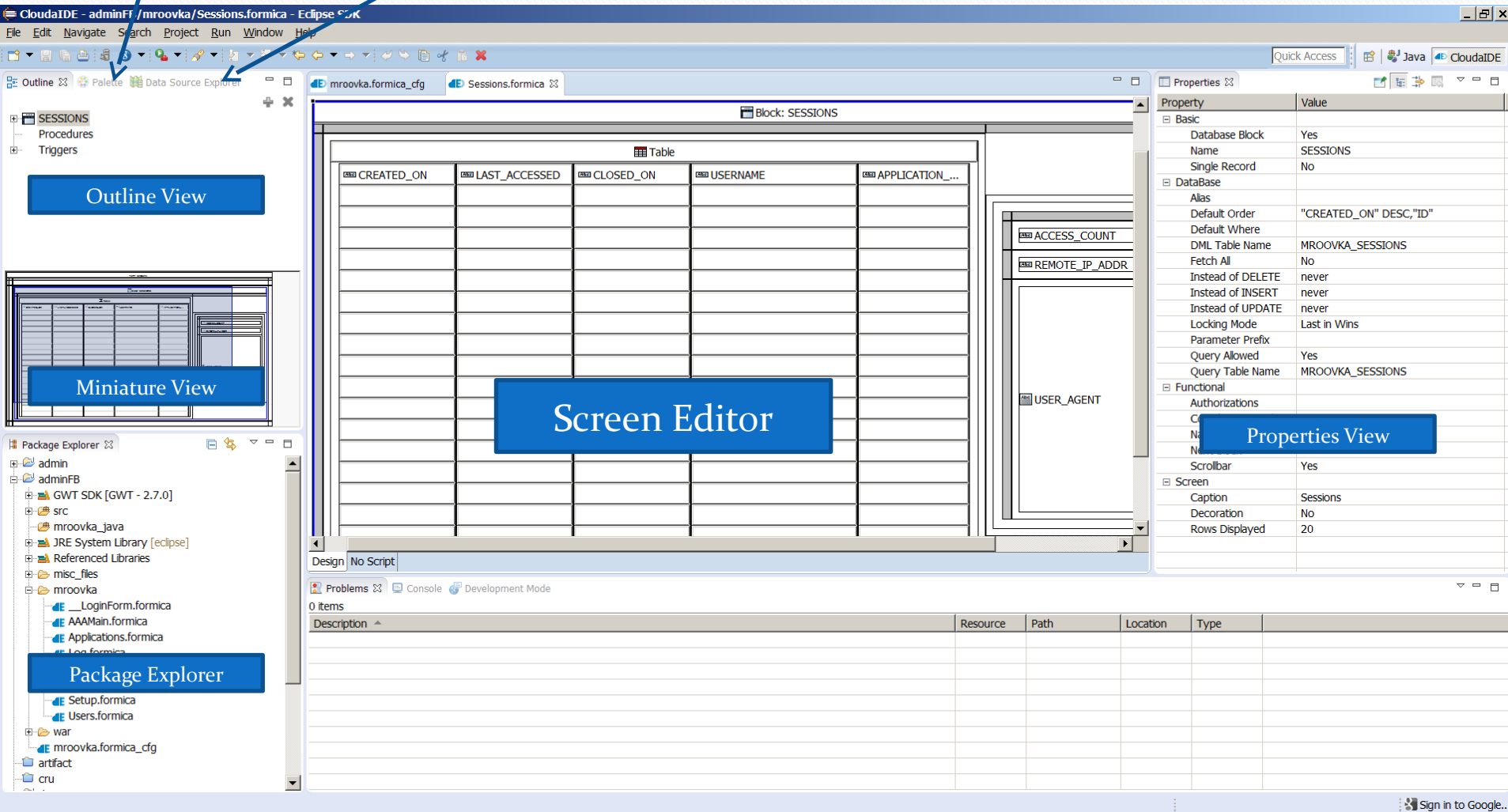
Outline View

Miniature View

Screen Editor

Properties View

Package Explorer



Fast-Track Development

To develop a screen a programmer can drag a database view or table from the Data Source Explorer and drop onto the Screen Editor

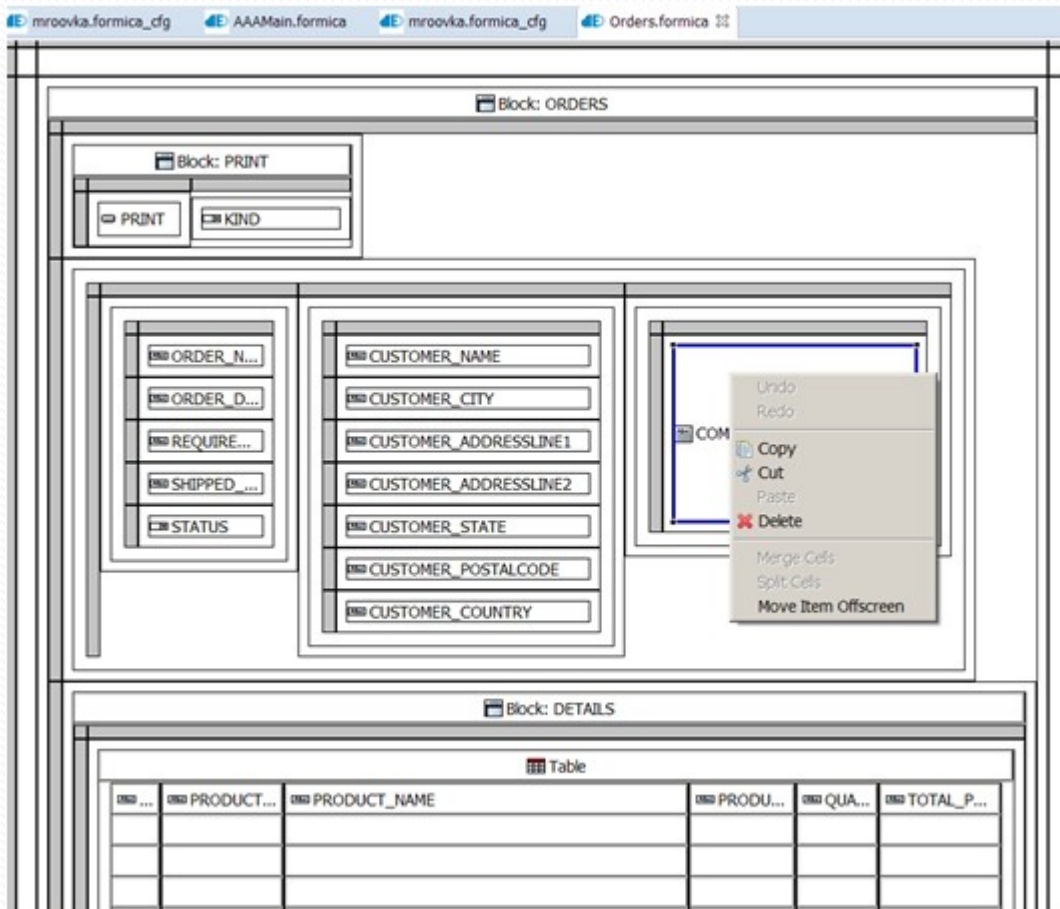
The screenshot displays the CloudaIDE interface with the following components:

- Data Source Explorer:** Shows a tree view of database objects. Under 'Views', the 'CUSTOMERS_V' view is selected, showing its columns: SALES_REP [VARCHAR(101) Nullable], SALESREP_ID [BIGINT Nullable], CREDIT_LIMIT [DECIMAL(12, 2) Nullable], COUNTRY [VARCHAR(50) Nullable], POSTALCODE [VARCHAR(15) Nullable], STATE [VARCHAR(50) Nullable], CITY [VARCHAR(50) Nullable], ADDRESSLINE2 [VARCHAR(50) Nullable], ADDRESSLINE1 [VARCHAR(50) Nullable], PHONE [VARCHAR(50) Nullable], CONTACT_FIRST_NAME [VARCHAR(50) Nullable], CONTACT_LAST_NAME [VARCHAR(50) Nullable], NAME [VARCHAR(50) Nullable], and ID [BIGINT Nullable].
- Screen Editor:** A 'Block: CUSTOMERS_V' is open, showing a 'Table' component with a grid. The columns of the table are: NAME, CONTACT_LAST_NAME, CONTACT_FIRST_NAME, PHONE, ADDRESSLINE1, ADDRESSLINE2, CITY, STATE, POSTALCODE, COUNTRY, and CREDIT_LIMIT.
- Properties View:** Shows the properties of the selected 'Table' component. The 'Basic' section includes: Database Block (Yes), Name (CUSTOMERS_V), and Single Record (No). The 'DataBase' section includes: Alias, Default Order ("NAME", "ID"), Default Where, DML Table Name (CUSTOMERS), Fetch All (No), Instead of DELETE, Instead of INSERT, Instead of UPDATE, Locking Mode (Last in Wins), Parameter Prefix, Query Allowed (Yes), and Query Table Name. The 'Functional' section includes: Authorizations, Coordination Style (Automatic), Delete Allowed (Yes), Delete confirmation text, and Insert Allowed (Yes).
- Package Explorer:** Shows the project structure for 'mroovka', including files like LoginForm.formica, AAAMain.formica, Administration.formica, Customers.formica, EmployeeList.formica, Employees.formica, Offices.formica, and Orders.formica.

Fast-Track Development

- Creates one of two selectable layouts – Table or Form
- Creates screen items based on columns and gives them properties
- Binds screen items with database columns
- Provides the programmer with default CRUD and QBE

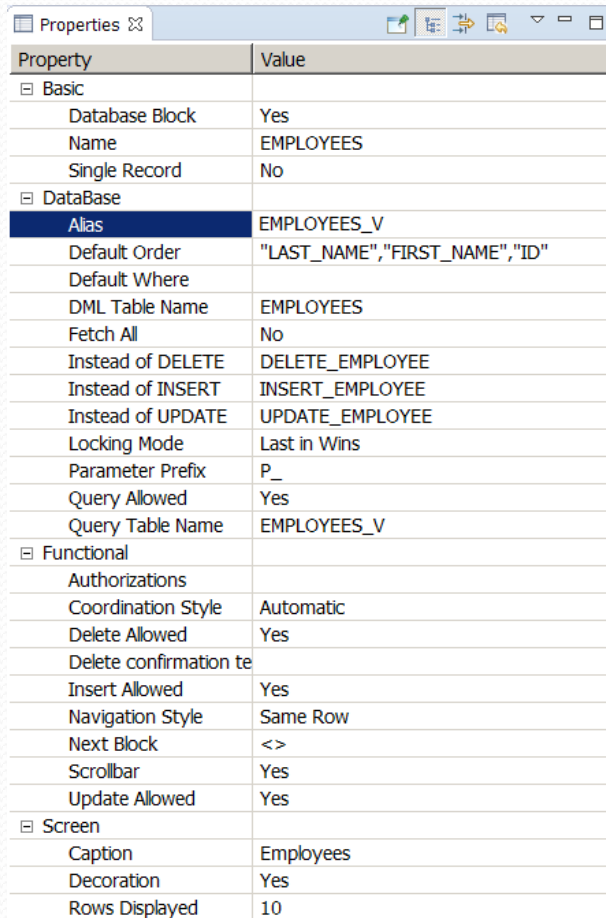
Screen Editor - Tool to sketch of screen layout



Features:

- Drag and Drop
- Undo Redo
- Cut and Paste
- Drop from Data Source Explorer

Properties View



The screenshot shows a 'Properties' window with a table of properties. The table has two columns: 'Property' and 'Value'. The properties are grouped into sections: Basic, DataBase, Functional, and Screen. The 'Alias' property under the 'DataBase' section is highlighted in blue.

Property	Value
Basic	
Database Block	Yes
Name	EMPLOYEES
Single Record	No
DataBase	
Alias	EMPLOYEES_V
Default Order	"LAST_NAME", "FIRST_NAME", "ID"
Default Where	
DML Table Name	EMPLOYEES
Fetch All	No
Instead of DELETE	DELETE_EMPLOYEE
Instead of INSERT	INSERT_EMPLOYEE
Instead of UPDATE	UPDATE_EMPLOYEE
Locking Mode	Last in Wins
Parameter Prefix	P_
Query Allowed	Yes
Query Table Name	EMPLOYEES_V
Functional	
Authorizations	
Coordination Style	Automatic
Delete Allowed	Yes
Delete confirmation te	
Insert Allowed	Yes
Navigation Style	Same Row
Next Block	<>
Scrollbar	Yes
Update Allowed	Yes
Screen	
Caption	Employees
Decoration	Yes
Rows Displayed	10

Properties View displays properties of a selected object

- Block is UI representation of a database table
- Item is a client object that corresponds to a database column

Properties View - Block

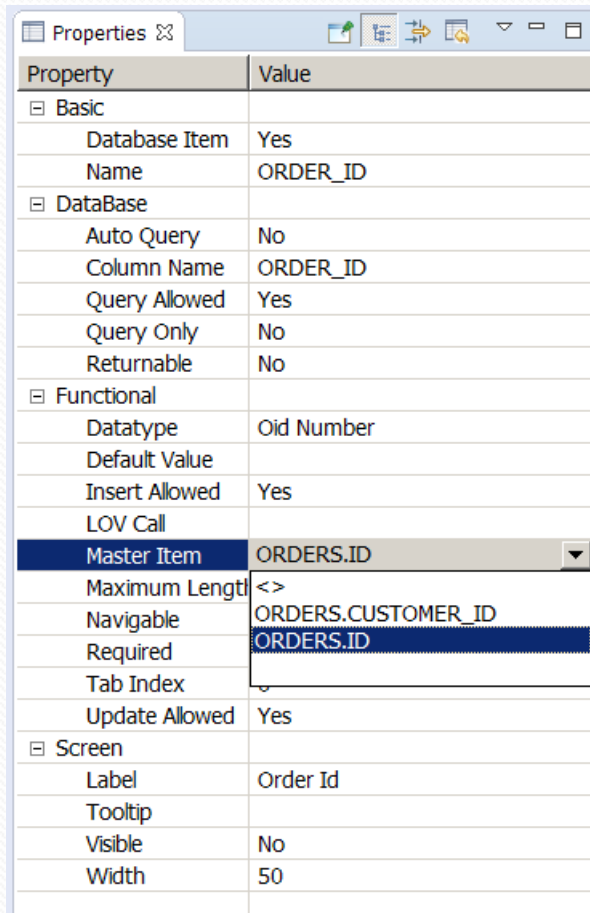
- **DML Table Name**
- **Query Table Name**

Different tables for DML and Query can be specified in order to facilitate updates on unupdatable views.

- **Instead of Insert, Instead of Update, Instead of Delete**

Programmer can define own procedures to handle DML. Updates can be replaced by application specific processing. Similar to database triggers. They give extra flexibility. For example – Instead of Delete can in fact mark a row as deleted without deleting a database row. The “Instead” procedures can also disallow performing DML operations or add extra database processing to UI events.

Properties View - Item



Property	Value
Basic	
Database Item	Yes
Name	ORDER_ID
DataBase	
Auto Query	No
Column Name	ORDER_ID
Query Allowed	Yes
Query Only	No
Returnable	No
Functional	
Datatype	Oid Number
Default Value	
Insert Allowed	Yes
LOV Call	
Master Item	ORDERS.ID
Maximum Length	<>
Navigable	ORDERS.CUSTOMER_ID
Required	ORDERS.ID
Tab Index	
Update Allowed	Yes
Screen	
Label	Order Id
Tooltip	
Visible	No
Width	50

Item is a client object that corresponds to a database column

- **Master Item** – each item can have a master item, an Item in the master block. This allows to express arbitrarily complex master-detail relationships between blocks

- ☑ Dekrety
- ☰ Obroty
- 📄 Konto
- 🔍 Szukaj Zapisów
- 🔧 Administracja

Dokumenty

Okres Dzień Rodzaj Numer Księgowal(a) Zamknięty
 Opis

REGISTER HEADER

Sprzedaż

Lp Zapłata
 Nr Rach. Konto Kontr.
 Data Wyst. NIP
 Termin Dni Nazwa
 Termin Data Nazwa alt.
 Data Sprzedaży Opis faktury
 Data Koryg. NIP Europejski
 Koryg. dokument Mies. Rej.

SALES ORDER HEADER

Data Raty	Kwota
2016-01-10	1 642 300,00
2016-01-30	100 000,00
2016-01-30	100 000,00
	1 842 300,00

INSTALLMENTS

Lp	Symbol Konta	Konto Nazwa	Opis	PKWiU	J.m.	Ilość	Cena	Stawka VAT
1	010-00-1-99	Środki trwałe	Prawa własności do systemu eksploatacyj				1 500 000,00	22%
2	201-1-3-1-00312	Poszewiecka Irena Olsztyn	dossosa	II	I	1,000	3 000,00	23%
2	201-1-3-1-00312	Poszewiecka Irena Olsztyn	dossosa	II	I	1,000	7 000,00	23%

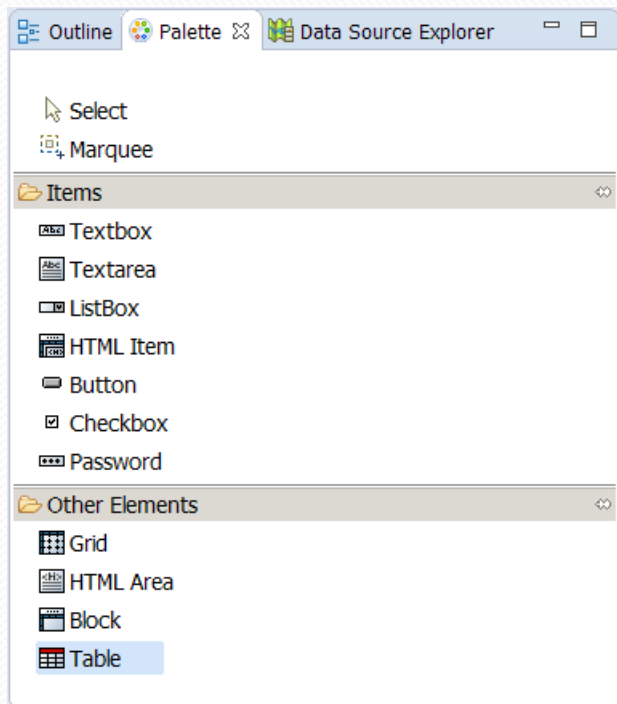
ORDER DETAILS

	Brutto	Netto	VAT
Faktura	1 842 300,00	1 510 000,00	332 300,00
Rejestr	1 846 853,46	1 513 702,00	333 151,46

Properties View – Item

- **Query Only** – if set to yes then the item belongs only to Query table (not to DML table). Because of this Query Only Item does not take part in DML operations.
- **Returnable** – Similar to SQL return column. After any DML operation this item is returned to the client.
- **Tab Index** – programmer can statically arrange any sequence of cursor navigation. The programmer can dynamically set next navigation item and also force cursor navigation using:
`set_item_property(next_item,'BLOCK_NAME.ITEM_NAME');`
- **LOV call** Programmer can specify the name of a Form with parameters. This turns Textbox into a List Item.

Tool Palette

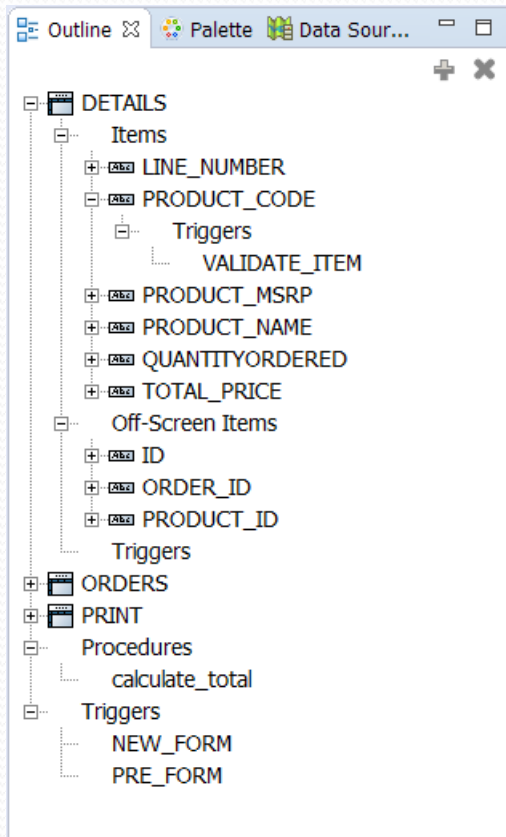


Tool Palette serves to create/select screen elements

It has two layout elements:

- **Grid** – to place other elements in HTML table
- **Table** – to place other elements in a table of horizontal rows (spreadsheet like)

Outline View



Tree structure of the form. Using it the programmer can see all the data elements of the form. Blocks, items and code

Through Outline View the programmer has also access to non UI elements:

- **Off-Screen items** – items that are never displayed
- **Triggers** – pieces of code reacting to client events
- **Procedures**

- Isolates the programmer from the complexity of asynchronous nature of screen interaction and AJAX calls
- No callbacks
- Close to PL/SQL. Key differences:
 - no SQL
 - case sensitive
 - datatypes

MT Triggers Language

Data Types of MT

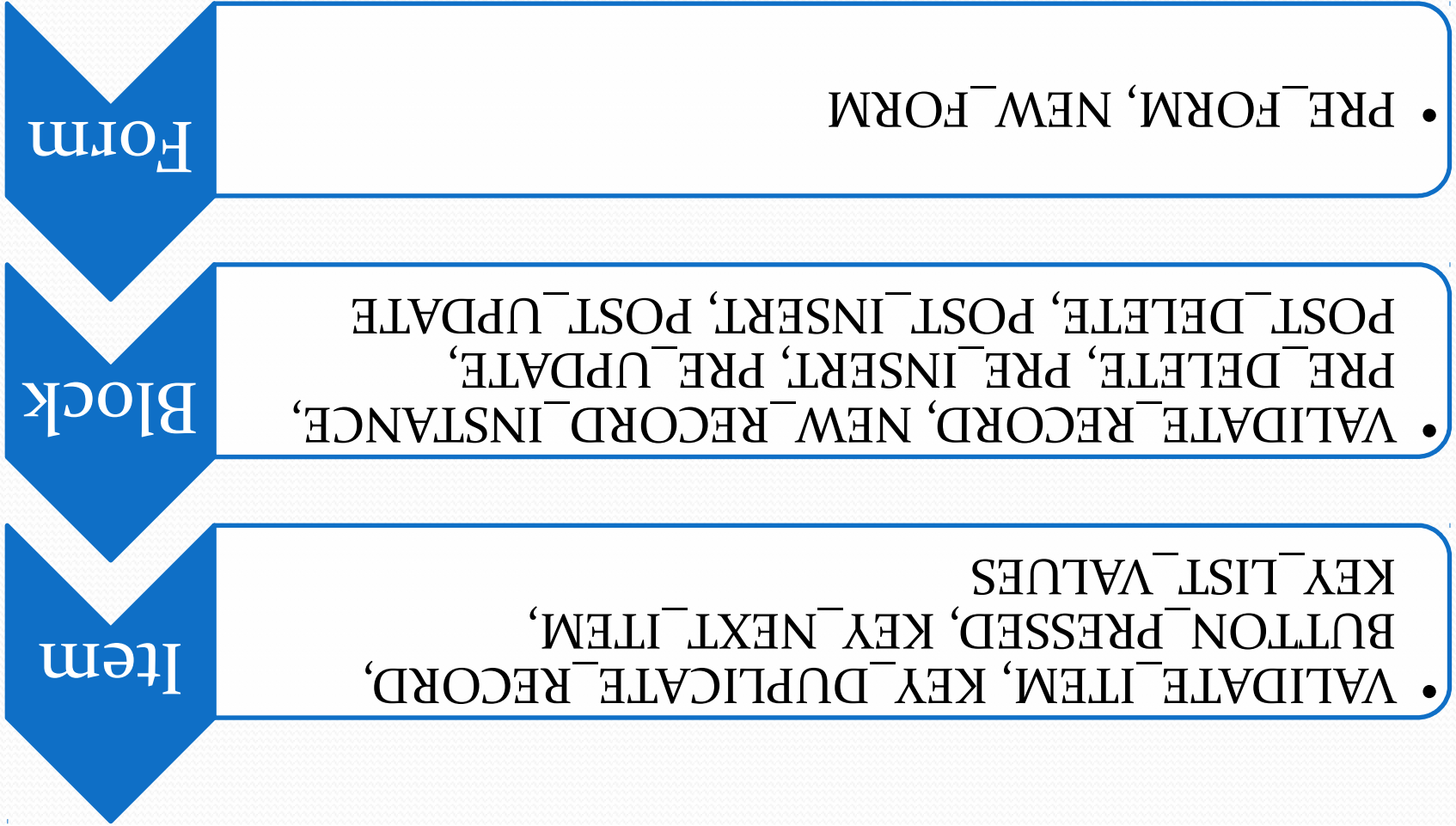
- text
- number
- date (timestamp)
- oidn – encrypted number
- etext – encrypted text

Main constructs

- Procedures
- Triggers
- Loops
- Conditionals
- Exceptions

Trigger Types

Depending on the declaration level



Example trigger

```
-- NEW_FORM
begin

  set_title('CloudaIDE user manager');
  if url_parameter('p_user') is not null then
    e_mail_verification;
    return;
  end if;
  set_block_property('THANKS', visible, false);
  if url_parameter('email_user') is not null then
    set_block_property('USER', visible, false);
    set_block_property('PASSWORD_EMAIL', visible, false);
    go_item('PASSWORD_CHANGE.PASSWORD');
    return;
  end if;
  set_block_property('PASSWORD_CHANGE', visible, false);
exception
  when others then
    message_error(error_message);
end;
```

Example procedure

```
procedure before_delete(p_id oidn) is
begin
  if p_id is null then
    return;
  end if;
  mr_dictionary_p.before_delete_test(p_id);
  confirm('Do you want to remove ?', true);
  if not success then
    return;
  end if;
  mr_dictionary_p.prune_bushes(p_id);
exception
  when others then
    message_error(error_message);
end;
```

What the client-side code is?

- Single page architecture - everything is downloaded as a single page
- Javascript:
 - Creates HTML
 - Reacts for events
 - Executes AJAX to communicate with the server
- Downloaded once for a compilation
- Downloaded incrementally – minimizing initial download

What the client-side code is?

- This way screen layout and behaviour is supplied to the client only once.
- The main network traffic after initial download is data.
- The layout and behaviour code is cached on the client (until next release)

Code splitting

- The application is built of forms
- Following this, code also is split into forms
- Code delivery to the client is split into engine (common functionality) initial download and form (on demand) downloads. This helps to minimize network traffic because no monolithic initial download is carried out and unused code never gets to a client
- Code is loaded in gzipped form (most browsers handle this)

Demo Application code splitting report

Google web toolkit

Compile report: Permutation 0

Full code size

432 258 Bytes

[Report](#)

Initial download size

300 184 Bytes

[Report](#)

Left over code

52 366 Bytes

[Report](#)

Split Points

#	Location	Size (Bytes)	% of total
1	@pl.mroovka.ap.client.AAAMain\$1::run()	26 891	6,2%
2	@pl.mroovka.ap.client.__LoginForm\$1::run()	8 021	1,9%
3	@pl.mroovka.ap.client.Administration\$1::run()	3 522	0,8%
4	@pl.mroovka.ap.client.Orders\$1::run()	18 827	4,4%
5	@pl.mroovka.ap.client.Sales\$1::run()	4 563	1,1%
6	@pl.mroovka.ap.client.Customers\$1::run()	4 565	1,1%
7	@pl.mroovka.ap.client.Products\$1::run()	5 215	1,2%
8	@pl.mroovka.ap.client.EmployeeList\$1::run()	885	0,2%
9	@pl.mroovka.ap.client.Offices\$1::run()	3 405	0,8%
10	@pl.mroovka.ap.client.Employees\$1::run()	3 814	0,9%

Report integration

- ClouderaIDE is integrated with BIRT – Business Intelligence and Reporting Tool
- ClouderaIDE can call a BIRT report using:
 - Parameters
 - SQL where phrase of a query last executed on a selected database block

Report integration

```
-- BUTTON_PRESSED
begin
  post;
  if not success then
    return;
  end if;
  if :print.kind = 'D' then
    if :orders.id is null then
      message_error('Please select order to print');
      return;
    else
      run_report(mv_order, :orders.id);
    end if;
  else
    run_report(mv_order, null) criteria_block orders;
  end if;
exception
  when others then
    message_error(error_message);
end;
```

- In the last case a programmer can use the clause in a query in the report
- The execution of the report is protected by a checksum and can be carried out only once for a call

Report integration - sequence of events




• The client asks the server for a URL



• The server prepares the URL and writes its sequence number and MD5 hash to the database



• The client receives the URL and sends it to the report server



• Reports server calculates the MD5 hash of the URL and checks if there is match between sequence number and the MD5 hash of the URL. If there is not - HTML error 404 is reported



• Reports server deletes the URLs hash in order to disable recurring reports call (from the browser history)



• Reports server executes the report

Lists of Values

The image displays the SAP NetWeaver IDE interface, illustrating the configuration of a List of Values (LOV) call for a customer name field. The main window shows the 'Orders.formica' form with a 'Block: ORDERS' containing a 'Block: PRINT' and a 'Block: CUSTOMERS_V'. The 'CUSTOMERS_V' block contains a table with a 'NAME' column and a 'SALESREP_NAME' column. The 'SALESREP_NAME' column is highlighted with a blue box, and a red box highlights the 'LOV Call' property in the Properties window, which is set to 'Customers'.

The Properties window for the 'CUSTOMERS_V' block shows the following configuration:

Property	Value
Basic	
Database Item	Yes
Name	SALESREP_NAME
DataBase	
Auto Query	No
Column Name	SALES_REP
Ignore Case	Yes
Query Allowed	Yes
Query Only	Yes
Returnable	No
Functional	
Case Conversion	None
Datatype	Text
Default Value	
Format Mask	
Insert Allowed	Yes
LOV Call	Customers
Master Item	<>
Matching	Contains
Navigable	Yes
Required	No
Tab Index	0
Update Allowed	Yes
Screen	
Label	Sales Rep
Tooltip	
Visible	Yes
Width	30

The Properties window for the 'Orders.formica' form shows the following configuration for the 'CUSTOMERS_V' block:

Property	Value
Basic	
Database Item	Yes
Name	CUSTOMER_NAME
DataBase	
Auto Query	No
Column Name	CUSTOMER_NAME
Ignore Case	Yes
Query Allowed	Yes
Query Only	Yes
Returnable	No
Functional	
Case Conversion	None
Datatype	Text
Default Value	
Format Mask	
Insert Allowed	Yes
LOV Call	Customers
Master Item	<>
Matching	Contains
Navigable	Yes
Required	Yes
Tab Index	5

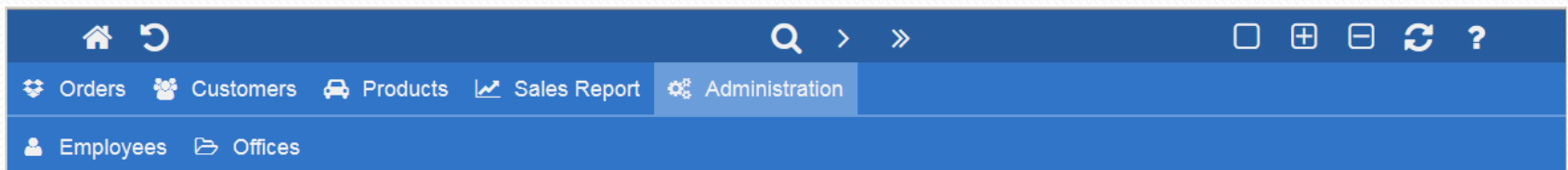
Lists of Values

Realized as forms

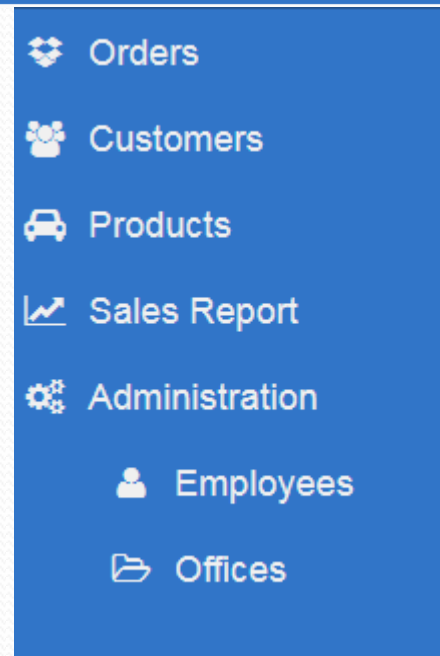
- Items of called form bound to Calling form items by a naming convention
- Automatic (in the background) selection of a list element (item List validable property)
- Possibility to enter missing list elements on the fly
- Using the list to drill/navigate data – after the selection is made

Menus

- Come in two flavours:
 - Vertical



- Horizontal
- Application – wide choice



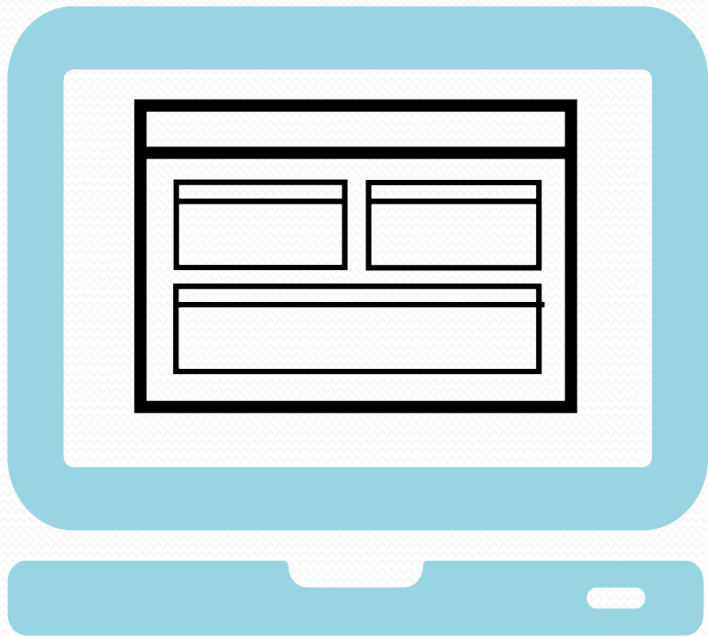
Menus

- Forms create stack
- Each Form can contribute options to the menu
- These options stay in the menu, until the form is closed
- If an option belonging to a particular form is clicked, all forms above the form are closed, with one exception
- An option resulting in opening a form stores reference to the opened form
- When clicked again the menu system closes all forms above the called form and displays the form opened by this option (not the one that has opened it). It gives an effect similar to a breadcrumb navigation

Built-in Security features

- Built-in authentication:
 - Table
 - Database Account
 - CAS
- Authorizations – definable by the administrator
- Protection against injection
- Object protection

Protection against injection



Protection against injection

- The application server knows application metadata. No direct SQL statements, phrases and procedure calls are passed to the server. Everyone of those are handled indirectly against application metadata

Object protection

- CloudaIDE uses encrypted numbers and texts. The encrypted data contains information about the source of this data. The system analyses the graph of possible assignments. The data of oidn and etext types is sent to a client in encrypted form. No other processing than assignments of this data is possible. When the server receives this data back from the client it checks whether it conforms to the assignment graph. Whether the target entity can be reached by the source. If not, then security exception is reported, otherwise the data after decryption goes to further processing.

Locks

- System uses connection pooling. Because of this every server call (default block DML, or a Stored Procedure call) constitutes a separate transaction
- Locking within the Updates are of two kinds:
 - Last in wins (no locking)
 - Optimistic locking



More information on:

cloudaide.org



QUESTIONS

?