1C:ENTERPRISE 8.3

Hello, 1C

Rapid application development tutorial

Contents

Hello, 1C rapid application development tutorial demonstrates the most basic features of the cutting-edge 1C:Enterprise 8 platform. You will understand the application development process and see that it is easy to learn and develop applications powered by 1C:Enterprise 8.

Having downloaded 1C:Enterprise 8.3 training version for free, anyone can repeat the application development process on his or her computer.

The third edition includes the description of new features implemented in 1C:Enterprise 8.3, which you can find in the Cross-platform design chapter on page 125.

What do you think of this book? You are welcome to send us feedback as well as ask questions and get support on the Studying 1C:Enterprise platform forum.

Hello, 1C rapid application development tutorial, version 3.

Russian text: V. Rybalka

Translation: T. Bugaevsky

Correction: A. Novikov, S. Polikarpov

Introduction

The main business of 1C Company (see page 154) is development of business management and accounting software, as well as educational applications development and publishing, and software distribution. Over 1,000,000 companies and over 4,000,000 employees are using business software powered by 1C:Enterprise (see page 151). Market requirements for developers and qualified users capable operating 1C:Enterprise grows up constantly.

The main goal of this tutorial is to demonstrate the basic features of the 1C:Enterprise 8 platform and its software engineering technology. There will be no secrets except for one: within a few minutes, using almost only a mouse, you can create a fully functional application to keep personal finance records that is compatible with different DBMS (database management systems), Windows, Linux, web browsers, and even iOS and Android mobile devises.

Perhaps you are already familiar with one of high-level programming language, for example Java, C++, or Delphi. Many books and programming training courses start with a simple task such as creating a program that would display a simple text on a screen, for example, "Hello, world!"

In fact, what you are going to do in this tutorial is "Hello, world!" powered by 1C:Enterprise. Indeed, the functionality of the demo application is going to be much broader than the simple output of a phrase on a screen. However, considering all capabilities of the 1C:Enterprise 8 platform, what you are going to develop is precisely "Hello, world!" in the world of 1C:Enterprise.

If you have everything you need?

Before you begin, make sure that you have everything you need to get started. That is make sure that the 1C:Enterprise platform is installed.

Check that when you click **Start**, and then point to **All Programs**, there is **1C Enterprise 8 (training version)** inside.

If this application icon is not in the menu, the 1C:Enterprise platform must be installed.

If you do not have the installer, you can download the 1C:Enterprise 8 (training version) for free on 1C:Developer Network. After extracting from the archive, run **setup.exe**.

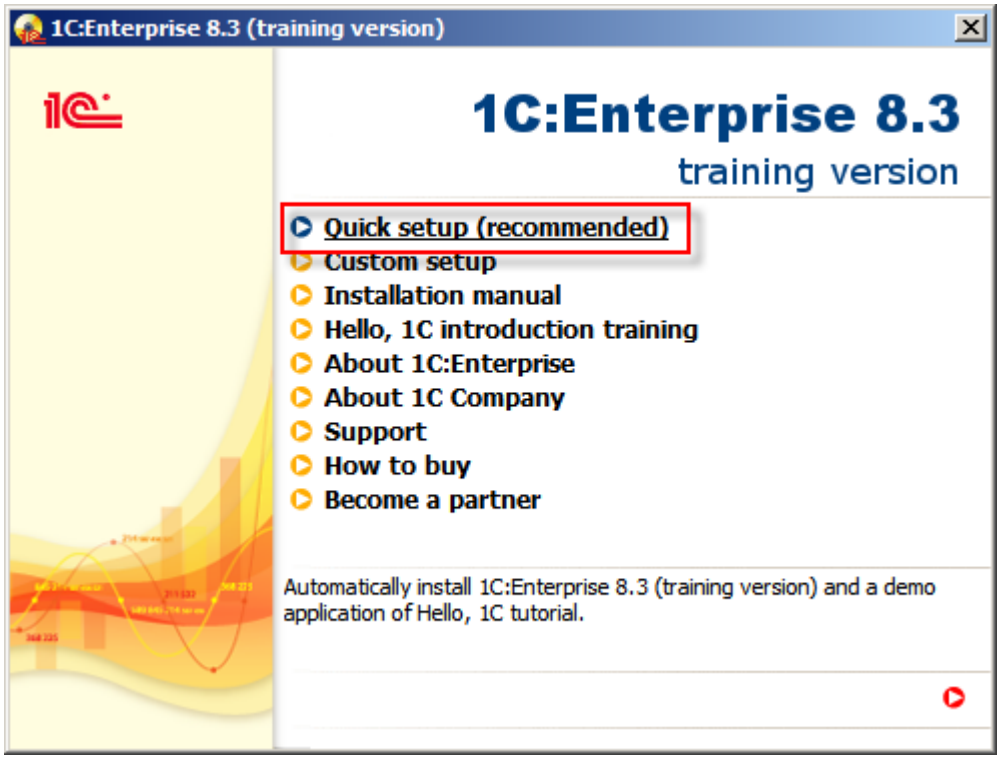

Figure 1-1. Installing 1C:Enterprise 8 (training version)

The installation procedure is simple. Agree with all default options, and continue clicking **Next** until the installation is complete.

Infobase

1C:Enterprise is not a universal IDE. You cannot create *any type* of program using 1C:Enterprise. It is designed for automation of business and individuals. For this reason, many concepts are already embedded into the heart of 1C:Enterprise, its technological platform.

Anywhere where there is 1C:Enterprise, there will be a technological platform. This provides uniformity so that the development and modification technology, as well as structure of 1C:Enterprise applications are always the same.

The major technological concept used in 1C:Enterprise is **Infobase**. Roughly, each infobase is a single 1C:Enterprise application. The unique characteristic is that each infobase contains not only the data the user works with, but also the program, which is named applied solution, that is executed by the platform. For example: 1C:AccountingSuite or 1C:Small Business both are applied solutions.

Thus, when you need 1C:AccountingSuite application, create an infobase with 1C:AccountingSuite applied solution and keep your finances accounting data in this infobase. You can create multiple infobases for keeping information for different companies with the same 1C:AccountingSuite applied solution, but different data. If you will need 1C:Small Business, you will have to create an infobase with another applied solution, 1C:Small Business. In that infobase you will keep goods production accounting data rather than finances accounting in 1C:AccountingSuite infobases.

Thus, if you have an infobase, you have everything you need for work: data and an applied that knows how to manage it.

The creation of any 1C:Enterprise applied solution starts with the creation of an infobase, where that applied solution and data, managed by it, will be stored.

Getting started. Start 1C:Enterprise: click **Start**, and then point to **All Programs**. Point to **1C Enterprise 8 (training version)**, and then click **1C Enterprise (training version)**.

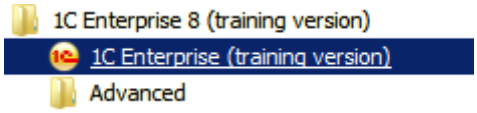

Figure 1-2. Starting 1C:Enterprise (training version)

The first thing 1C:Enterprise will do is open a list of available infobases. If you installed 1C:Enterprise (training version) that includes this tutorial, you can find the **Hello, 1C (demo)** infobase there. In this tutorial, you will create the same infobase step-by-step. Click **Yes** to add a new infobase to the list.

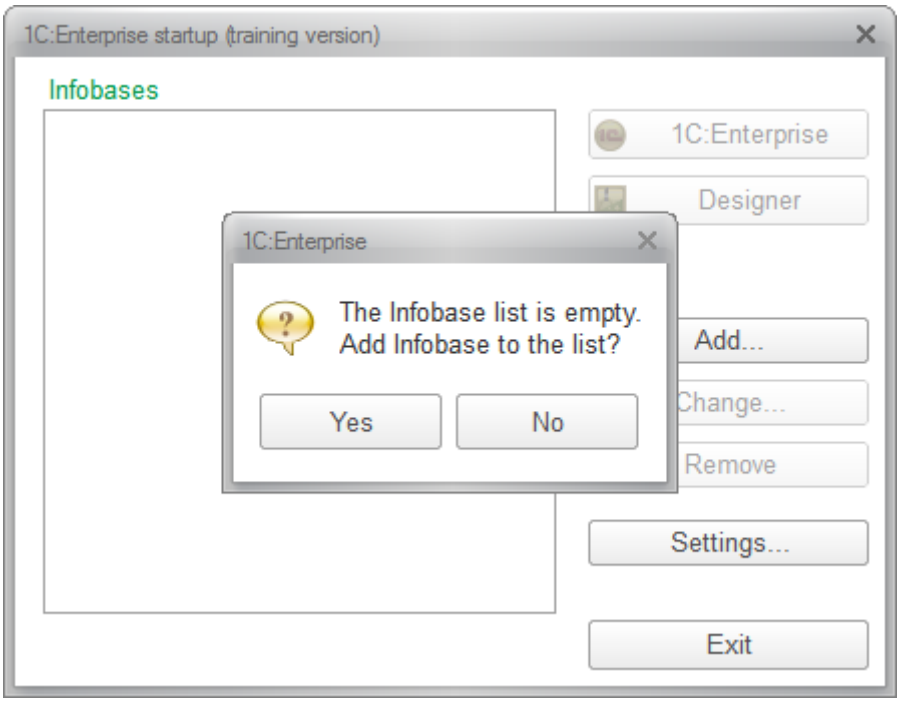

Figure 1-3. Adding a new infobase to the empty list

Notice: If you already have infobases in the list, there will be no suggestion to add a first infobase. In this case click **Add...** to add an infobase.

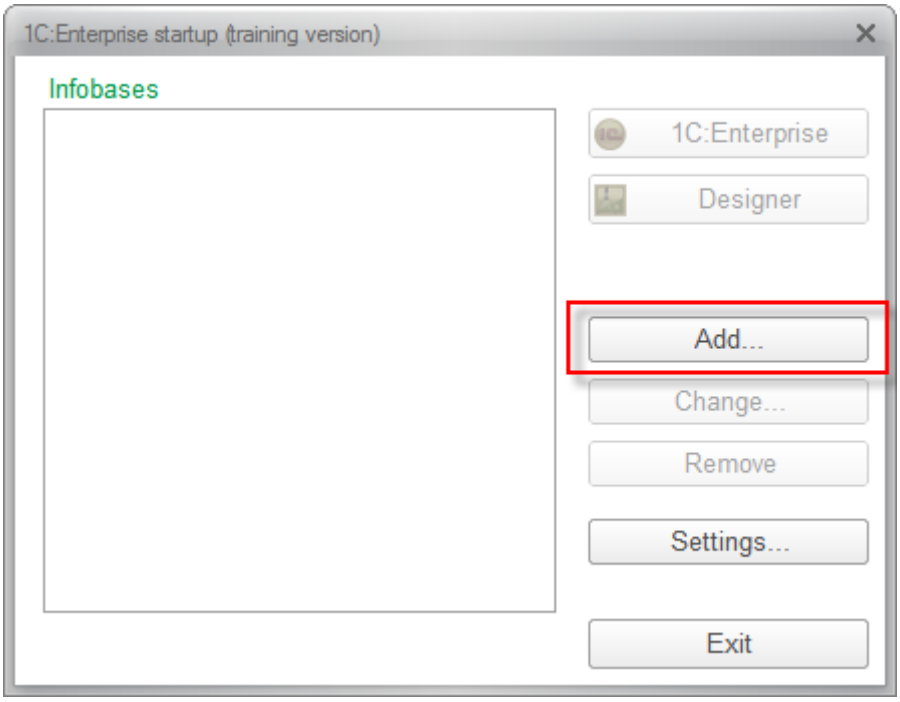

Figure 1-4. Adding a new infobase

Now the wizard will ask what you would like to add to the list of infobases: a completely new infobase, which does not exist yet, or an existing infobase, for example, the one that already present on a local network server.

The default option is to create a new infobase, and this is what currently required. Thus click **Next**.

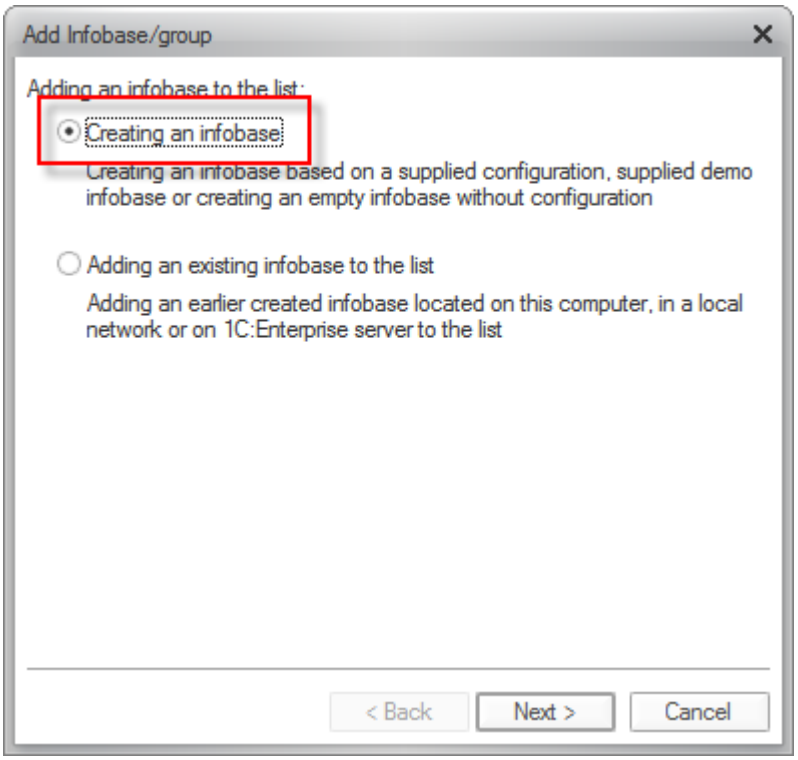

Figure 1-5. Creating a new infobase

There are two ways to create a new infobase: an empty infobase that will contain neither data nor applied solution, or use a template that can contain an applied solution and maybe even demo data. To create an infobase with preset application you can use first option on the next page of the infobase creation wizard.

Since you have not installed any template yet, the wizard will set an option to create a new empty infobase as default and you can continue by clicking **Next**.

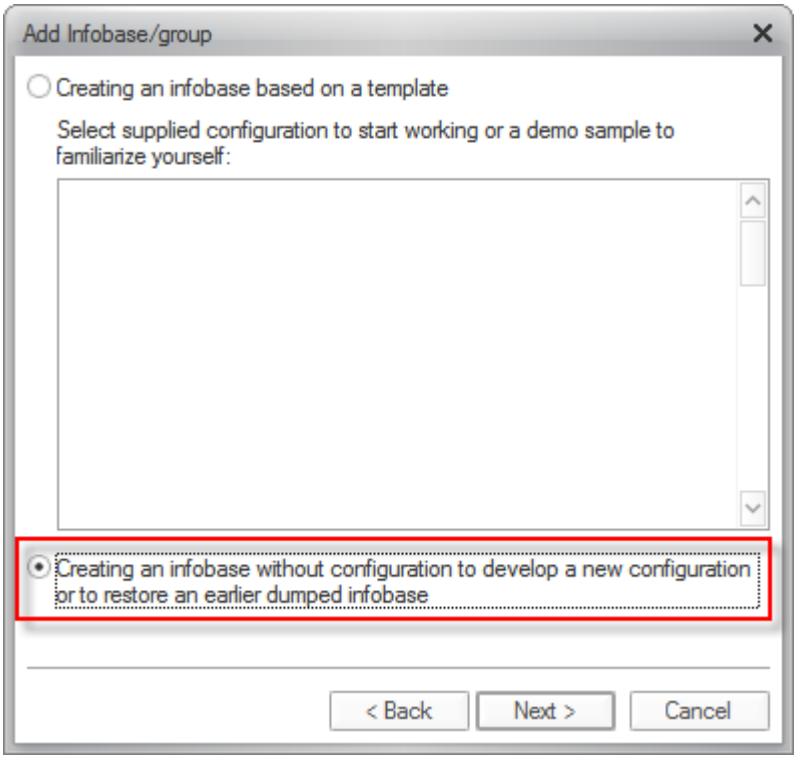

Figure 1-6. Creating an infobase without an applied solution

The wizard will now ask for a name that you would like to give to the new infobase. The infobase name does not affect anything so any name can be used, but for referring matter name it **Hello, 1C**. This name will be displayed in the infobase list at startup. Click **Next**.

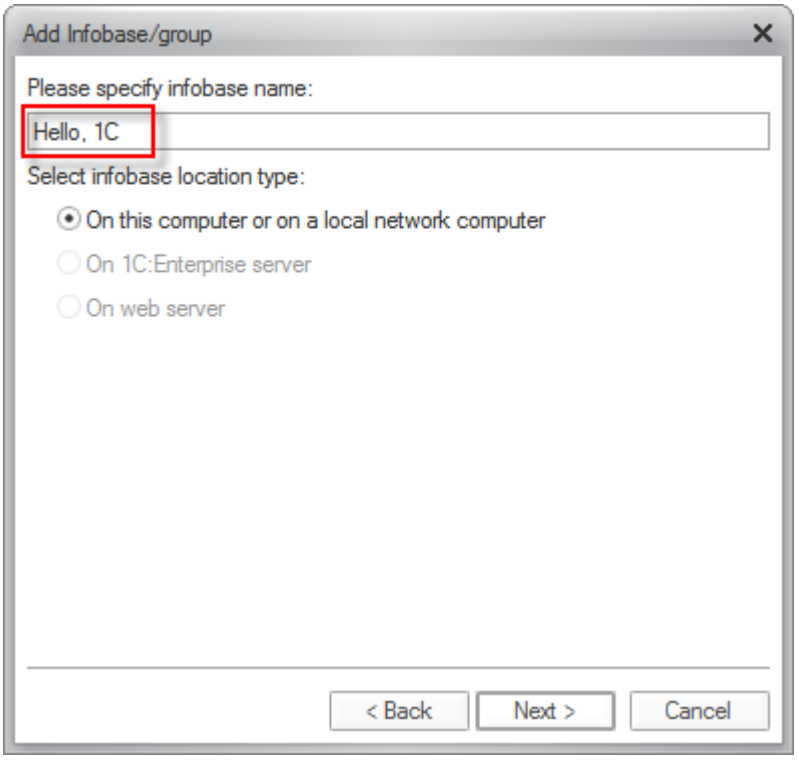

Figure 1-7. Specifying the infobase name

Finally, the wizard will ask you for a folder where the infobase will be stored. Default value is the user profile. Enter **C:\hello_1c** in the **Infobase directory** field. If you have insufficient space on your system drive, you can specify a different location by clicking Select .... However, in most cases you can accept the default value and click **Next**.

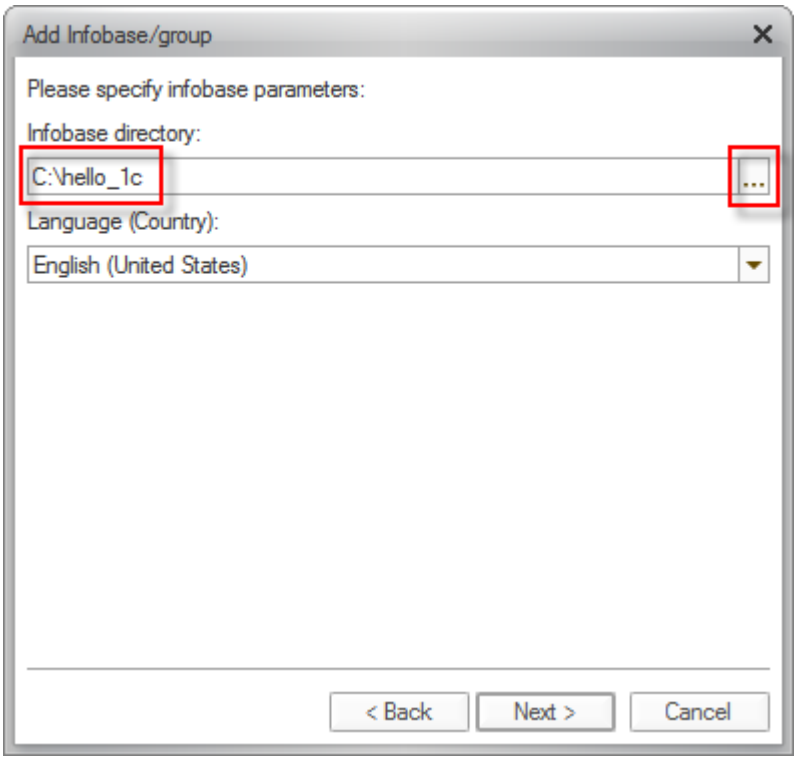

Figure 1-8. Specifying the infobase location

On the next page you need not to change anything, click **Ready**.

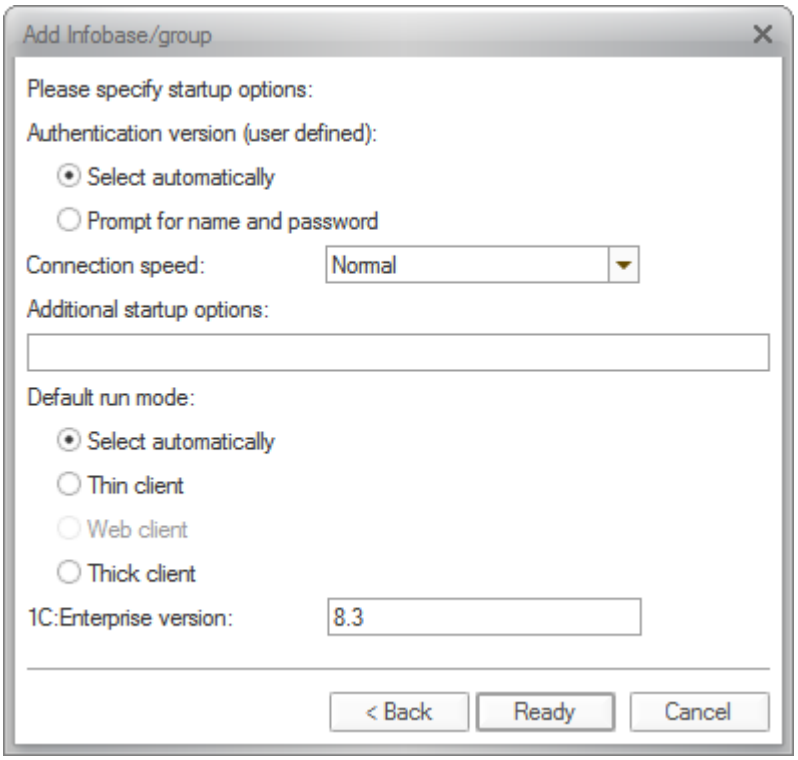

Figure 1-9. The infobase is ready

As a result you will see the new infobase named **Hello, 1C** in the list of infobases. Click **Designer** to start developing the **Hello, 1C** application.

Designer mode is the developer mode of running 1C:Enterprise platform that is used to create or modify applications that already exist in infobases.

1C:Enterprise mode is a user mode for working with the data stored in infobases.

In this tutorial you are a developer, therefore click **Designer**.

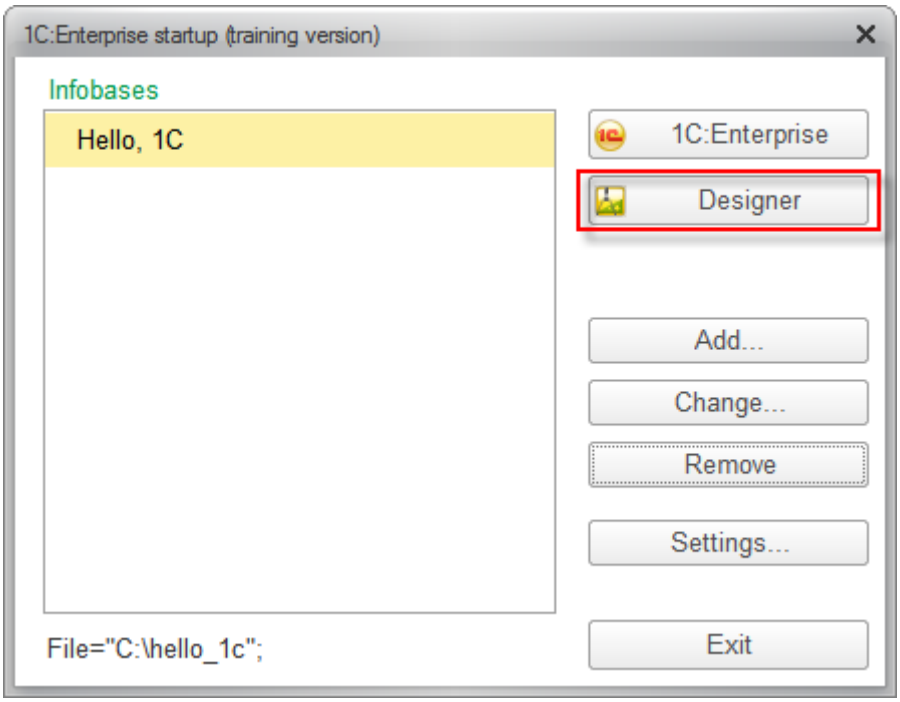

Figure 1-10. Running the Designer mode

Start "programming"

It is no accident that word "programming" is used within quotation marks in the title. Programming itself in its common use, writing code, is an essential part of 1C:Enterprise, but not the number one.

Since 1C:Enterprise, as it was said in Introduction chapter on page 4, is a problem-oriented platform, it hides from the developer many boring routine actions. All 1C:Enterprise applications are built using the same design methods, each application is put together from ready-to-use building blocks. The number of building blocks is limited, the platform has implemented functions of all of them and how they interact between each other.

Therefore, a developer simply needs to add necessary building blocks to the infobase and it will work right away. The platform itself will ensure the proper functioning of building blocks.

Of course, the number of default functions is quite limited. In practice, everything is much more interesting and full of surprises. For this, there is a built-in script language, named **1C:Enterprise script**, and a built-in query language, named **1C:Enterprise query language**. With them, you can adjust the behavior of building blocks, define algorithms of interaction between building blocks, implement your own data processing algorithms, etc.

As it was said in Introduction chapter on page 4, this tutorial is about writing "Hello, world!" using 1C:Enterprise. Therefore, you are going to use less of script and use close to no query language. The use of script and query language in 1C:Enterprise is a complicated subject that requires a separate book.

The current task is to create a simple application from building blocks. In other words, to demonstrate basics of the development process. Adding bells and whistles, improvement, and modification can be done later, if you are interested and have time to do it. However, it is still important that even in such a skeleton form, the application will be fully functional with minimum efforts.

Notice: To make sure, you can create an application with similar functions using any other universal IDE, and compare the necessary knowledge and time that you spent.

Now, get back to building blocks. Since purposes of 1C:Enterprise applications are predefined, building blocks are not abstract, but problem-oriented and related to practical entities. For example there are building blocks of catalogs and documents classes, which business and individuals use in daily work.

In 1C:Enterprise, these building blocks are called configuration objects. All configuration objects are grouped into a tree. Thus, by looking at the **Configuration** object tree, you can observe the architecture of any application. You can quickly locate the object and learn its properties.

Now the configuration is opened after you clicked **Designer** in the end of previous chapter. To see the **Configuration** object tree, you need to click **Open configuration** .

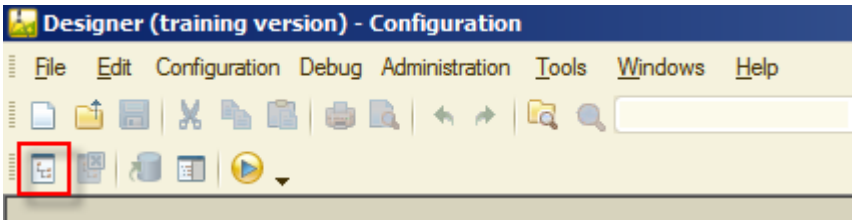

Figure 2-1. Opening the Configuration object tree

For now this tree is empty, it contains only top-level nodes, which can be found in any 1C:Enterprise infobase.

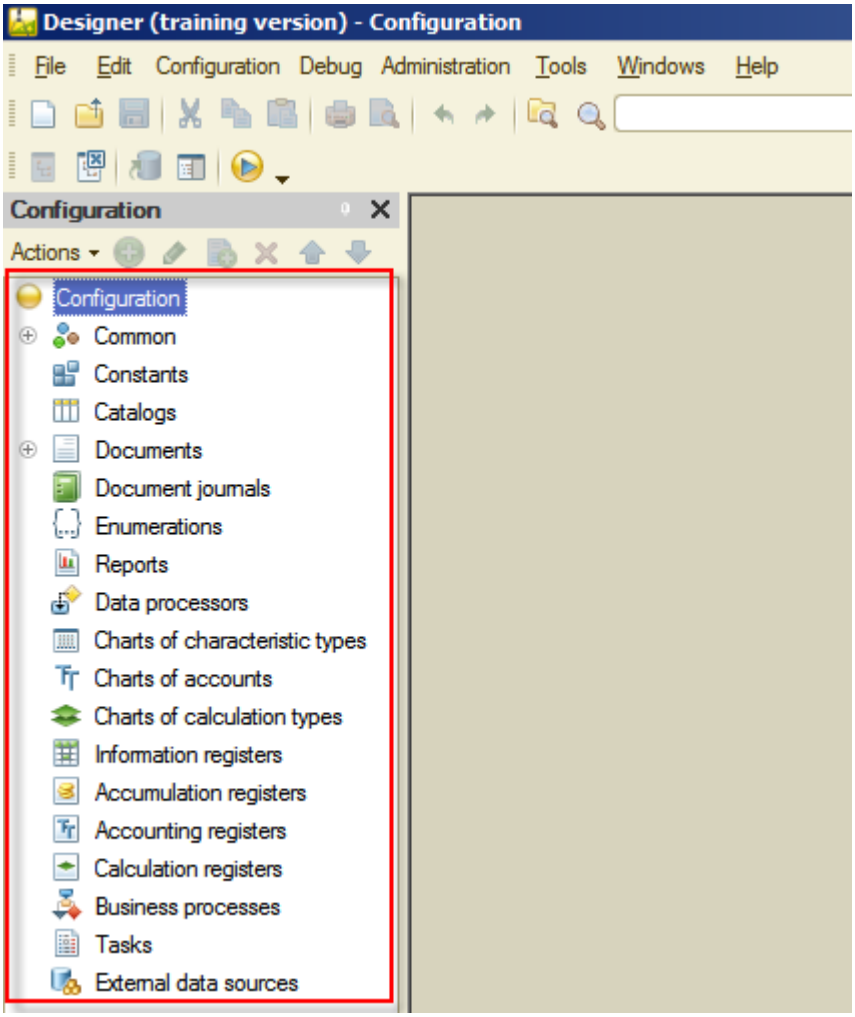

Figure 2-2. the Configuration object tree

The things that you will do next are adding configuration objects that are the basis of the **Hello, 1C** application. Along the way you do it, there will be some "bells and whistles" added, but without of going into much detail.

What configuration objects shall you add? It depends on the purpose of the automation, which is reflected in project requirements.

In this tutorial, according to the project requirements you will create a simple personal CRM (Customer Relationship Management) application. On the one hand, this application will store information about all your friends and acquaintances, in other words, maintain a contacts database. On the other hand, it will track different kinds of events, both past and future. At the same time, it will be able to track financial activities: receipts and expenditures related to both your friends and events in your live. In

addition, there will be added some features to make use of the application simple and easy.

That is it, nothing too complicated.

Subsystems

The first step is to define subsystems. The function of a subsystem is to group configuration objects by their business purpose. Using subsystems in the future, for example, will allow you easily create the application interface. An application interface is how an application provides a user with an access to its functionality.

First, add several subsystems. Later, when adding new configuration objects, you will include them to these subsystems.

If observe entire the project requirements that were described in Start "programming" chapter on page 4, you can see that there are three separated business purpose groups:

- Processes related to people.
- Processes related to events.
- Processes related to financial activities.

That is why you will create three subsystems: **Contacts**, **Events**, and **Finances**.

There is a common way to add configuration objects. Right click the branch of the configuration tree where the necessary objects will be located, and from the context menu choose **Add** (Ins) command.

The **Subsystems** branch is located in the **Common** group of the **Configuration** tree. Add a subsystem.

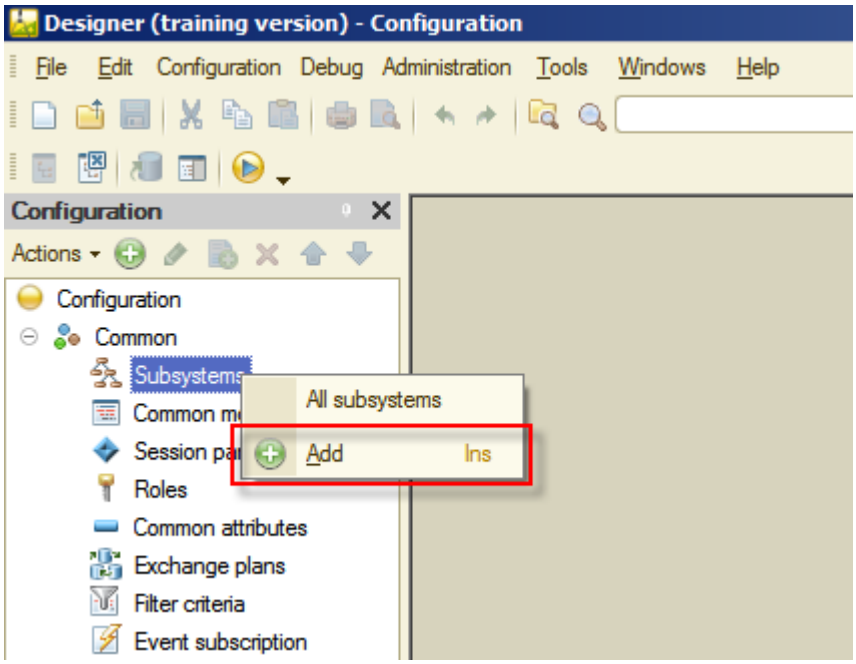

Figure 3-1. Adding a subsystem

On the right side a new configuration object editor will open. This window contains properties of the new subsystem. Name the first subsystem **Contacts**. 1C:Enterprise will automatically add a value to **Synonym** property once you press the **Enter** or the **Tab** keys being positioned in the **Name** property. For now, there is no need to change anything else in this window, so click **Close**.

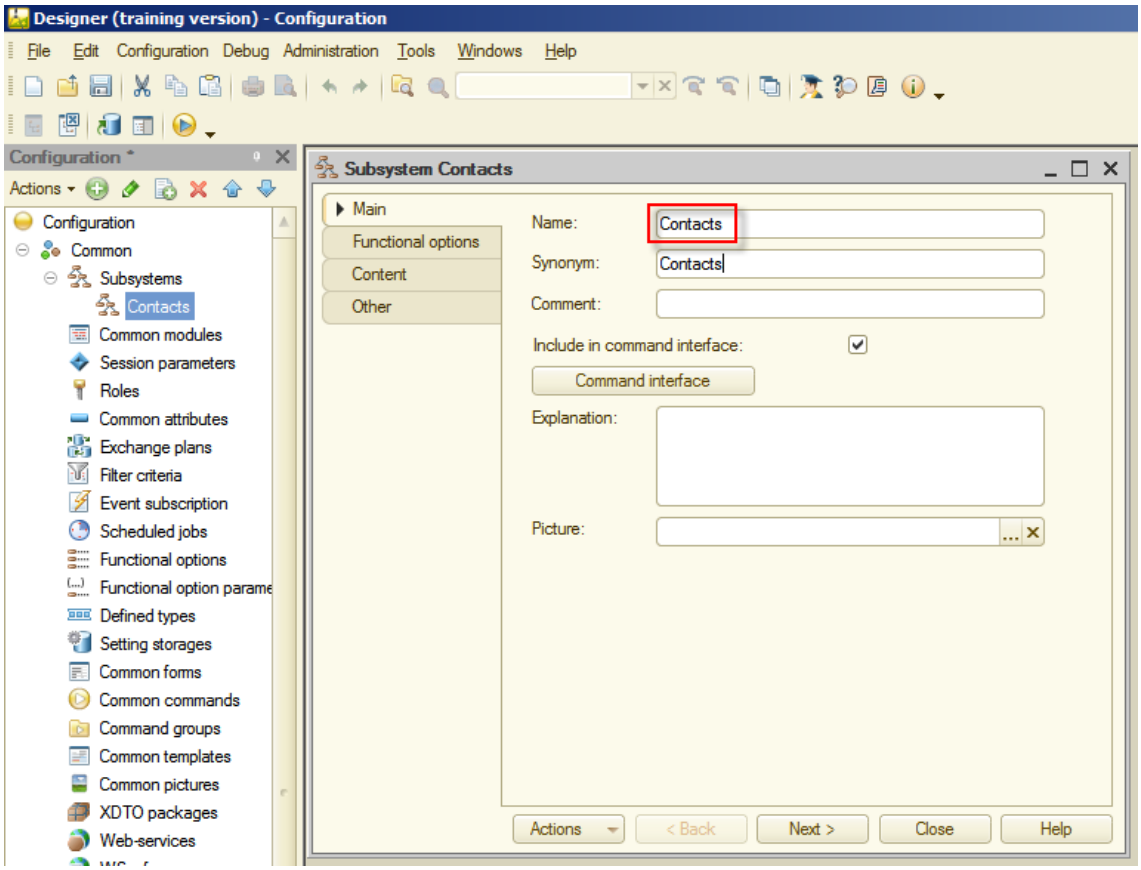

Figure 3-2. Creating the Contacts subsystem

In the same way, create other two subsystems: **Events** and **Finances**. As a result, you will have the following tree:

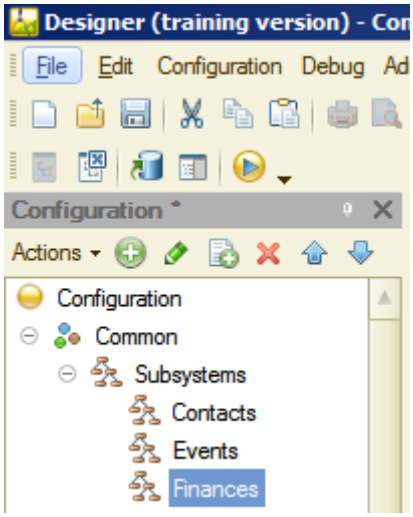

Figure 3-3. The Configuration object tree with newly added subsystems

You added service building blocks. Next, you will add business building blocks.

Give a name to the applied solution. right now, it has a generic name **Configuration**. In addition, at the same time, learn one more way to edit object properties, which is the **Properties** window. To open this window, double-click the top row of the **Configuration** object tree or right-click on the top, and then click **Properties** (Alt+Enter).

Now, in the same way as you did for subsystems, type in the **Name** property the name of applied solution**: Hello1C**. Then edit the automatically generated value of the **Synonym** property to **Hello, 1C**.

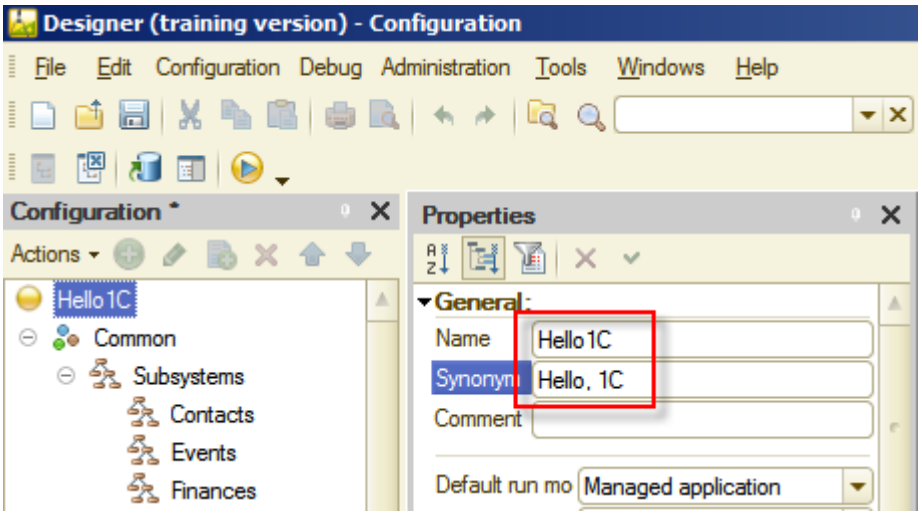

Figure 3-4. Applied solution properties

You can verify on your own that there is a shortcut to nearly any property of configuration object in the **Properties** window. In this tutorial, you will use **Properties** on numerous occasions.

Catalogs

By the project requirements, the applied solution must be capable of storing a list of people that the user contacts with and a list of events that are related to the user finances. In addition to storing data, those lists must be capable of classifying it. For the list of people, it must support the following classification: friend, acquainted, family, and so on. For list of events also must let to classify events by status.

Thus, for meeting these requirements, you will create three catalogs. For friends and other people create the **People** catalog. For contact information of various type (phone, address, email, and so on) create the **Contact types** catalog. For storing types of people relation to user (family, friend, acquaintance, and so on) create the **People relation types** catalog.

Another two catalogs will be used to store data regarding events: the **Events** catalog, where the actual data on occurred and planned events will be stored, and the **Event categories** catalog, intended to store categories of events (for example, study, sport, vacation, and so on).

Catalogs are located in the **Configuration** tree branch named **Catalogs**. Add a new catalog, for this right-click the **Catalogs** branch, and then click **Add** (Ins).

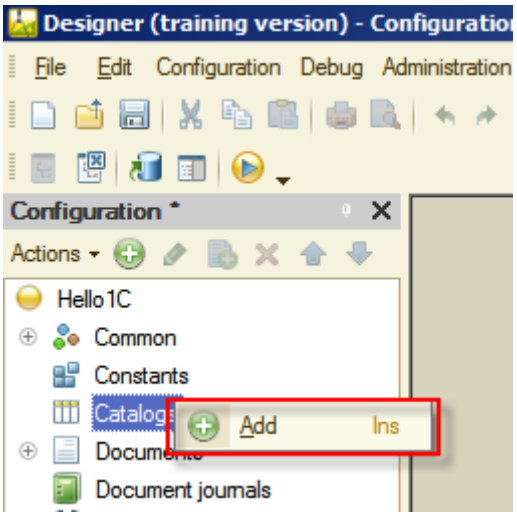

Figure 4-1. Adding a catalog

When you add a catalog, a new configuration object editor is opened. You already saw this window when created subsystems. Catalog is a complex object to configure, therefore, this editor is intended to ease and speed up the configuration procedure.

Although the same catalog properties can be specified in Properties on the right, it is more convenient to use this dialog window. Following all the tabs of this editor ensures that all the necessary properties are filled in and nothing is missed.

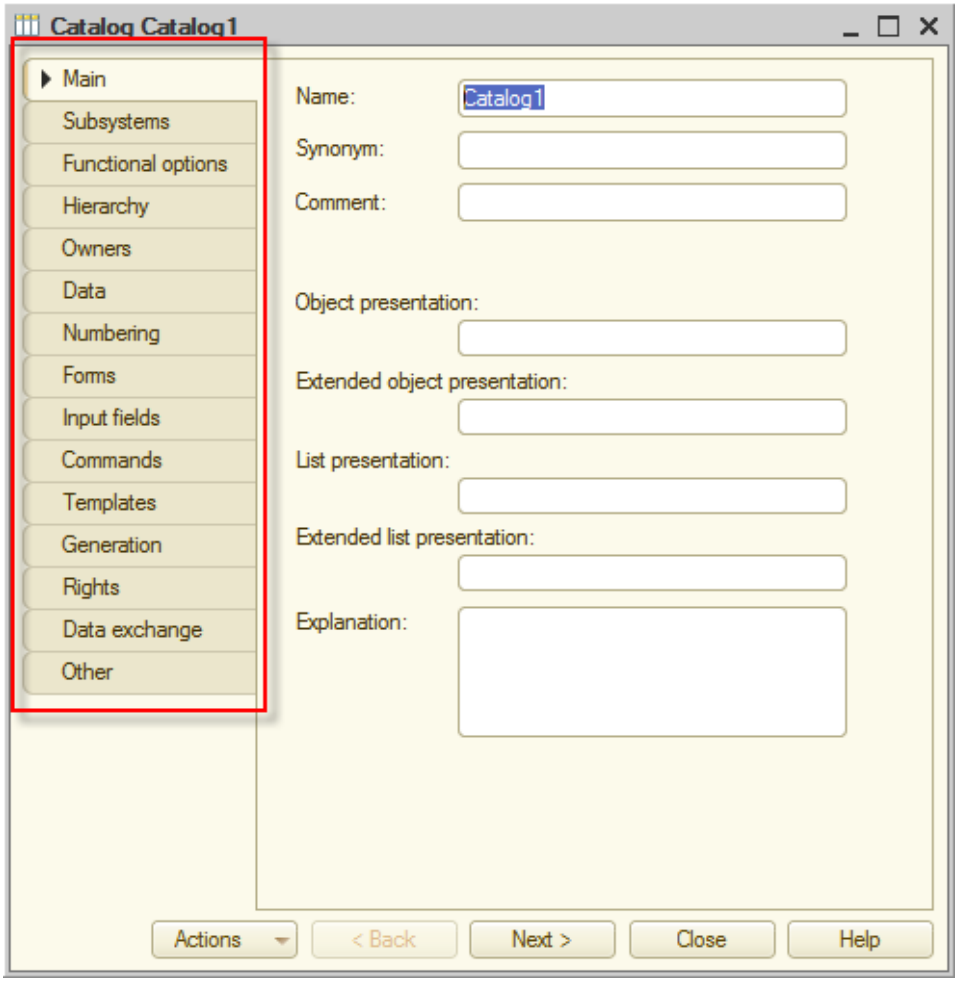

Figure 4-2. The Catalog configuration object editor

Type **ContactTypes** as the catalog name.

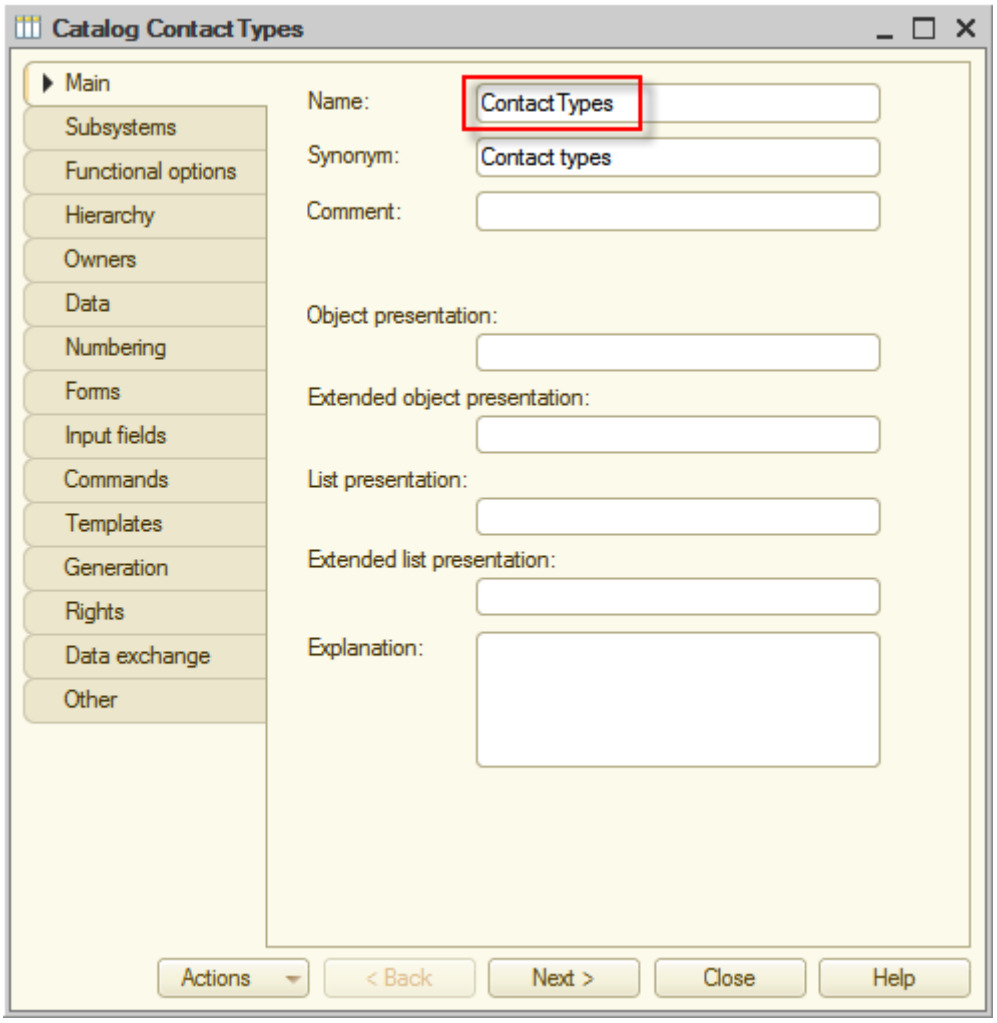

Figure 4-3. Adding the name and clicking Subsystems tab

There is nothing else to add on the **Main** tab, so click the **Subsystems** tab. On this tab, specify that this catalog belongs to the **Contacts** subsystem.

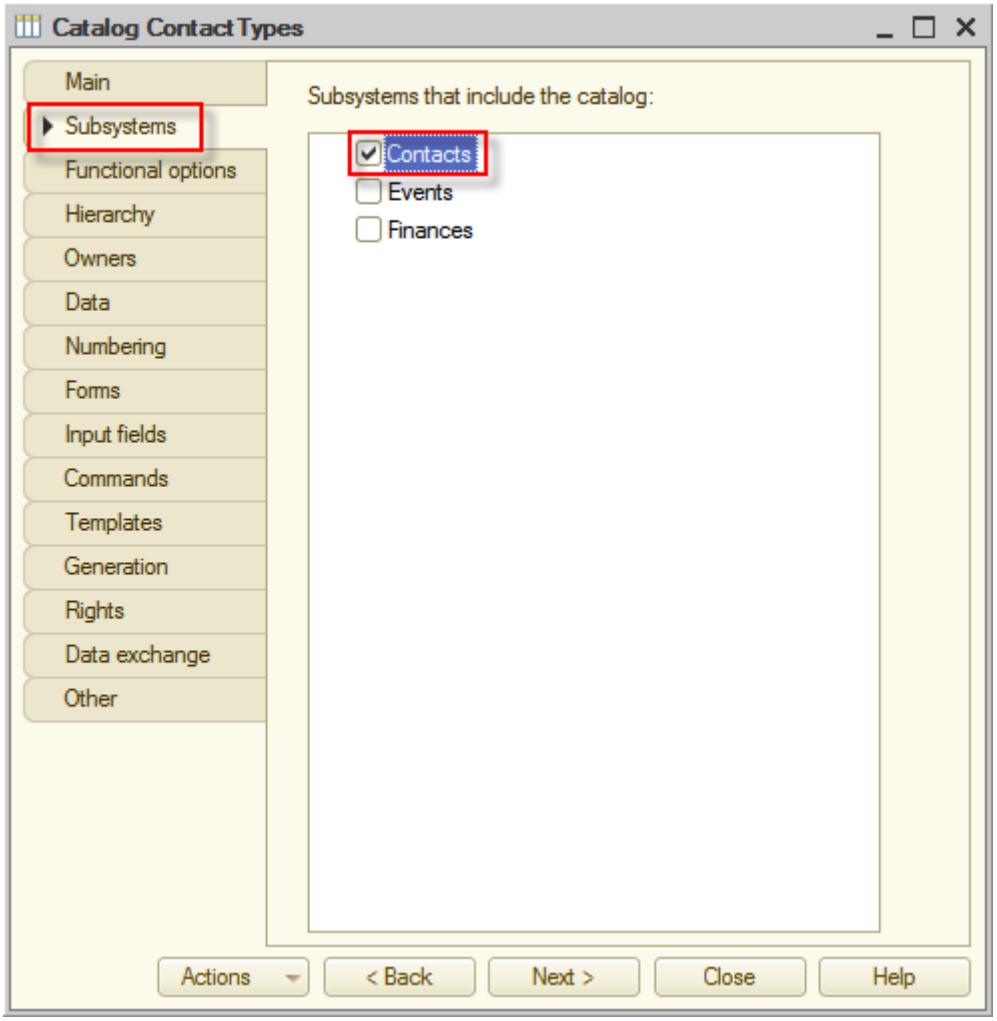

Figure 4-4. Specifying the subsystem

The storage for contact types is configured.

To start the application in the 1C:Enterprise mode from the Designer mode, click **Start Debugging** (F5). In general, this is not the 1C:Enterprise mode, but the Debug mode, but in this tutorial this is not important. Now look at how the **Contact types** catalog looks like.

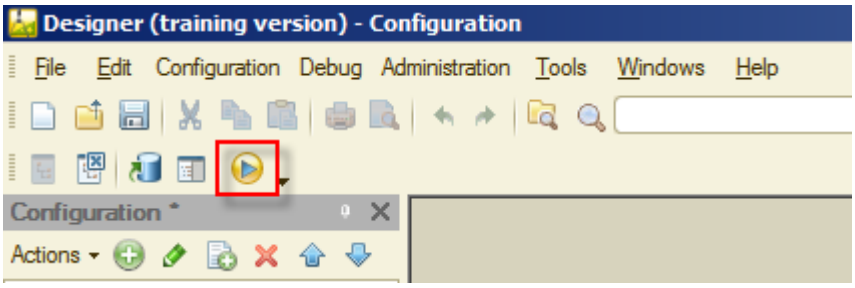

Figure 4-5. Start debugging

The platform will prompt you to update the database configuration, confirm this process.

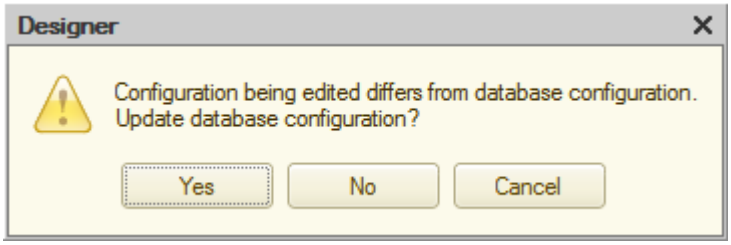

Figure 4-6. Updating the database configuration

The platform then will analyze the changes that had been made in configuration objects and inform you that a new object is added, named **Catalog.ContactTypes**.

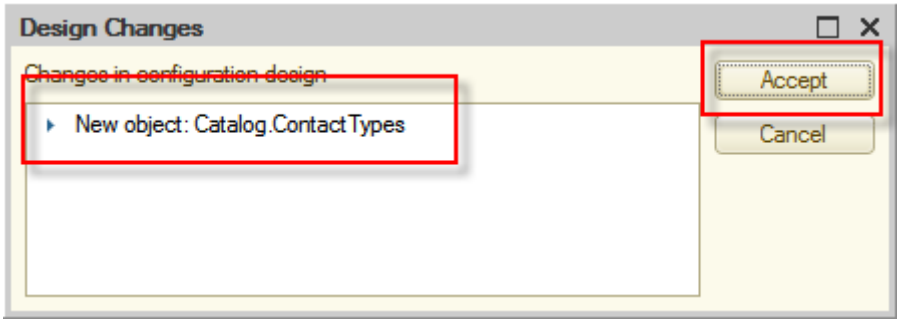

Figure 4-7. Restructuring data

Accept changes by clicking **Accept**. This type of extensive system check might seem odd to you, but for large infobases that contain hundreds of configuration objects, it is necessary.

After a short period, the platform will start in the 1C:Enterprise mode. You will see a blank main window of your application that will contain four sections, generated by 1C:Enterprise 8. These sections are quick menu, and previously added subsystems: **Contacts**, **Events**, and **Finances**.

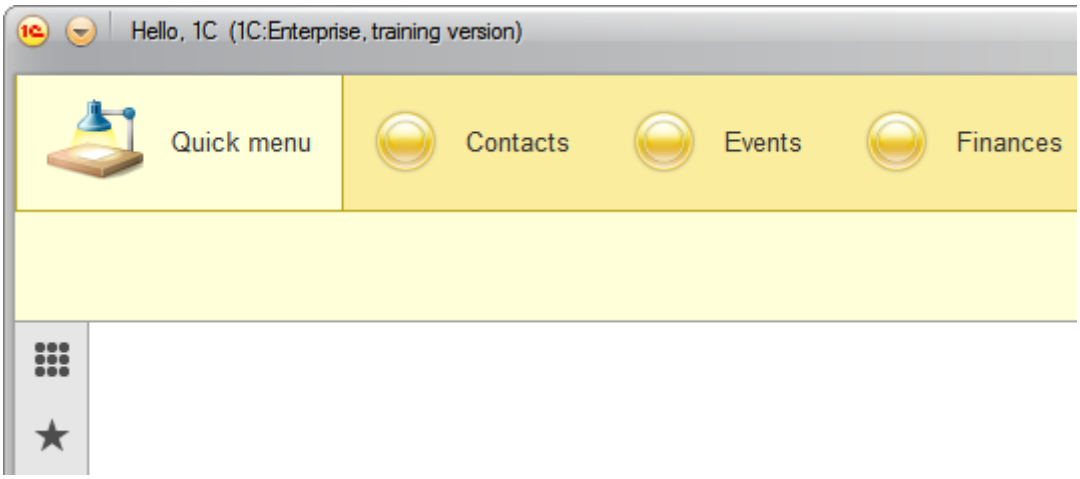

Figure 4-8. Subsystems in the 1C:Enterprise mode at first run

The catalog that you have just created is included in the **Contacts** subsystem. Thus, click **Contacts**, and see the **Contact types** link on the screen.

After clicking this link, you will see a list of contact types, which is currently empty.

To reduce the amount of manual data entry, this tutorial includes an import utility and the sample data needed to populate the catalog. This

utility is the data processor that will fill in the **Contact types** catalog data without entering the data manually.

The name of the data processor file is **ImportXMLData83.epf**. You can find it in the current distribution, for this, run **setup.exe**, click **Custom setup** and then click **Hello, 1C additional files**. In opened folder, you can find the data processor. Copy it and all other files in that folder to your computer.

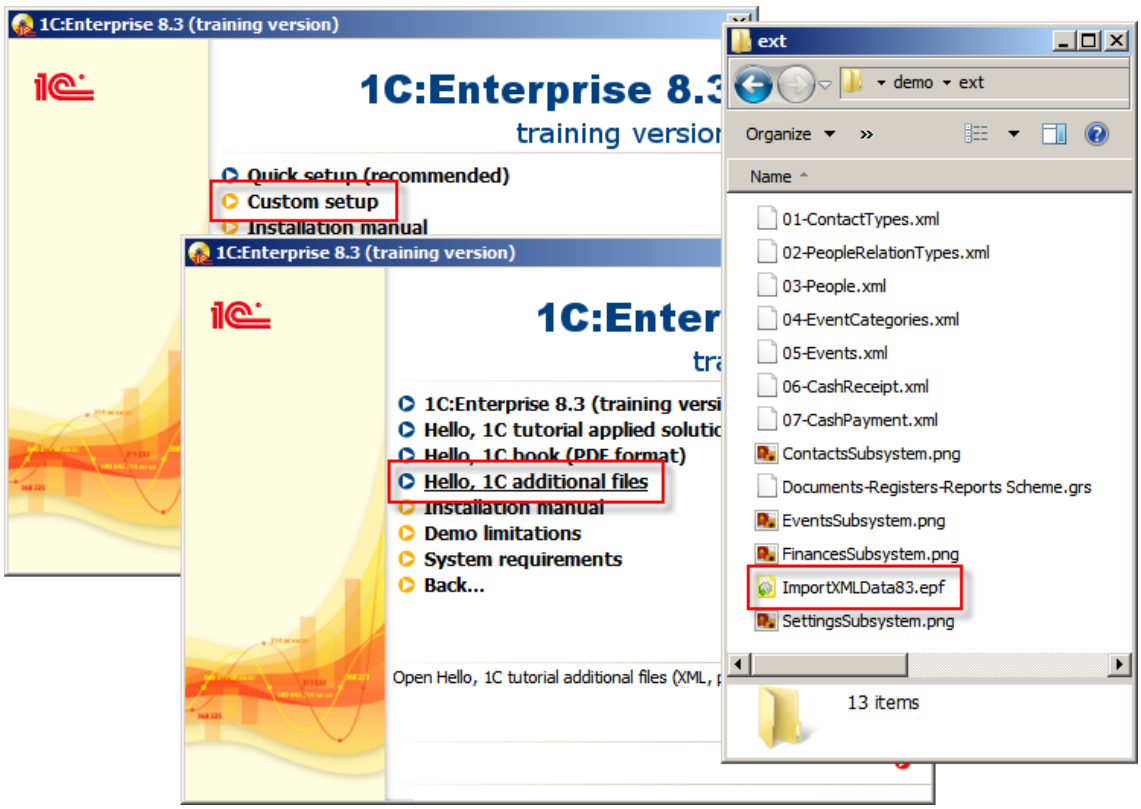

Figure 4-9. The data processor file

Important! This tutorial is designed and thoroughly tested using 1C:Enterprise version 8.3.5. It includes a few synchronous method calls. In 1C:Enterprise versions 8.3.6 or later synchronous method calls are disabled by default. If you are using 1C:Enterprise version 8.3.6 or later, double-click the root configuration node to open the property palette. Scroll down the property palette to find the **Synchronous call usage mode for extensions and add-ins property** and change its value to **Use**.

To run this data processor, click **Main menu** that is at the top left corner of the main application window.

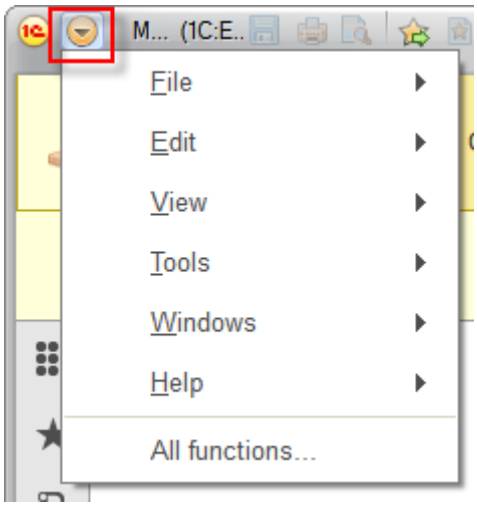

Figure 4-10. Main menu

In **Main menu** click **File** and then click **Open...** (Ctrl+O). In the opened dialog select **ImportXMLData83.epf**, and click **Open**.

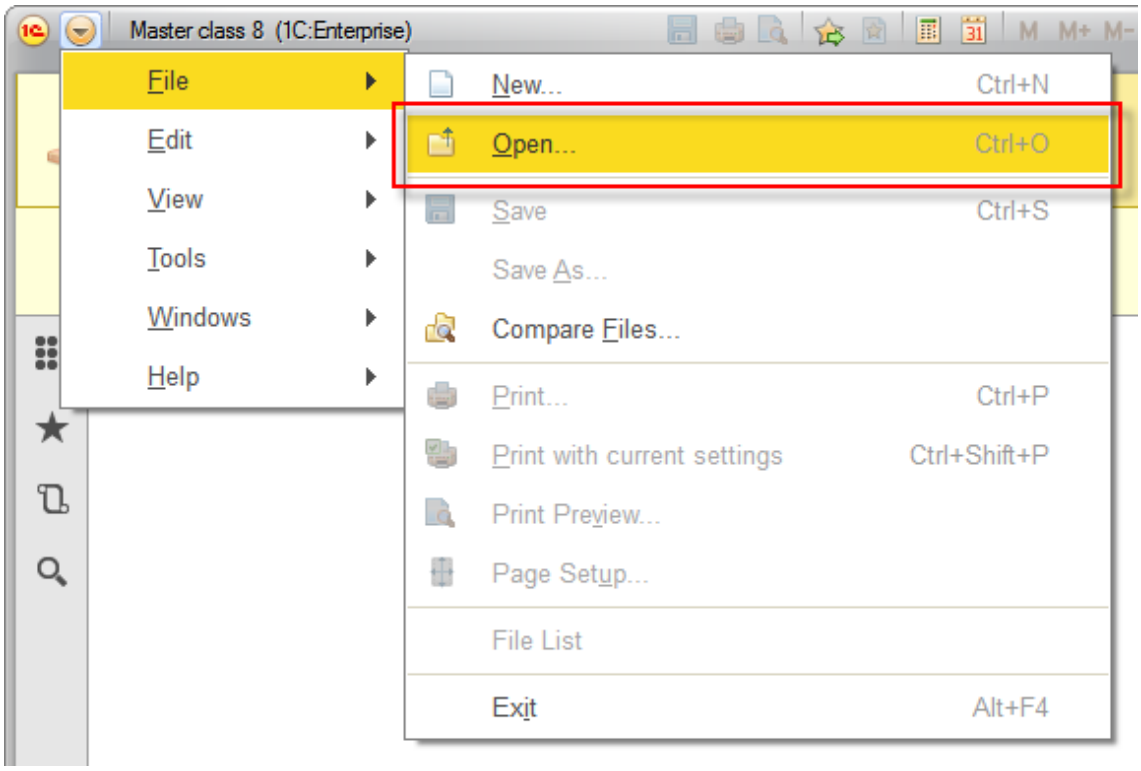

Figure 4-11. Opening the file

The data processor will let you to import the data from XML files.

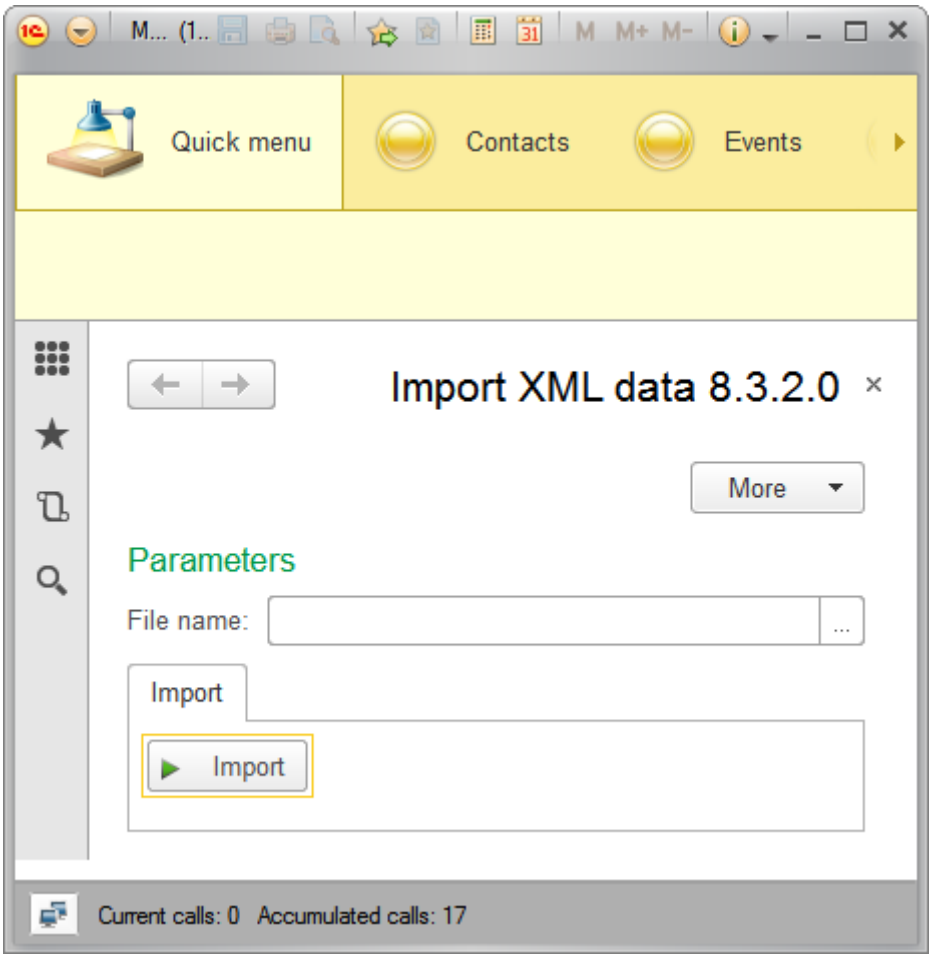

Figure 4-12. Importing data from XML file

In the opened data processor, click **Select** ..., and then find and select the **01-ContactTypes.xml** file.

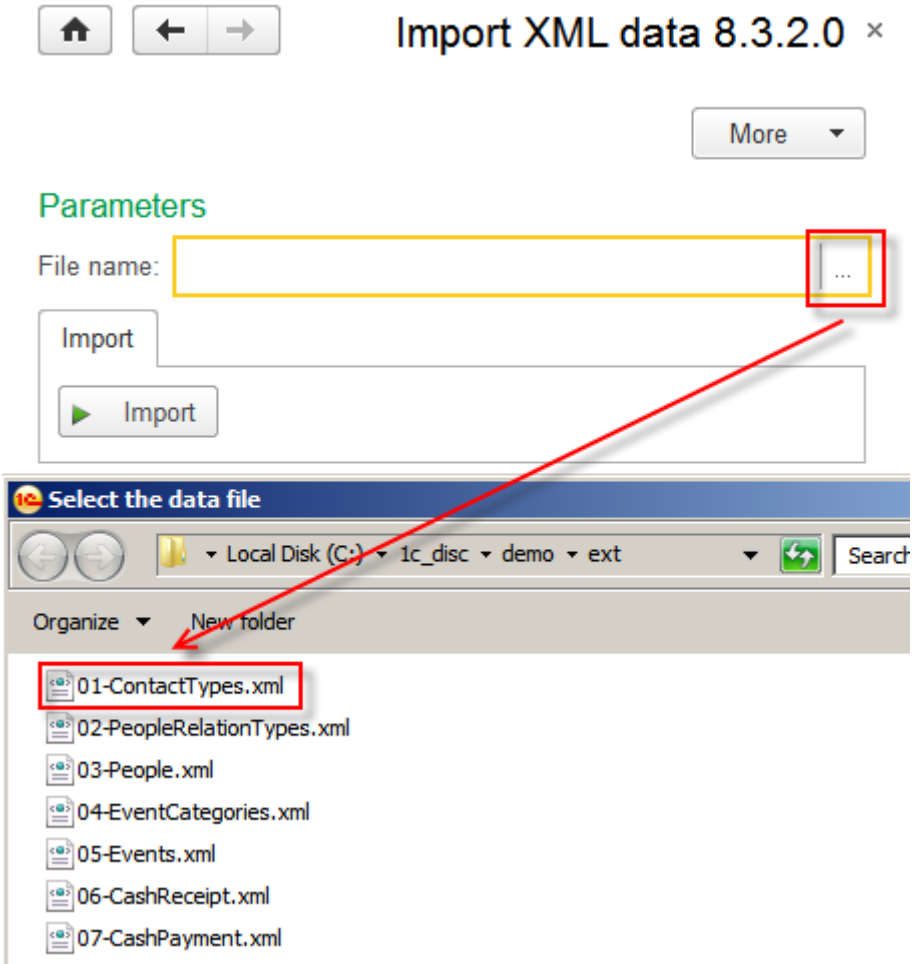

Figure 4-13. Selecting a file to be imported

After selecting the file, click **Import**. Shortly, the data processor will notify you that the data is imported.

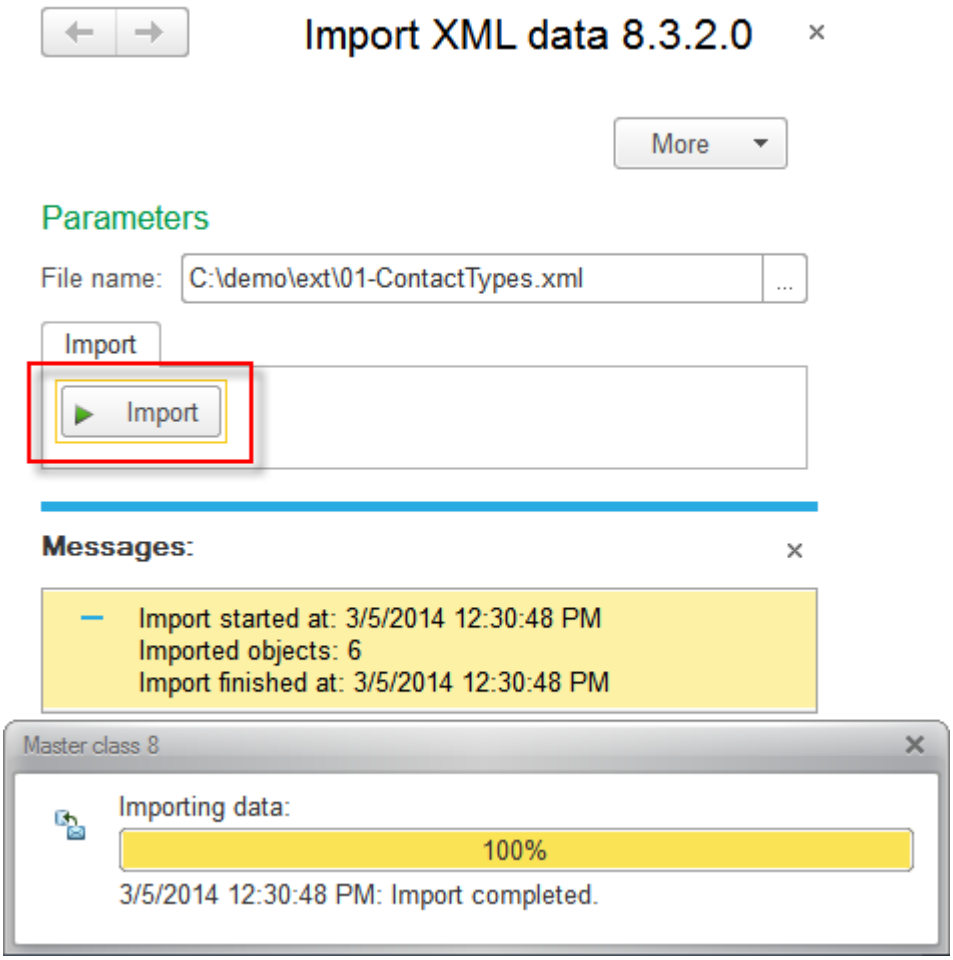

Figure 4-14. Importing the data

Close the data processor and open the list form of the **Contact types** catalog. If this form is already opened, press the F5 key to update it.

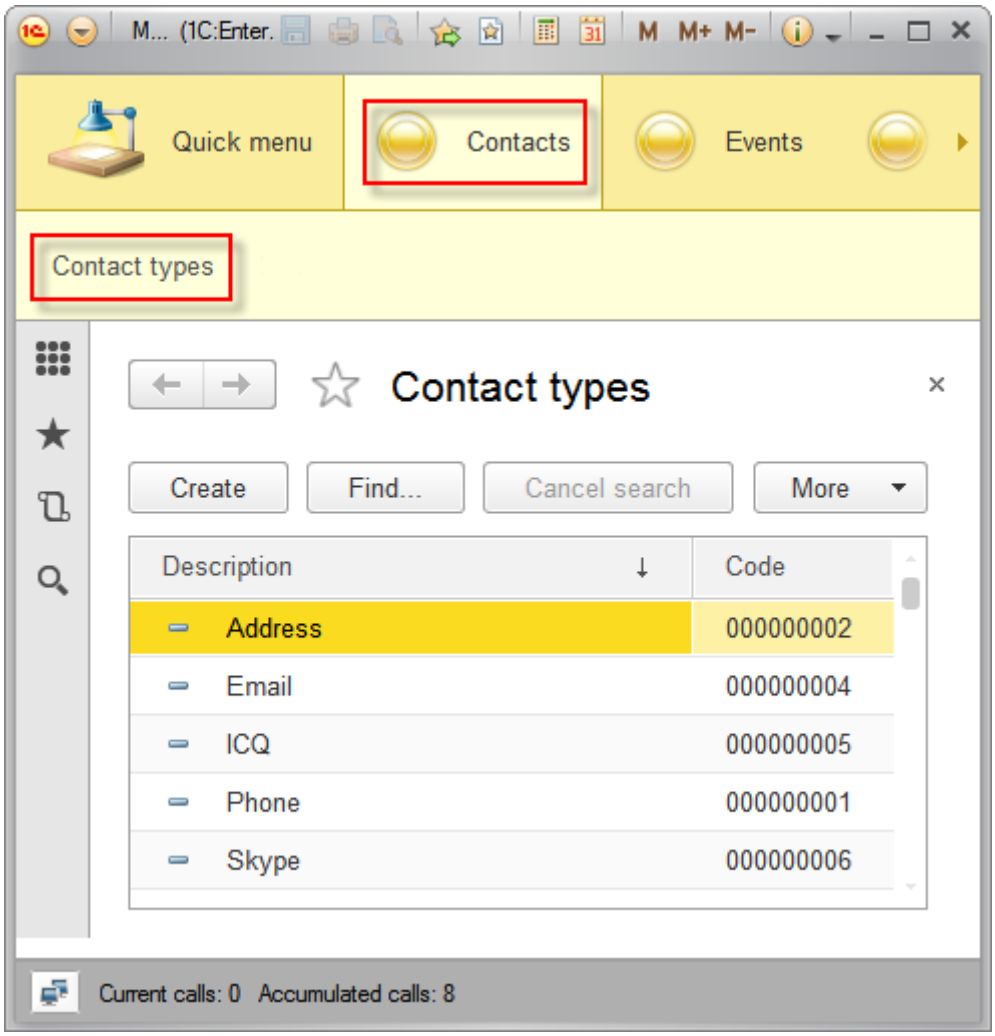

Figure 4-15. The list of items in the Contact types catalog

See that the data is correctly imported.

Notice that you did not create the list form for this catalog. A list form of a catalog is a list where all items of this catalog can be found. 1C:Enterprise generated this form for you automatically based on how you defined this catalog in the **Configuration** object tree. At the same time, all catalog commands are available, user can create new item, delete, search, and execute other actions.

Excellent. Close the main application window and return to the Designer mode. Here you will create remaining catalogs, and during that, you will learn about their individual properties.

Now create the **People relation types** catalog similar to the **Contact types** catalog. The only difference is that on the **Data** tab you will have to increase the length of the **Description** attribute to up to 150 characters.

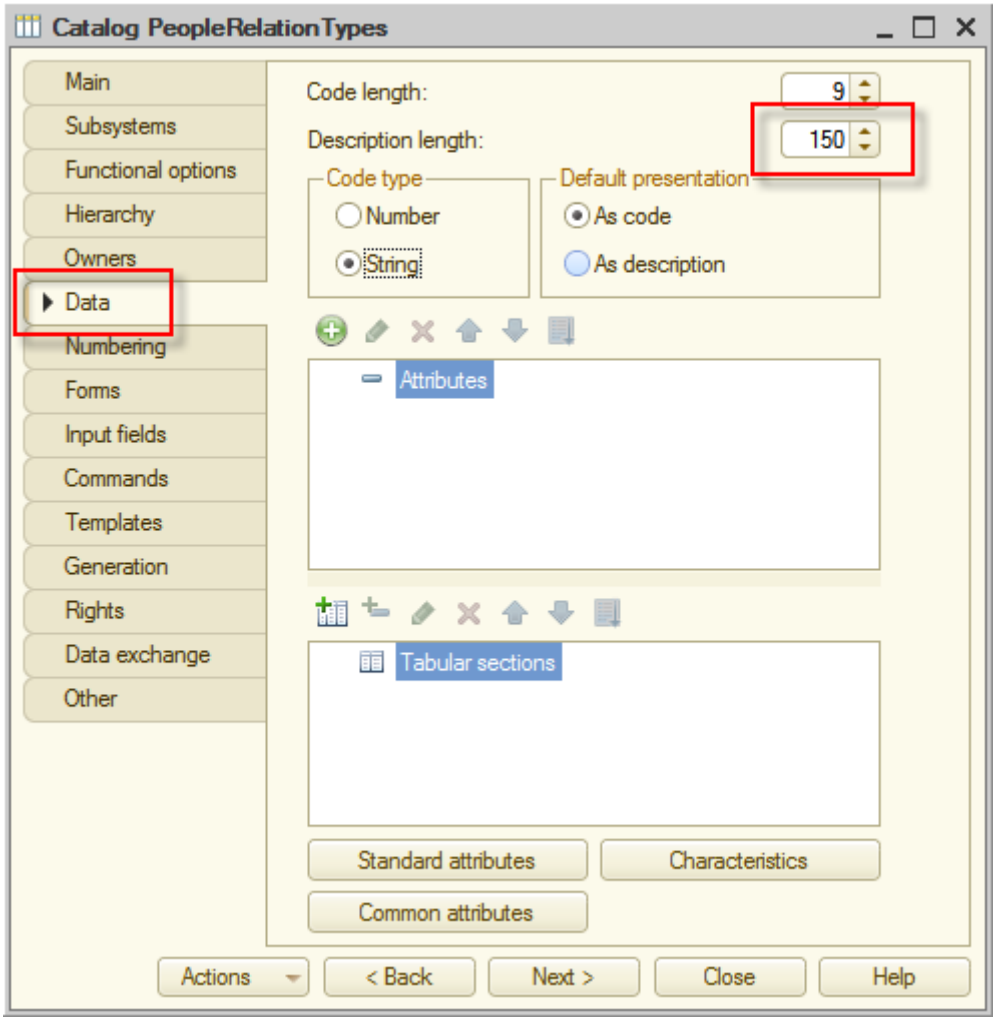

Figure 4-16. The People relation types catalog

Start the application in the 1C:Enterprise mode again and import the prepared People relation types data from **02-PeopleRelationTypes.xml**.

Because of changes that had been made previously, the **People relation types** catalog will be displayed as follows:

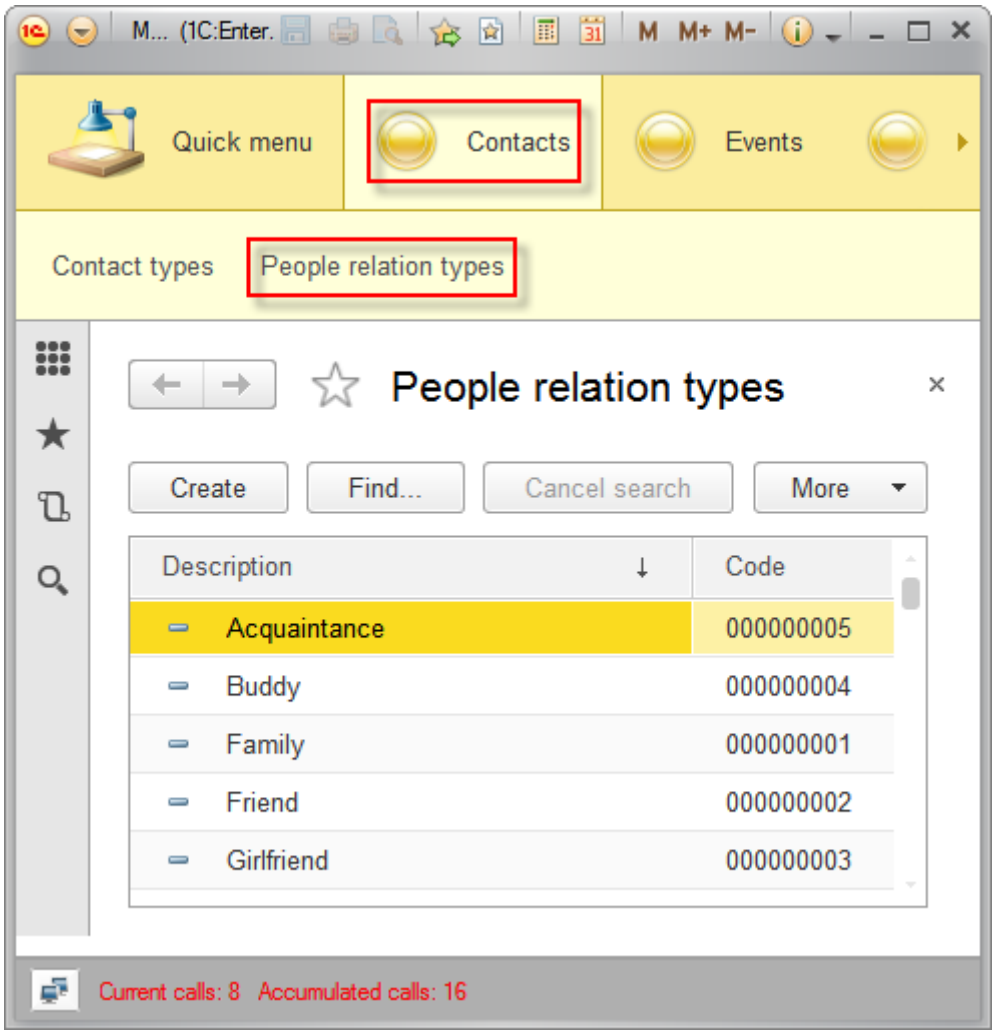

Figure 4-17. List of items in the People relation types catalog

Now, create a primary catalog, named **People**, where all people that a user knows will be stored. Same as previously created catalogs, include this catalog to **Contacts** subsystem. Then, click the **Data** tab, set length of the **Description** attribute to 150 characters. This attribute will be used to keep names of people.

Up to this point, it was nothing special, you have just created another catalog that will keep the list of names of friends and acquaintances and store those names in the **Description** attribute. However, there is very little benefit comparing to simply storing a list of names in a spreadsheet document. That is why you will use more features of 1C:Enterprise 8 and add more attributes to store additional information about people.

To store additional information, you can add any number of catalog attributes. In this tutorial, start with the following attributes: **Gender**, **RelationType**, and **Comment**.

- **Gender** attribute will store information about gender of people.
- **Relation type** attribute will store types of relation between users and people.
- **Comment** attribute will store any additional information about people that might be necessary to keep.

To add **Gender** as a new attribute, click the **Data** tab, then click **Add** (Ins) for the **People** catalog.

In **Properties**, enter **Name**, and **Synonym** once again will be generated automatically. There is one more thing to do. You need to specify the way to store data in the **Gender** attribute. As you can see in **Properties**, the default type for an attribute is **String**.

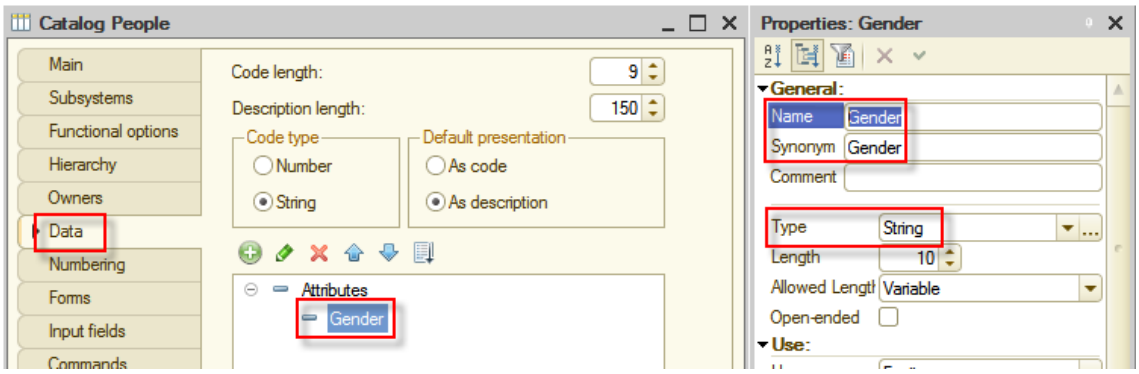

Figure 4-18. Adding the Gender attribute

You can leave everything as it is, but in that case, every time when adding a new person to the catalog, the user will have to enter gender manually in the **Gender** attribute. First, typing information manually for a large number of records is a very time consuming job. Second, there is a high risk of human error such as typo, wrong information input, and inconsistent values as there are many variations of writing genders. These errors, regardless of their nature, will make complicated any future automatic analysis of this data.

The first thing that may come to mind is to create the another catalog where specify both genders, and then use those catalog items. It is a reasonable idea, but creating a new catalog for just two objects is a redundant task. Which way is better then? There is another, new for you configuration object.

Enumeration is an object that is created and filled with data at the development of an applied solution stage. An enumeration can store a limited, known beforehand, and same type of data. **Gender** is precisely the type of data that is good to be stored as an enumeration, rather than as a catalog.

To create a new enumeration, leave **People** catalog as is for now and find **Enumerations** section in the configuration tree, then repeat the already familiar process of adding a new configuration object.

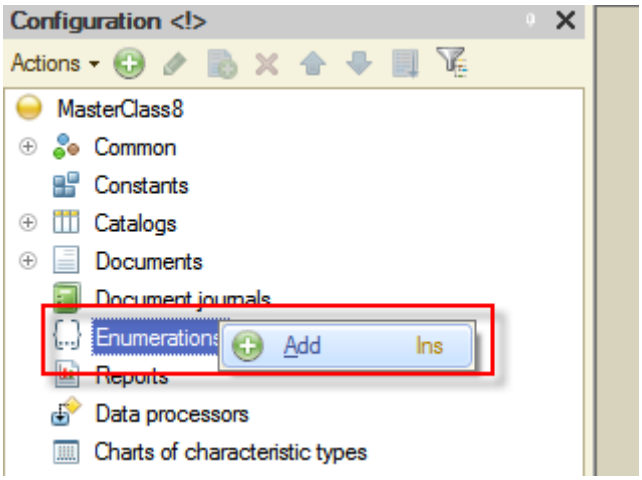

Figure 4-19. Adding an enumeration

In the opened window, as the **Name** of the new enumeration enter **Genders**. Then include the new object to the **Contacts** subsystem.

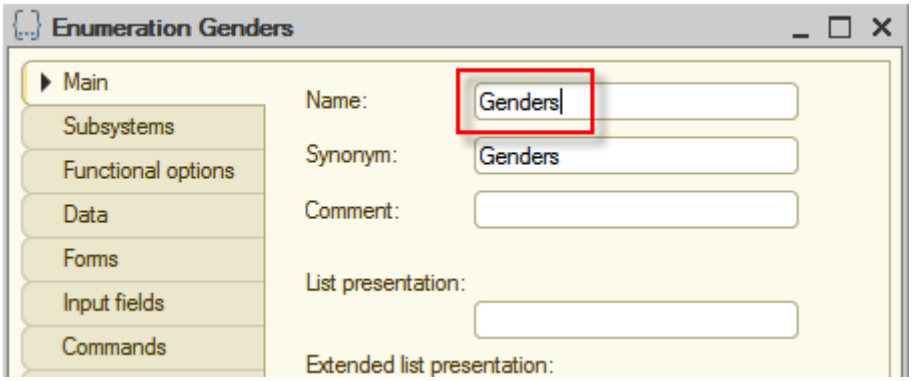

Figure 4-20. The Genders enumeration

Click the **Data** tab, and then add two values using **Add** (Ins), those values are **Male** and **Female**.

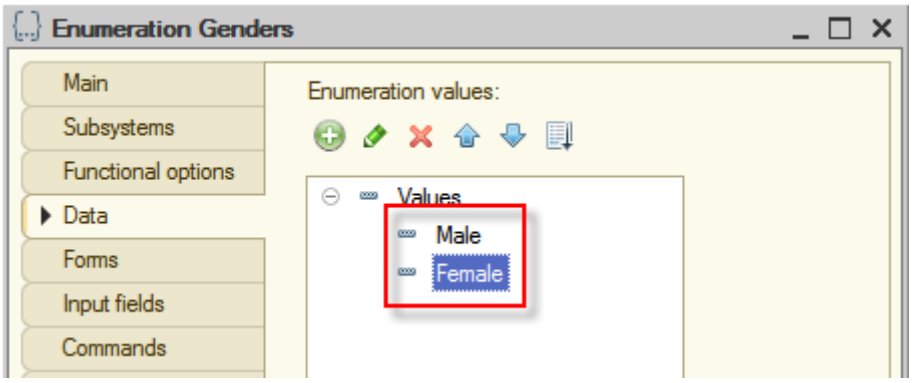

Figure 4-21. Values of the Genders enumeration

After the values are added, close the enumeration editor by clicking **Close**. You will see the previously opened **People** catalog editor. Now you can change the type of the **Gender** attribute from **String** to the newly created **Genders** enumeration. To do that, select the **Gender** attribute and in the **Properties** window in the **Type** property click **Select** button. If you accidently closed the **Properties** window, you can always open it by right-clicking on the configuration object and selecting **Properties** or by double-clicking the attribute.

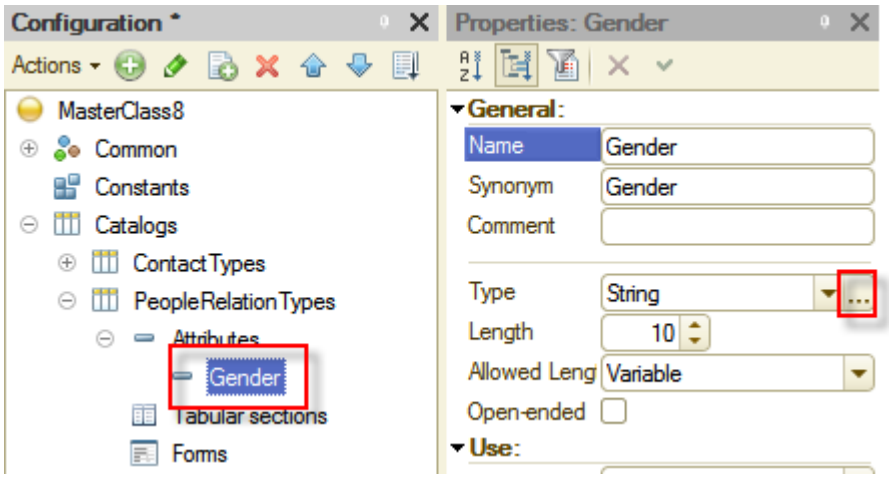

Figure 4-22. Changing the type of the Gender attribute

In the opened window, find and select the **Genders** enumeration, then click **OK**.

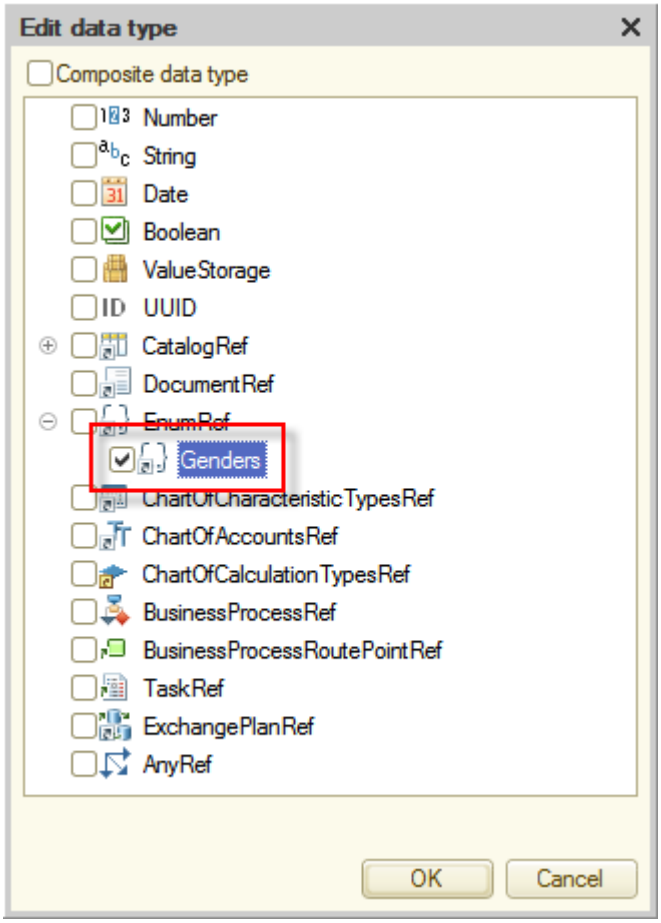

Figure 4-23. Selecting the Genders enumeration as the type of the attribute

The accuracy of the selection for the type can always be verified in the **Properties** window.

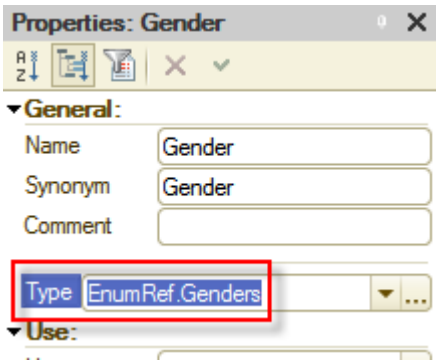

Figure 4-24. Verifying the attribute type

After making sure that everything is done correctly, return to the **People** catalog editor, and continue adding new attributes. Next attribute to be added is **RelationType**. Similar to **Gender**, add **RelationType** as an attribute and then select **PeopleRelationTypes** as its type, which is placed in the **CatalogRef** group in **Edit data type** window.

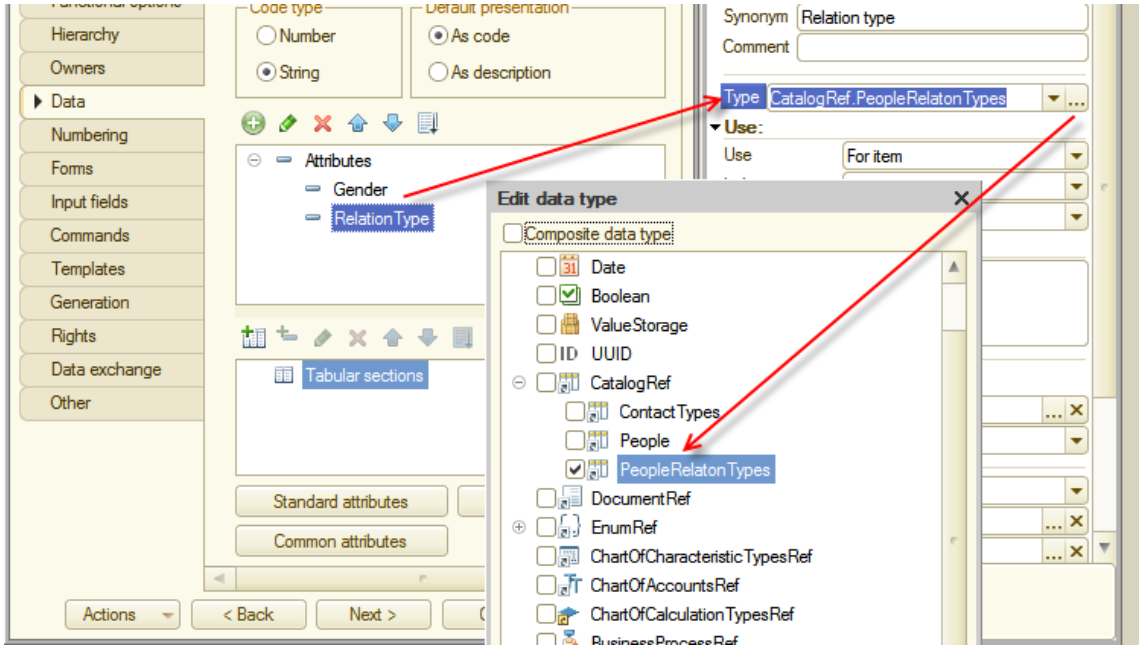

Figure 4-25. Creating the RelationType attribute

The last attribute to be added is **Comment**, which will keep notes and thoughts about a person. When adding this attribute, accept the default data type, **String**. However, the length value needs to be changed. To avoid any guessing while determining the appropriate length of comments, simply set the length of the **Comment** attribute string to **Open-ended**. For this select **Open-ended** check box in **Properties**.

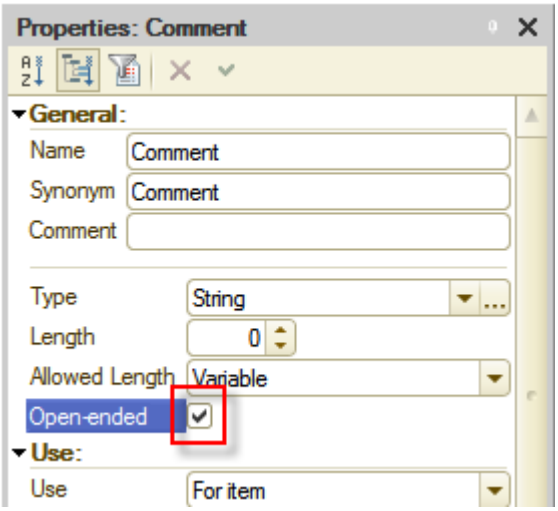

Figure 4-26. Setting length of Comment attribute string to Open-ended

Keeping in mind the task requirements, you should create another set of attributes where contact information of people will be stored. You might conclude that since you have created **Contact types** catalog, then should add the **Contact type** attribute of the **CatalogRef.ContactTypes** type, that will keep the information about the type of the contact. Then add at least one more attribute that will store the value of that contact, and name it **Contact value**. In other words, to store a person's address, you will have to select the **Address** as the type of the contact, and then input the actual address in **Contact value** attribute.

At this point, consider one more thing. Once you will need to keep more than one contact, if you will store them in the same way as described above, you will have to add as many pairs of **Contact type**/**Contact value** attributes as you need to keep a required variety of contacts. But it breaks down the efficiency of the storage since every person will have own pairs of data. Furthermore, every time a new type of contact appears, you will be required to add a new pair of attributes. This procedure have to be repeated each time a new piece of information occur.

Fortunately, there is no need to do that. It is clear that every person has an own set of contact information. 1C:Enterprise 8 solves this problem by allowing you to create tables for each catalog item.

Add the **Contacts** tabular section for **People** catalog. Click **Add tabular section** , and then name the table **Contacts**.

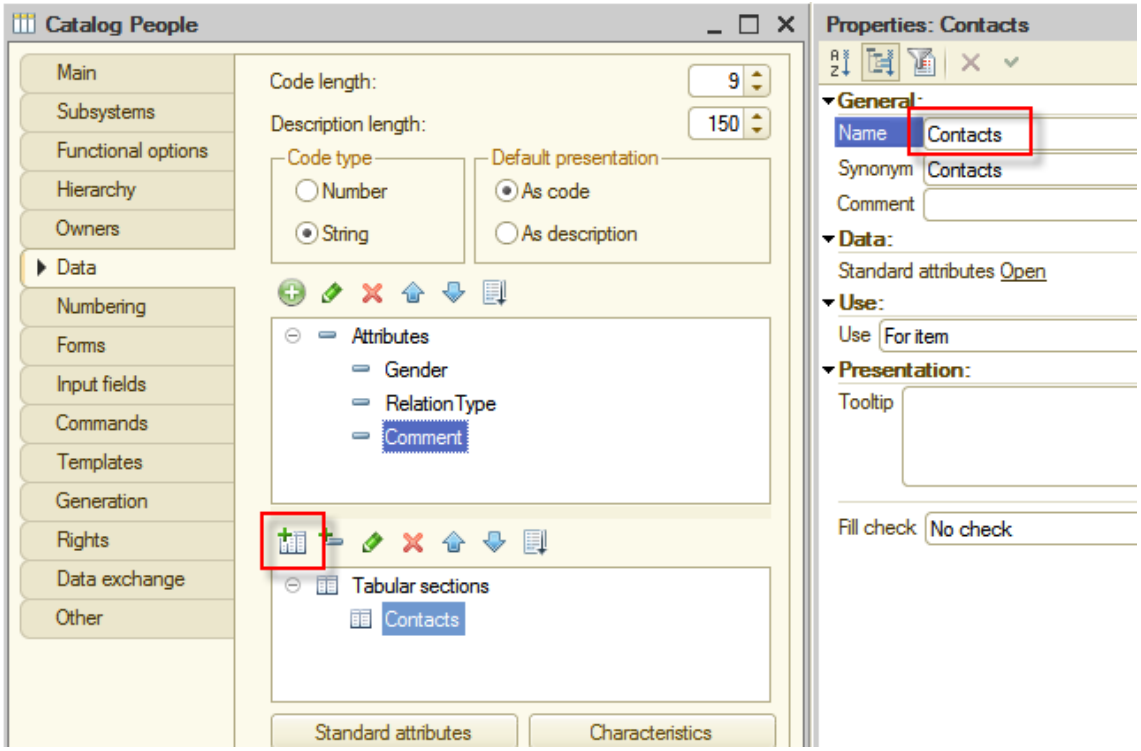

Figure 4-27. Adding the Contacts tabular section

Return to the **People** catalog editor, click on the **Contacts** tabular section, and add a new attribute to that tabular section by clicking **Add attribute** . Name a new attribute **Type** and select the **ContactTypes**, which is located in the **CatalogRef** group as its type.

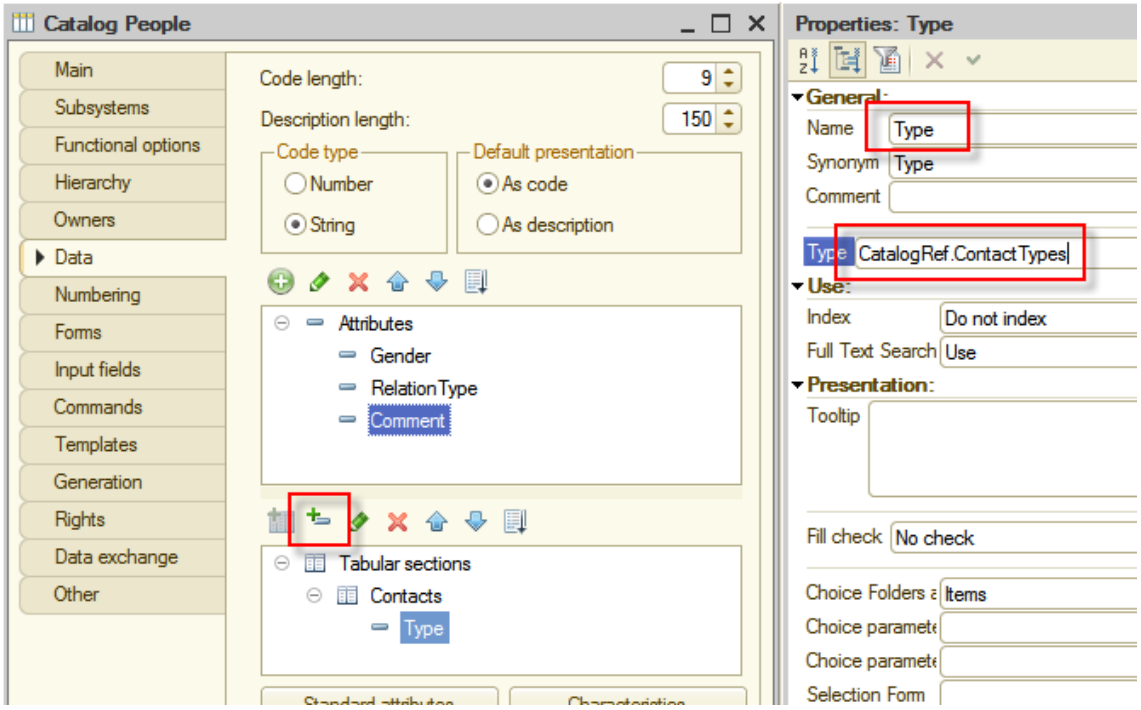

Figure 4-28. The Type attribute of the Contacts tabular section

In the same manner, add the **Value** attribute, which is going to be a **String** with length of **1,024** characters.

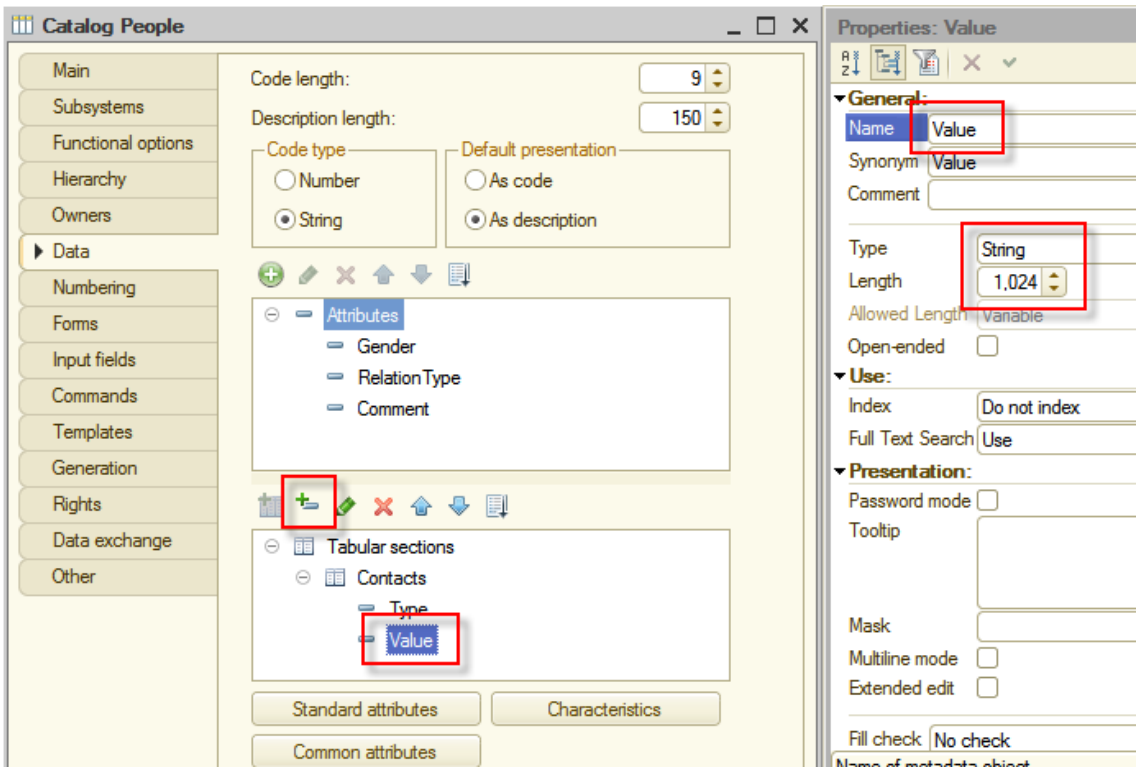

Figure 4-29. The Value attribute of the Contacts tabular section

At this point the **People** catalog is considered finished. Take a look what you have now. To start the application in the 1C:Enterprise mode click **Start Debugging** (F5).

Confirm the system request to update the configuration, and then accept the changes to the configuration.

Now go to the **Contacts** section that you have seen already, and look at what you have created.

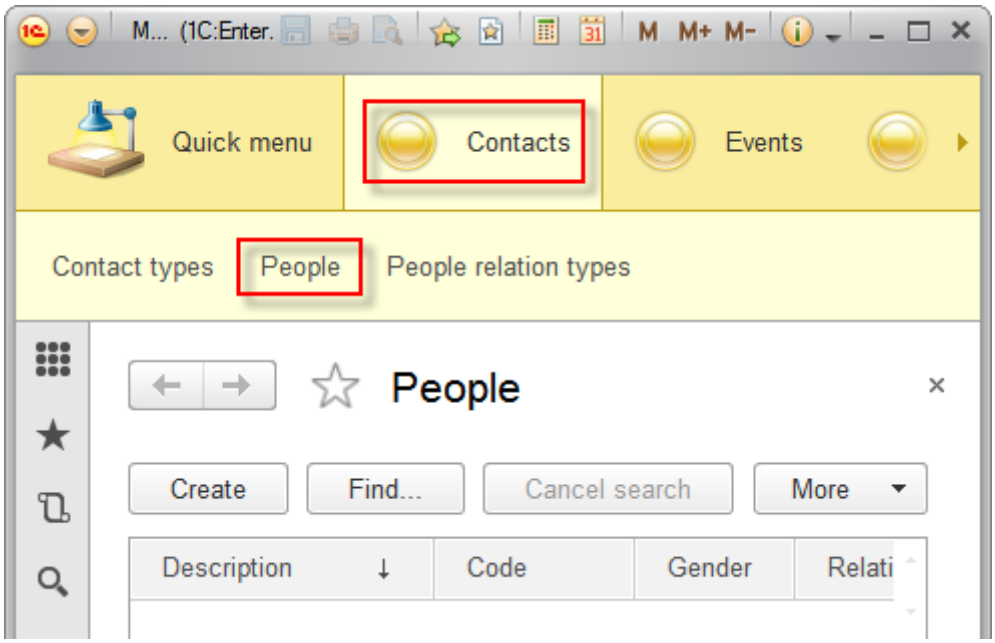

Figure 4-30. The People catalog in the 1C:Enterprise mode

For now, this catalog is empty. Open the **ImportXMLData83.epf** data processor and import data from the **03-People.xml** file. Press the **F5** key to update the list, if it is empty.

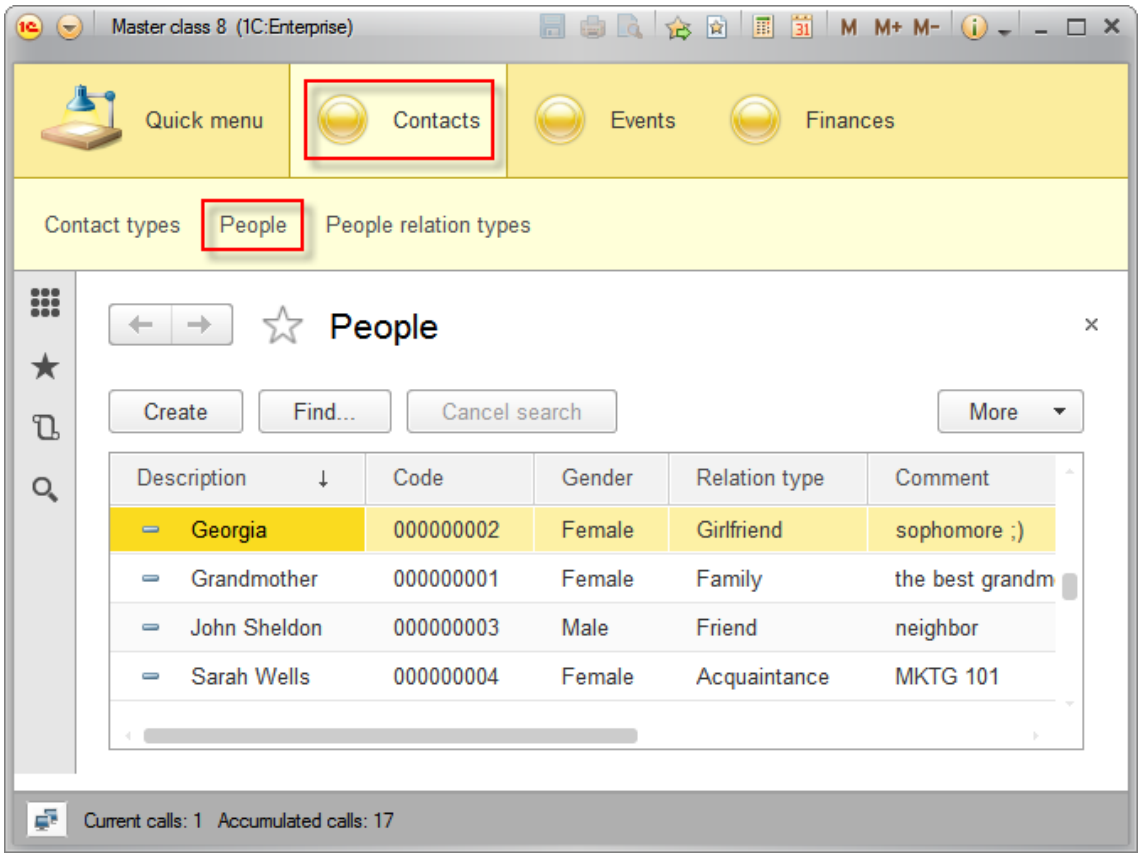

Figure 4-31. The People catalog, filled with data

Now the list contains people. To verify, which data is contained in individual records, double-click any record.

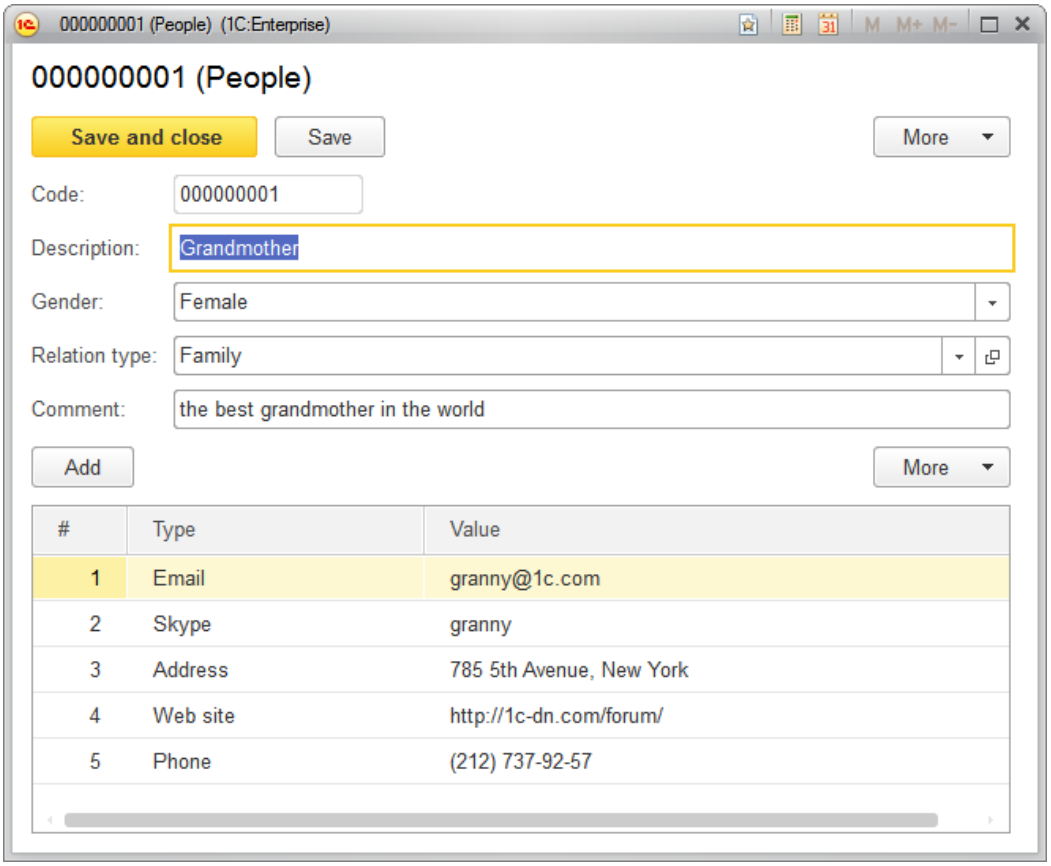

Figure 4-32. The item of the People catalog

As you can see, main attributes, related to a person, are located on the upper part of the window, and the contact information is conveniently arranged in the tabular section on the bottom.

Note that in addition to generation of forms to display lists of catalog items, the platform also automatically generated a form of catalog items.

Close the main application window and return to the Designer mode to add two more catalogs, which are **Event categories** and **Events**.

The **Event categories** catalog is created in the same way as the **People relation types** catalog. The **Description** attribute length will be 150 characters as well. The only difference is the association with the subsystem. **Event categories** will be included to the **Events** subsystem.

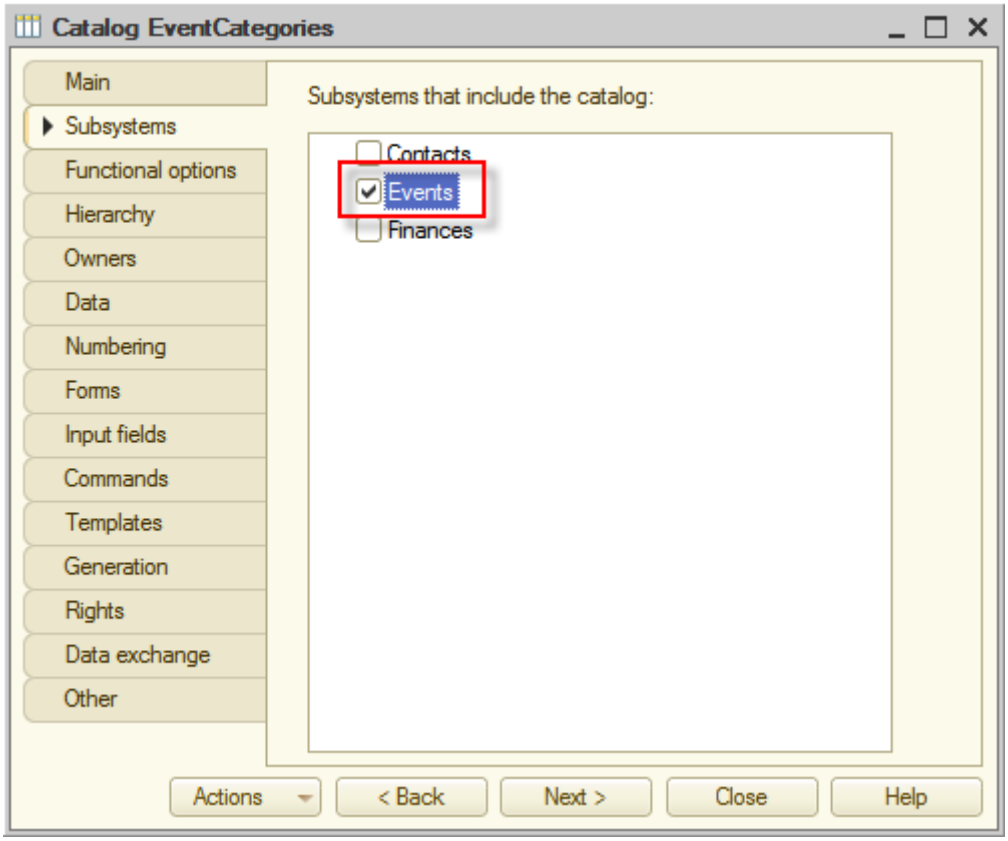

Figure 4-33. Making Event categories a part of the Events subsystem

In the 1C:Enterprise mode, import the data for this catalog from the **04-EventCategories.xml** file.

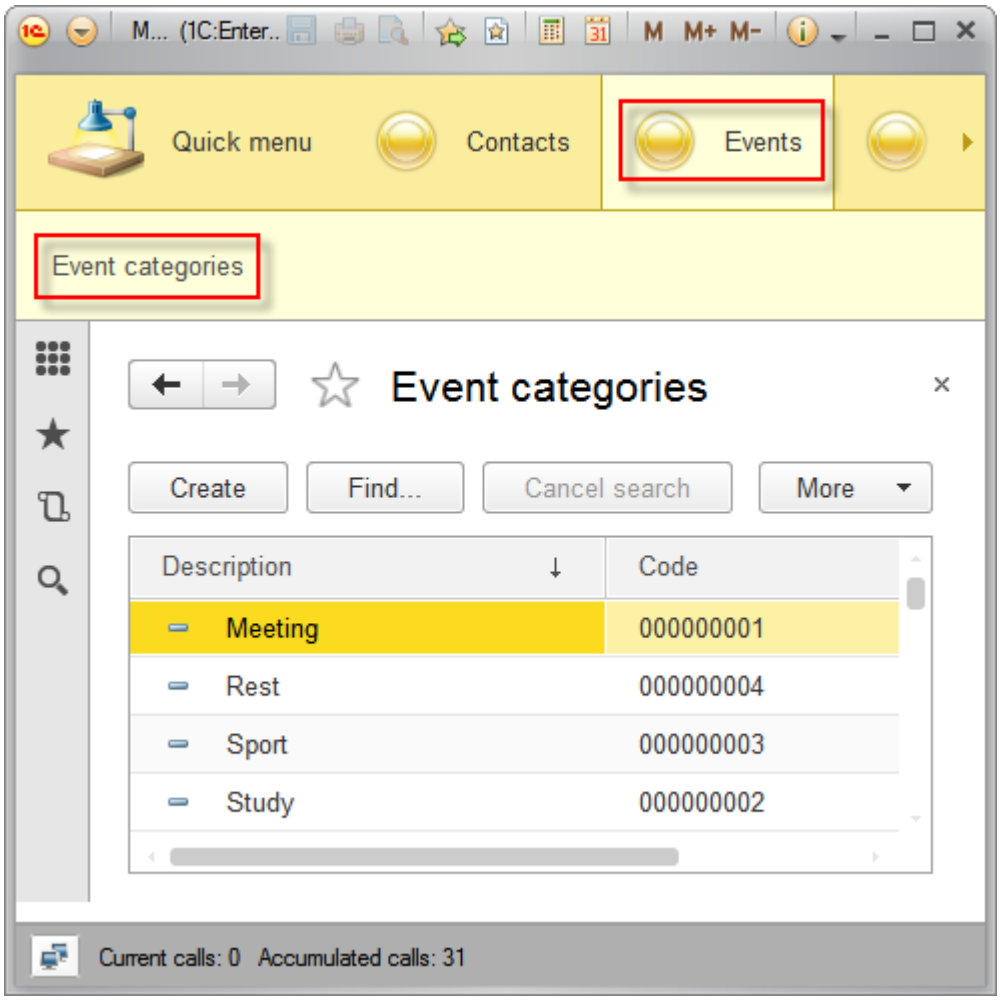

Figure 4-34. The Event categories catalog, filled with data

The last thing you need to do in this chapter is to add the only remaining catalog, named **Events**.

The **Events** catalog will also be included to **Events** subsystem. Set the length of the **Description** attribute to 150 characters. Then, add following attributes to this catalog:

- **Begin date** with **Type** equals to **Date** and **Date Content** – **Date and time**.
- **End date** with **Type** equals to **Date** and **Date Content** – **Date and time**.
- **Category** with **Type** equals to **CatalogRef.EventCategories**.
- **Details** with **Type** equals to **String** and checked **Open-ended** box.

Add a new tabular section, named **Participants** and add an attribute **Participant** of **CatalogRef.People** type to this tabular section.

As the result of these changes, the **Data** tab will look as follows:

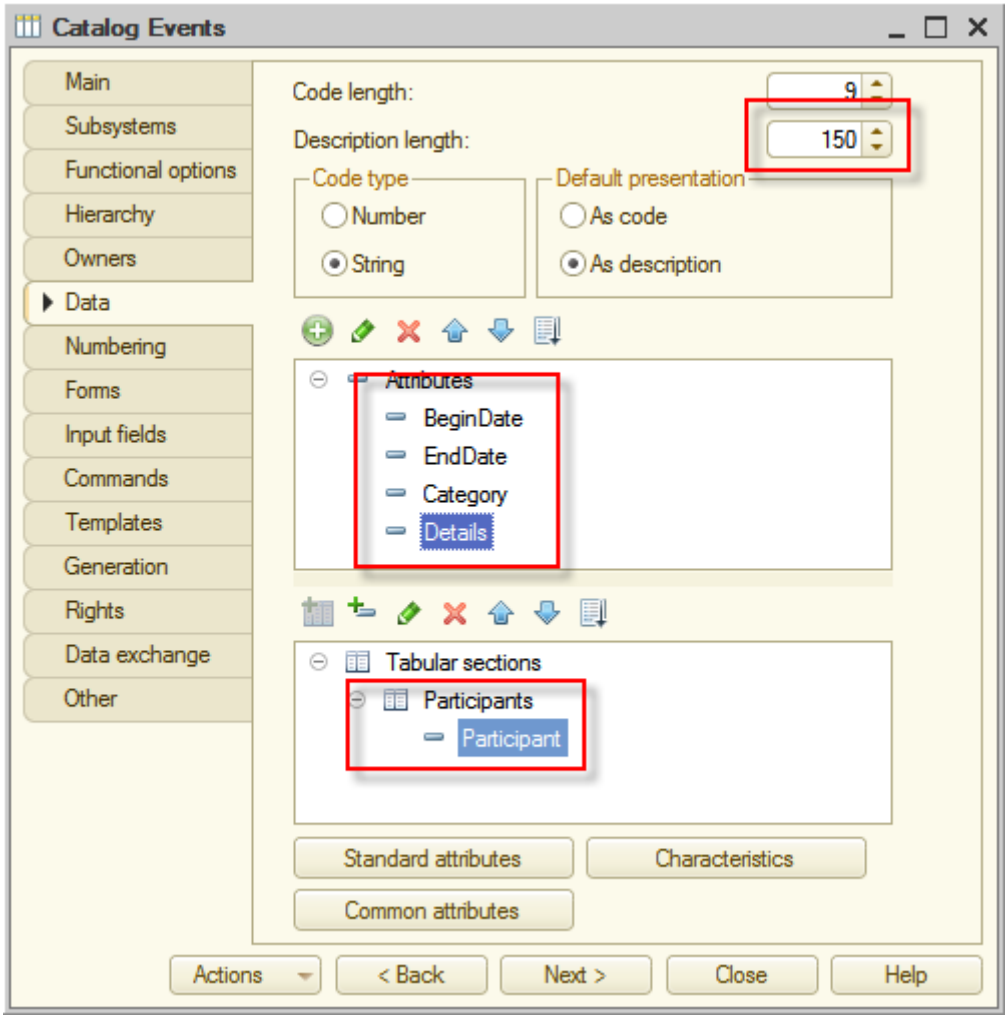

Figure 4-35. Properties of the Events catalog on the Data tab

Click **Start Debugging** (F5), and confirm all changes. In the 1C:Enterprise mode import data into the **Events** catalog from the **05-Events.xml** file.

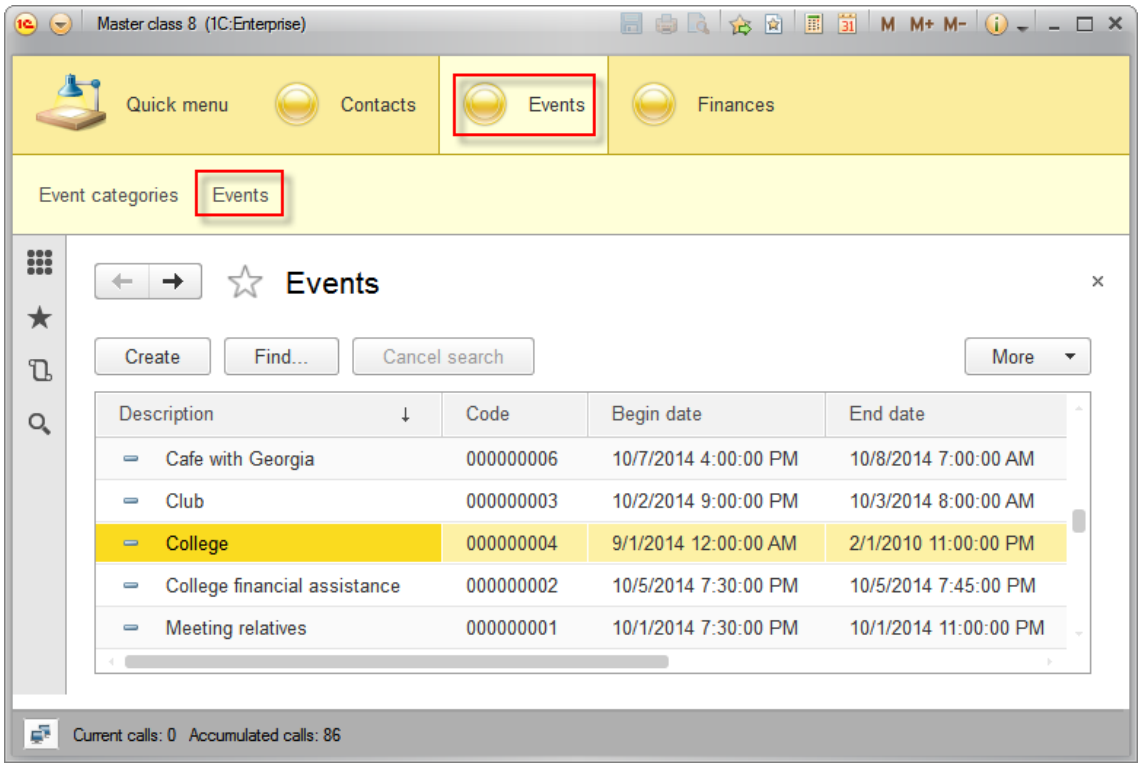

Figure 4-36. The Events catalog, filled with data

See the result in one of **Event** items.

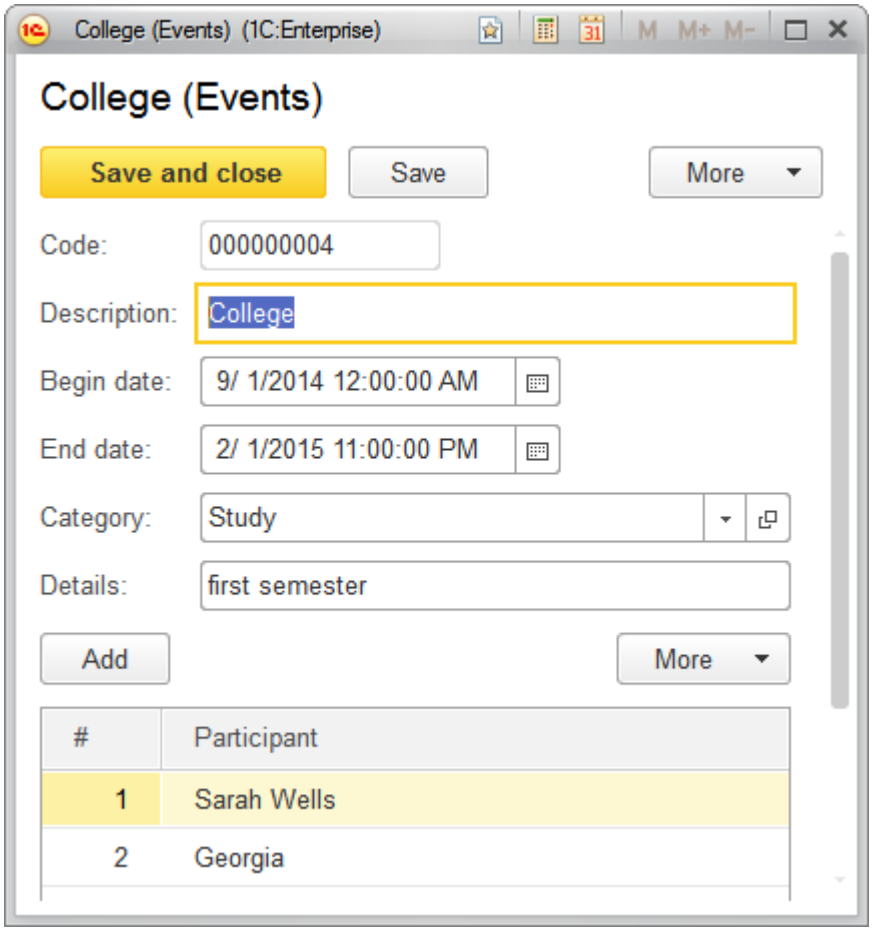

Figure 4-37. The Events catalog item

Now you have created everything that is required to keep track of events in life of users.

Excellent! The **People** catalog can store the information about all of user's relatives, friends, and acquaintances, including a variety of contact information of these people. The **Events** catalog stores the information about past and planned events, also participants can be specified here. At the same time, three auxiliary catalogs, **ContactTypes**, **PeopleRelationTypes**, and **EventCategories** help users to customize data, related to primary catalogs.

Moving further.

Register

Now add a building block that may not be obvious from the first sight. It will be the **Financial transactions** accumulation register.

Questions, "What is a register? What is its purpose?" immediately arise.

Below you will find a simple explanation.

According to project requirements, the application should not only be capable of storing a list of friends and tracking events. It also should be capable of keeping records of financial transactions. At a minimum, it should keep records of cash flow and provide simple financial reports.

Registration of various events related to money income and outcome will be implemented by using configuration objects, named **Documents**. Later, you will add a couple of documents to the applied solution.

Together with registering money incomes and outcomes, it is required to create some reports, demonstratively displaying what happens to finances. These reports can be created using data that is contained in user input documents. These documents reflect actual receipts and expenditures of money. However, imagine that in a month (or a year), for example, you decide to supplement your financial records with new documents or functionally expand existing ones.

Making configuration changes, as you have already seen, is a simple task. However, what will happen with reports in this case? It might turn out that financial reports that you created previously will not work properly, because they use only income and outgoing records from the old documents as data source. This means that you might have to rebuild all related to this data reports. Live applied solutions usually contain a large number of reports.

To avoid this work, and a large number of errors that may become a result of this alteration, 1C:Enterprise uses the following development methodology:

To store data regarding activity of accounting subjects, finances for example, configuration objects named **Registers** are used. **Documents** that reflect business activities, record data in these registers. Reports, then, use these registers as data sources.

As a result, you have a separated structure. On one hand, once there is a complete set of registers, it is easy to create required reports that will demonstratively display information contained in these registers. On the other hand, when a new type of document appears in the applied solution, you only need to add the correct algorithm to place data in registers. This

methodology guarantees that previously created reports will not require any alterations and will display correct data.

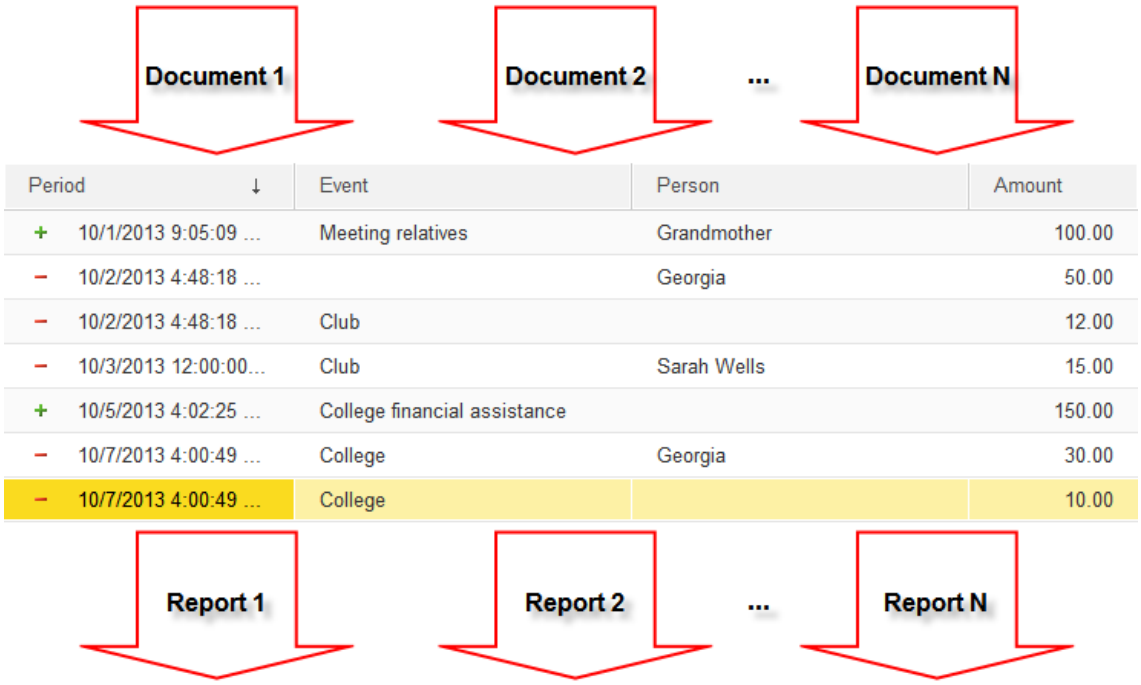

Figure 5-1. The Documents–Registers–Reports schema

So, for meeting project requirements of this tutorial it is enough to create only one accumulation register, name it **Financial transactions**. Where the application will store data on how much and how often a user receives and spends money.

Accumulation registers are located in the **Accumulation registers** branch of the **Configuration** object tree. Add a new accumulation register.

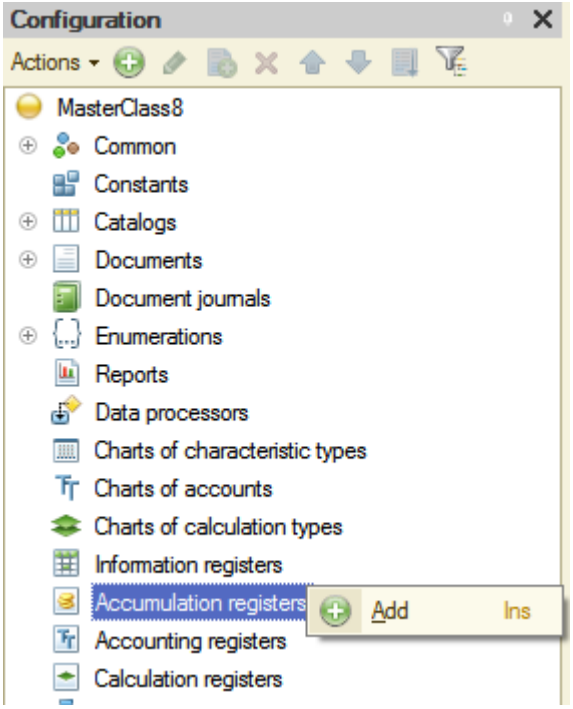

Figure 5-2. Adding an accumulation register

Name this register **Financial transactions** and include it to **Finances** subsystem. Then click the **Data** tab.

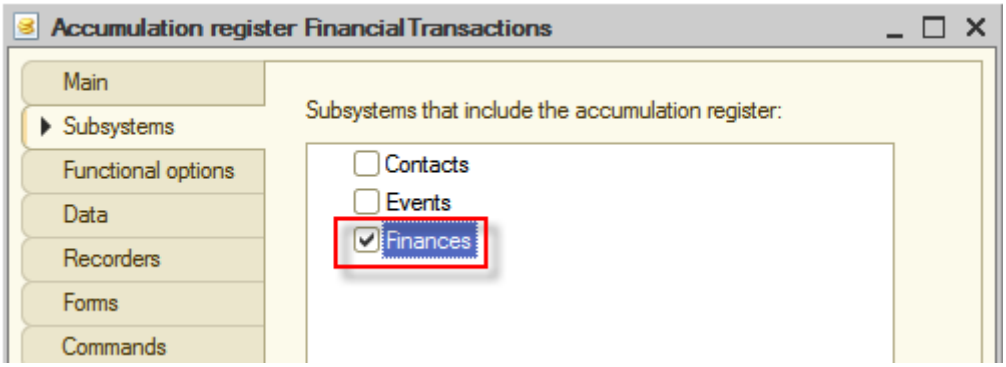

Figure 5-3. The Financial transactions accumulation register

Now add dimensions, resources, and attributes to this register.

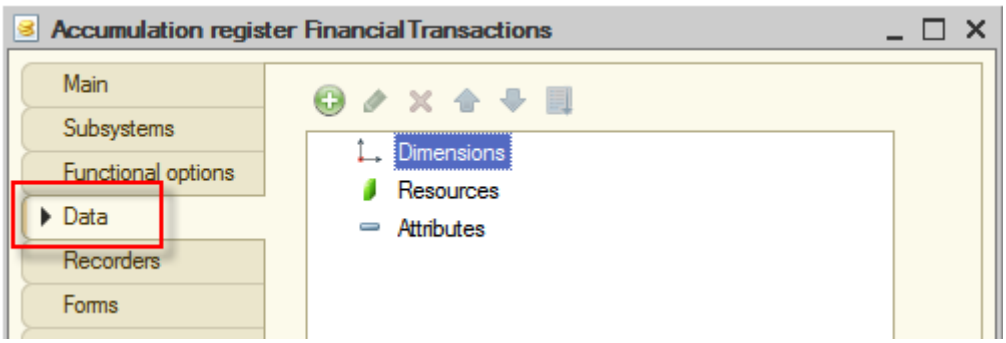

Figure 5-4. The ccumulation register Data tab

Resources are the data that you are going to obtain from the register. For **Financial transactions** you need to know the amount of transactions. Hence, only one resource of **Number** type named **Amount** is required.

Dimensions refer to slices of information that is required to be obtained from the register. It is unlikely that you will be interested only in the overall balance. For example, to create a finance flowchart you may be interested in which events or people bring you most of the money, or which consume the most. To track this, you will need to have two dimensions **Event** and **Person**.

Attributes keep additional information that accompanies each record in the register. For **Financial transactions** you will not use attributes. However, nothing prevents you from adding them to the register in future and then keeping any auxiliary information there.

Add mentioned above dimension and resources. To add them, right-click the **Dimensions** and the **Resources** groups of the accumulation register, and in the context menu click **Add** (Ins).

Now, add a dimension.

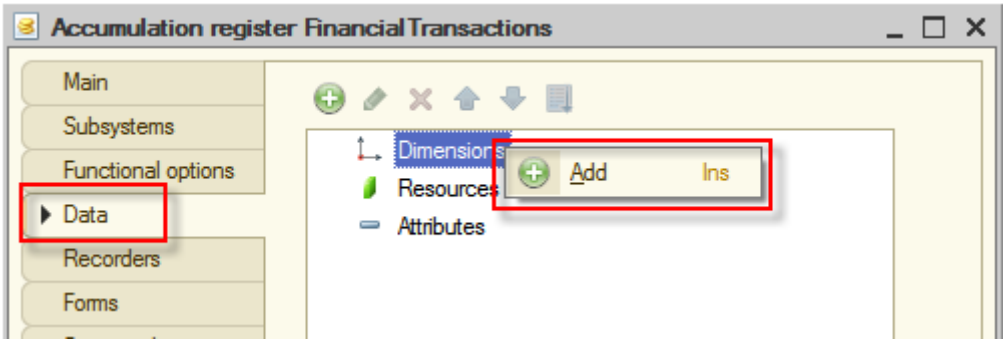

Figure 5-5. Adding a dimension

In the **Properties** window, define that the **Name** of the new dimension is **Event**, and its **Type** is **CatalogRef.Events**.

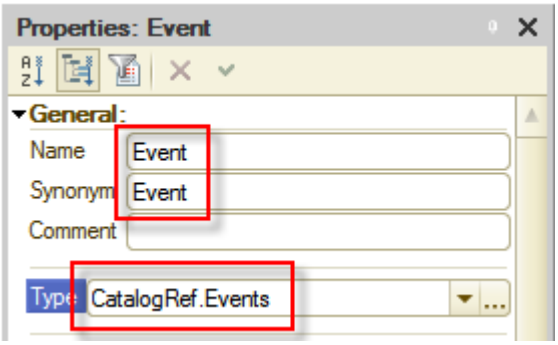

Figure 5-6. The Event dimension properties

Add a second dimension, named **Person**. Type is **CatalogRef.People**.

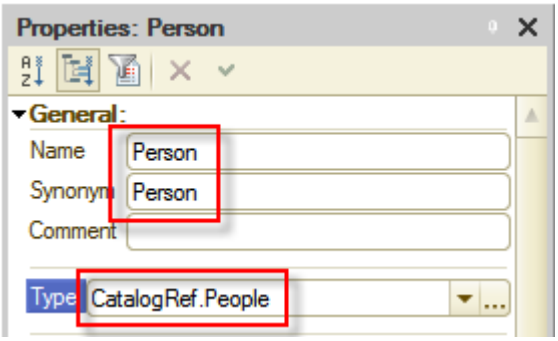

Figure 5-7. The Person dimension properties

Now add the **Amount** resource. Accept default values of **Type** and **Length**, and adjust only **Precision**, increase it from **0** to **2**.

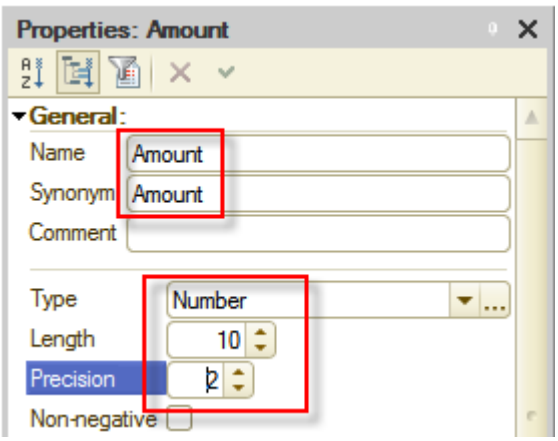

Figure 5-8. The Amount resource properties

As a result, the **Financial transactions** accumulation register will look as follows:

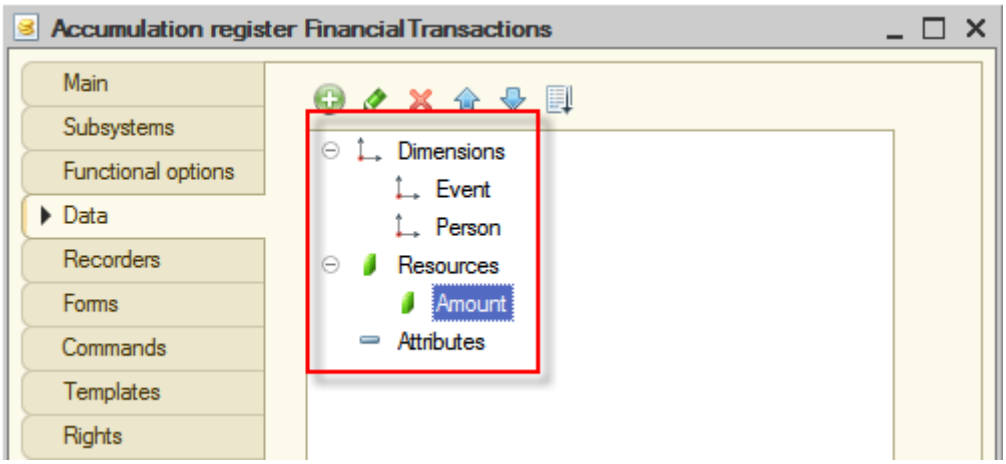

Figure 5-9. The Financial transactions register

Documents

Now, proceed to adding documents that record money income and outgoings. The first document will be **Cash receipt**.

Documents are located in the **Documents** branch of the configuration object tree. Add a new document.

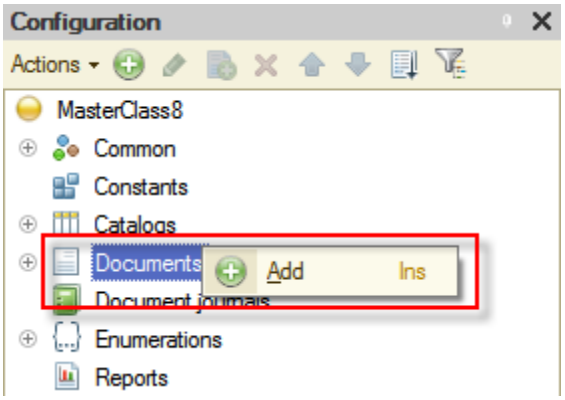

Figure 6-1. Adding a new document

Name it **CashReceipt**, and include it to the **Finances** subsystem, and then click the **Data** tab.

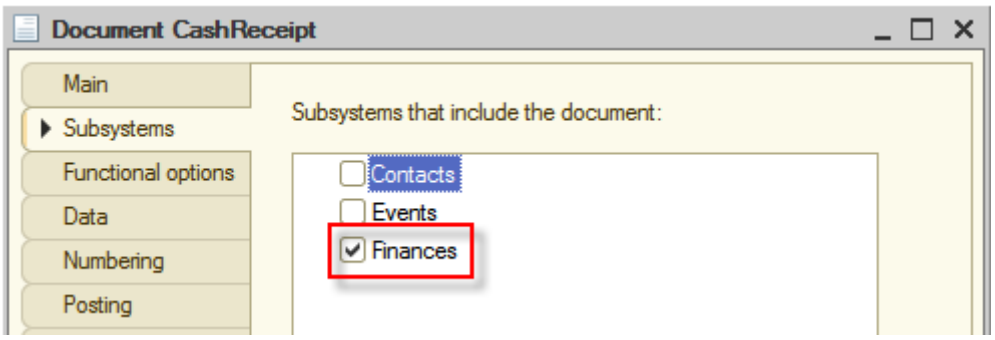

Figure 6-2. Creating the Cash receipt document

Each document, by default, has two attributes, **Number** and **Date** that indicate the sequential number of this document and the date when it was

created. However, in addition to these attributes, it is usually required to have other information regarding specific business activity in the document. For example, in this master class users would like to understand what was the source of money: who or which event. In addition to simplify data input, make possible the same document to register several similar events. For example, it is convenient to group all receipts for one day in one document; this will allow a user to supervise documents that this user enter.

As you might have guessed already, a **tabular section** will help us to accomplish this.

Create the **Receipts** tabular section.

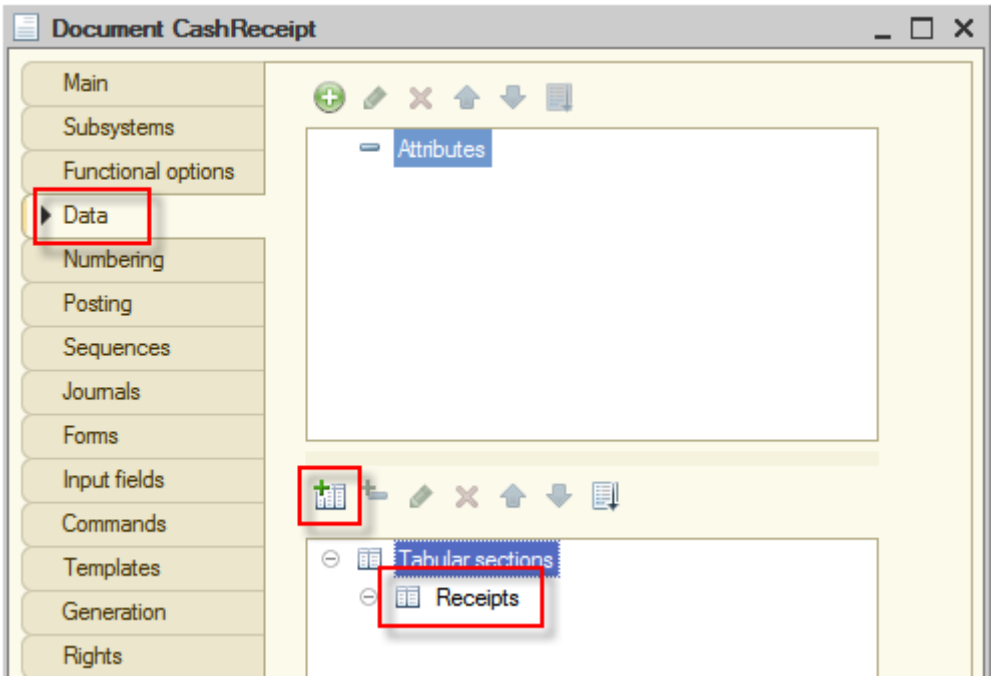

Figure 6-3. Creating the Receipts tabular section

In order to be able to track events and people, you need to create two attributes in the newly created tabular section, **Event** and **Person**. To track the amount of transactions, create the **Amount** attribute.

- **Event** with **Type** equals to **CatalogRef.Events**.
- **Person** with **Type** equals to **CatalogRef.People**.
- **Amount** with **Type** equals to **Number**, **Length** equals to 10, **Precision** equals to 2, and select the **Non-negative** check box.

Because of these actions, **Data** tab of the document will look as follows:

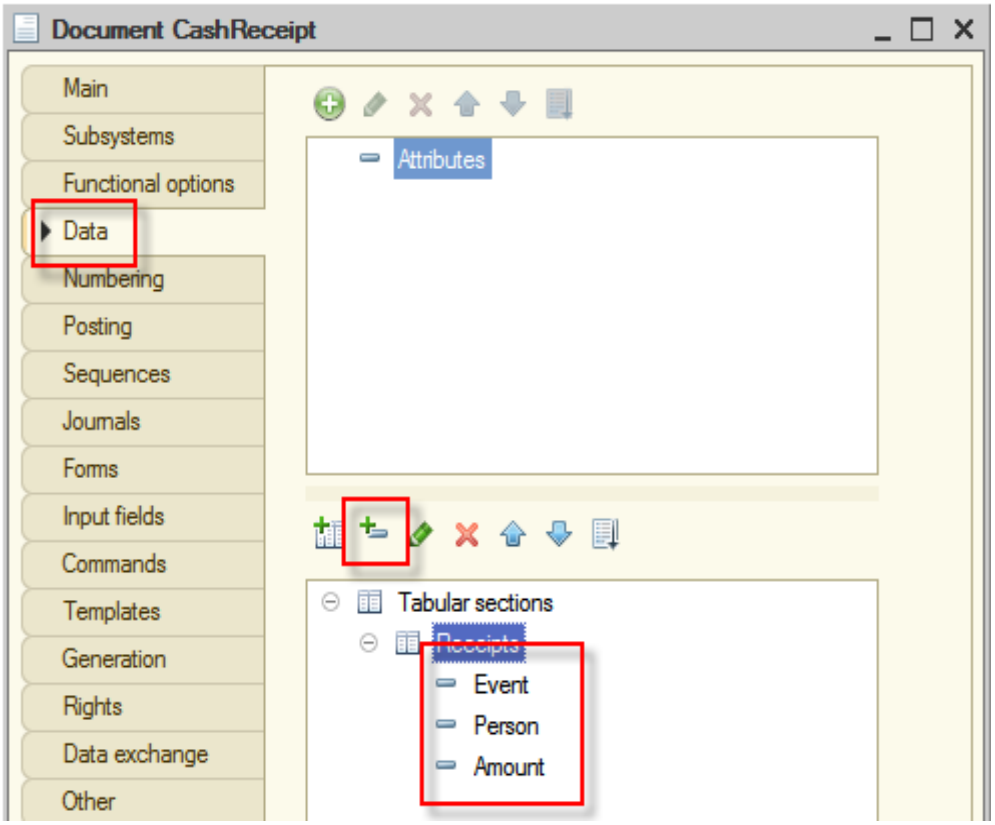

Figure 6-4. Adding attributes to the tabular section

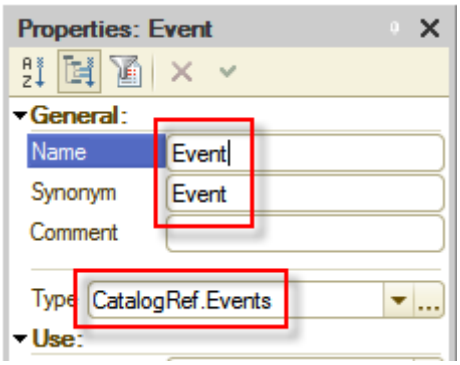

Figure 6-5. Event attribute

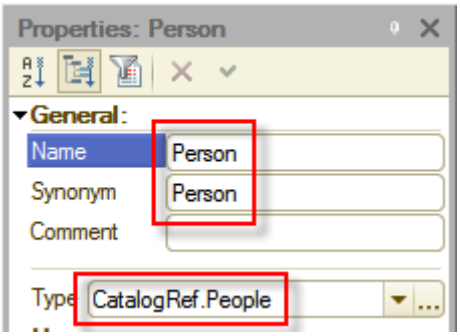

Figure 6-6. Person attribute

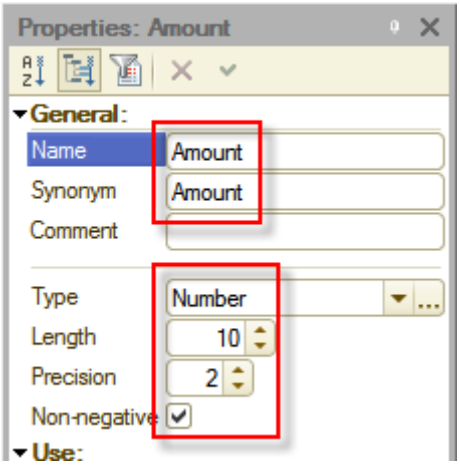

Figure 6-7. Amount attribute

To finish configuring this document, click the **Posting** tab.

As it is said above, documents write data to registers, and reports get this data from registers, and display in a convenient form to users.

Document register records are records that documents make in registers; in this master class records will be made in **Financial transactions** register.

Expand **Accumulation registers** node and check **Financial transactions** accumulation register. By doing this you define that **Cash** receipt document will write records in the **Financial** transactions register. Then click **Register records wizard** and the platform will assist you in creating the data-writing algorithm for registers.

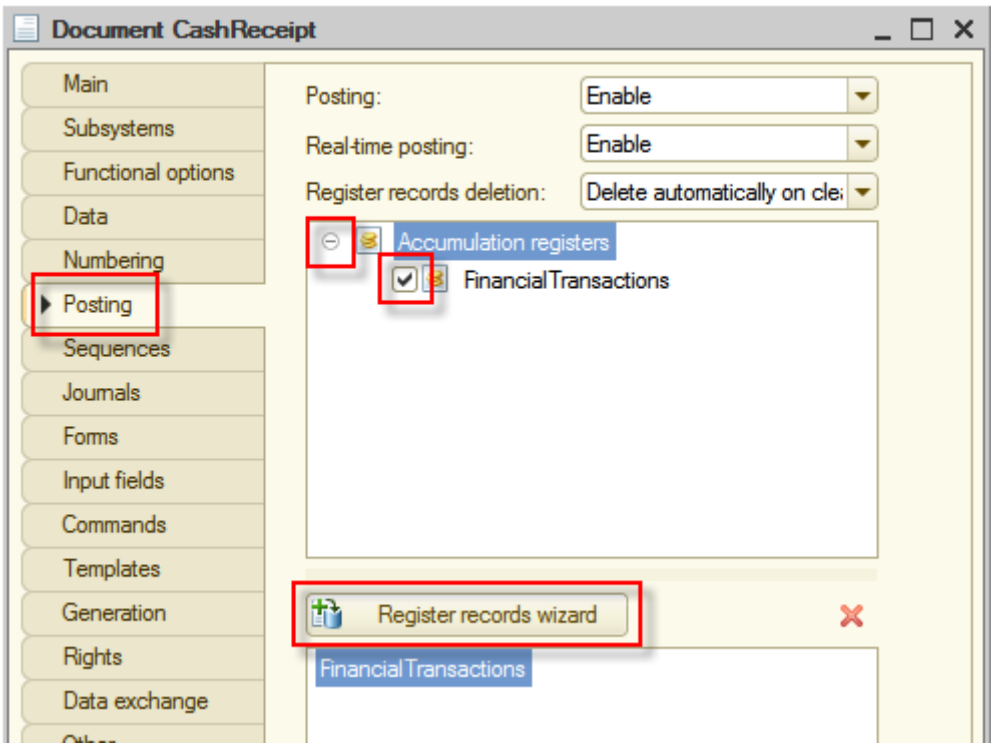

Figure 6-8. Defining document register records

The **Register Records Wizard** window will open. Leave **Register record type** unchanged, equal to **Receipt**, since this document is a receipt of money. Then select **Receipts** in the **Tabular Section** field and click **Fill Expressions**.

In the table at the bottom of the window, see that the platform automatically found the correspondence between accumulation register attributes and document attributes. The value of the **Event** tabular section attribute of the document will be placed in **Event** register dimension. The value of **Person** tabular section attribute of the document will be placed in the **Person** register dimension. The value of the **Amount** tabular section attribute will be placed in the **Amount** register resource.

After filling out and verifying the correspondence table, click **OK**.

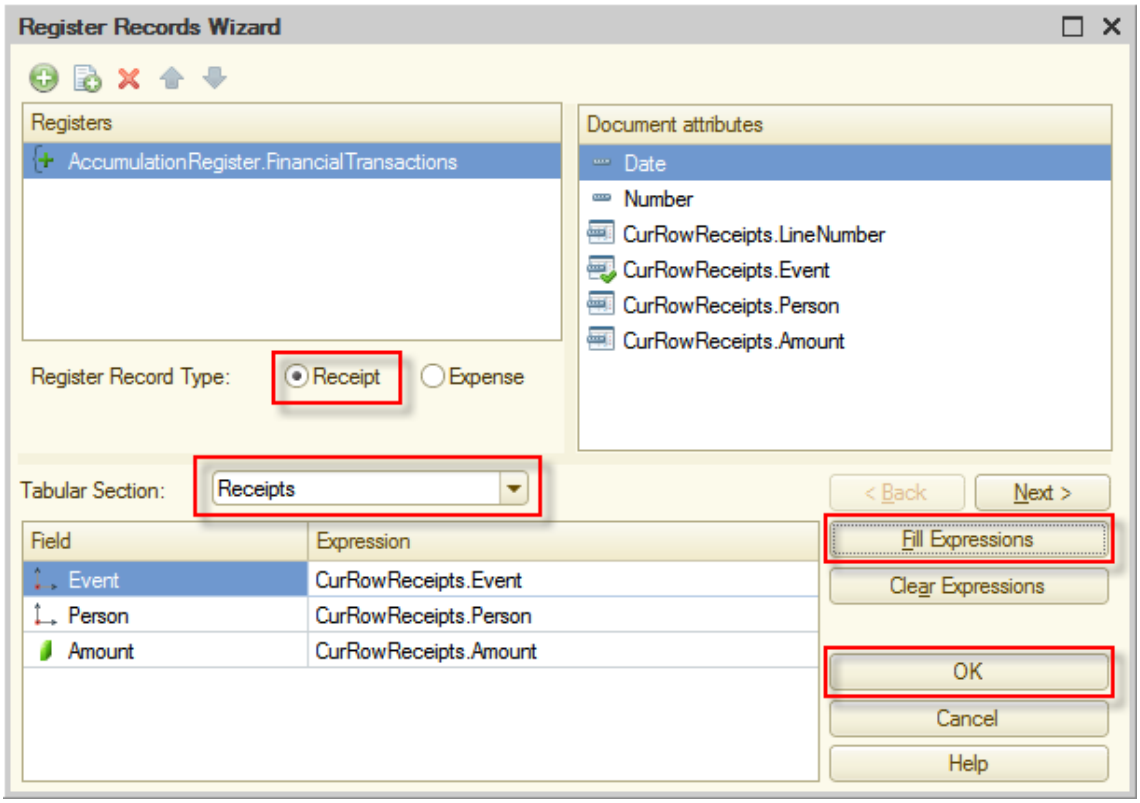

Figure 6-9. Register Records Wizard

The wizard will generate the procedure for writing document register records, in other words, for posting the document, and then display it on the screen.

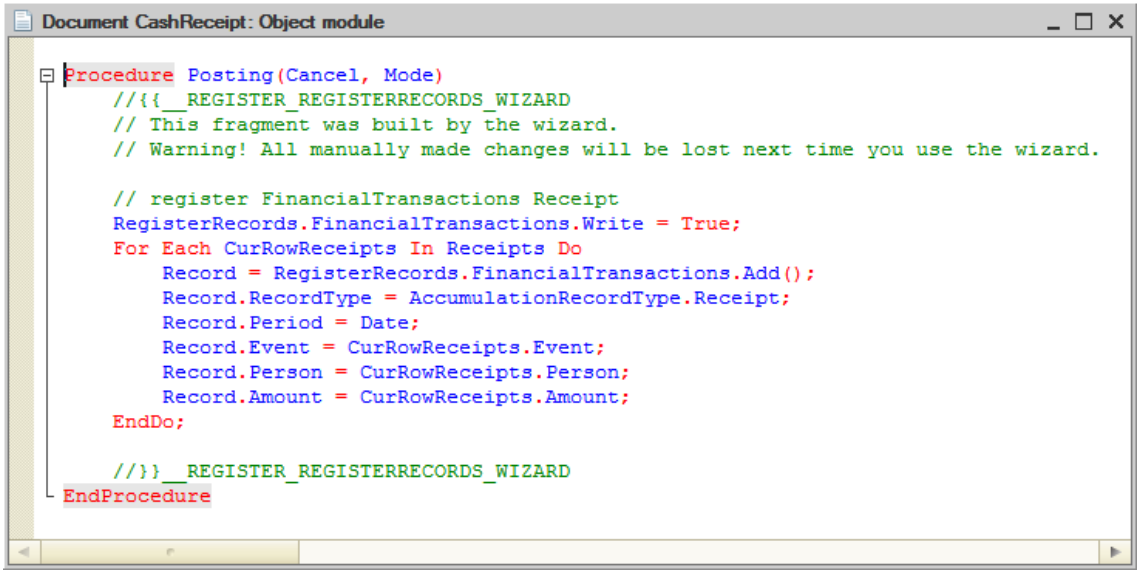

Figure 6-10. Procedure for posting the document

As you can see, the procedure is quite simple. For each row of the tabular section in the document, the new record is created in the register, and this record contains the data from the row.

You could write this algorithm yourself, but to minimize the amount of work it is better to utilize **Register Records Wizard**. The wizard generated this code automatically, which is mentioned in comments at the begin and the end of the procedure.

Now start the application in the 1C:Enterprise mode by clicking **Start Debugging** (F5), and using the **ImportXMLData83.epf** data processor import demo data into the **Cash receipt** document from the **06-CashReceipt.xml** file.

Click the **Finances** tab and open the list of **Cash receipt** documents.

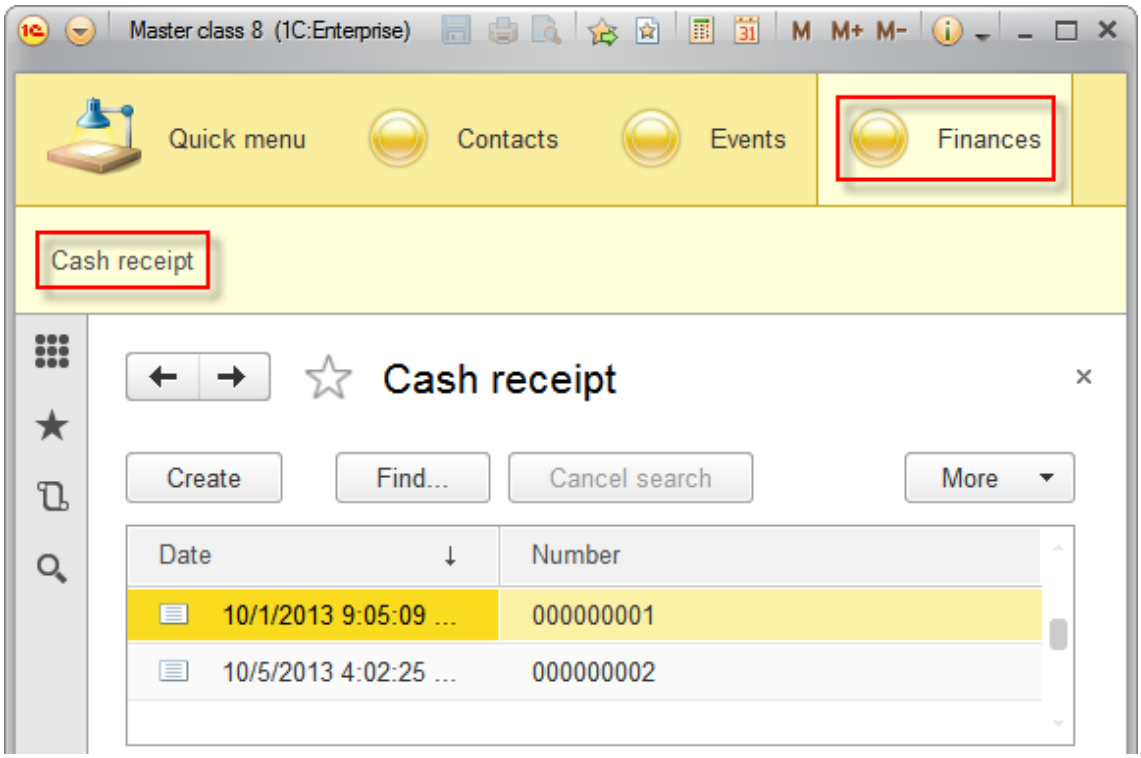

Figure 6-11. Cash receipt documents in the 1C:Enterprise mode

For now, documents are just imported, and they do not affect reports yet. In order for the data to be displayed in reports, documents have to be posted. When documents are posted, the data is being added to the **Financial transactions** accumulation register.

Open the first document. To post the document, click **Post and close**.

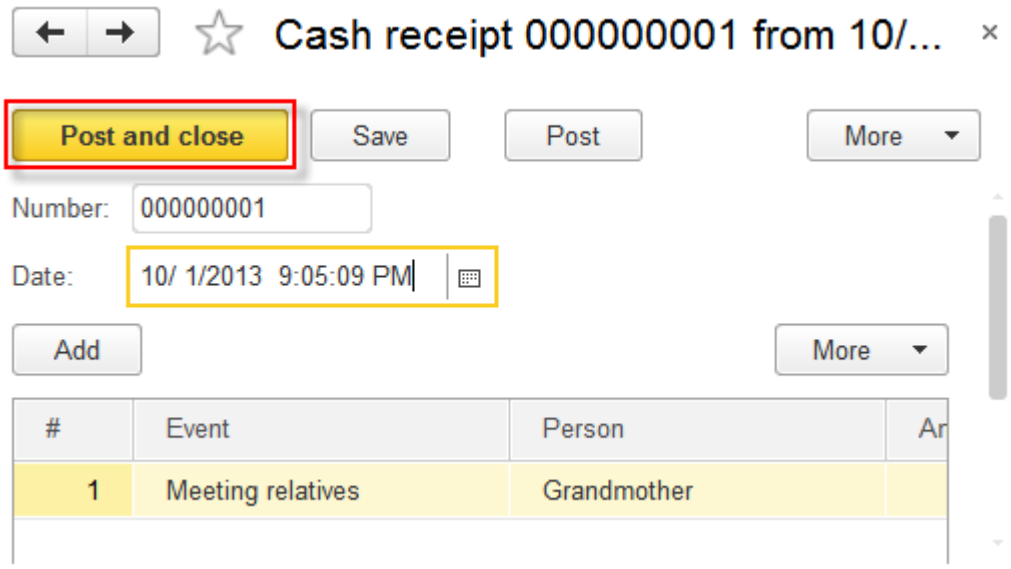

Figure 6-12. Posting the Cash receipt document

Successful document posting will display a notification on bottom right of the screen.

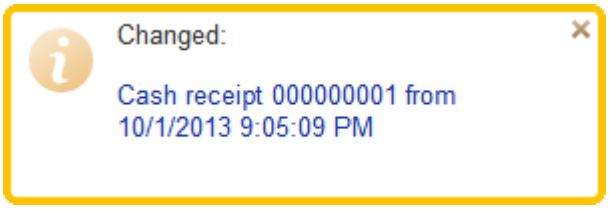

Figure 6-13. Successful posting of the Cash receipt document

The fact that the document posting was successful is displayed as a check mark on the icon of this document in the list of documents.

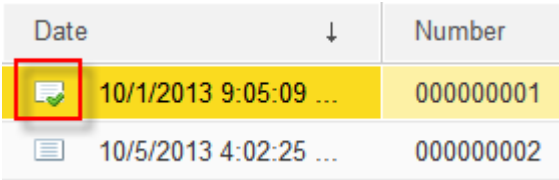

Figure 6-14. Posted document in the list of documents

To verify the way the document was posted, meaning the required information is recorded in the **Financial transactions** accumulation register, open **Main menu** of the application, click **All functions...**, then expand accumulation registers node, then click **Financial transactions**, and then click **Open**.

Notice: The **All functions...** menu item is available by default if an application is started in the debug mode from the Designer mode.

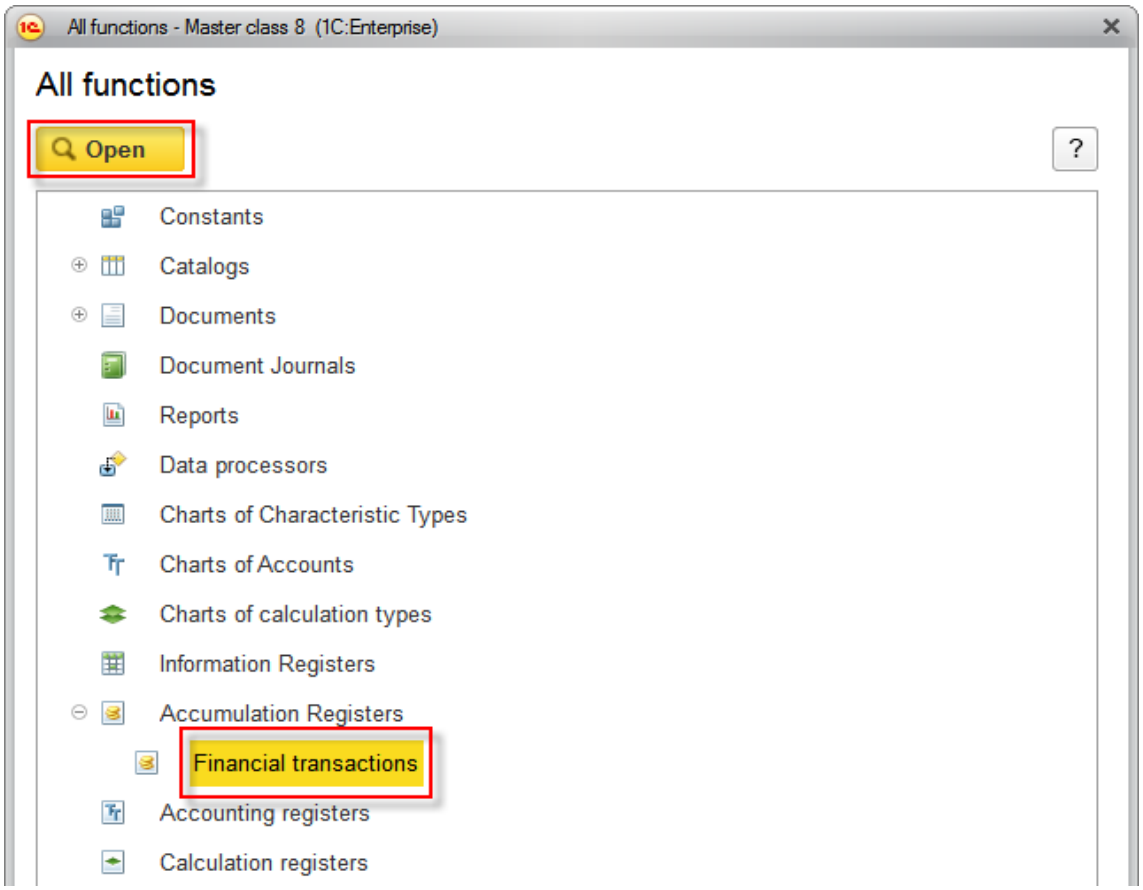

Figure 6-15. All functions

The list form of the **Financial transactions** accumulation register will open. For now, the register contains only one record that belongs to the first posted document. By double-clicking this record, you can always open the original document.

Notice the + icon, which means that the values of resources of this record are added to total values of resources.

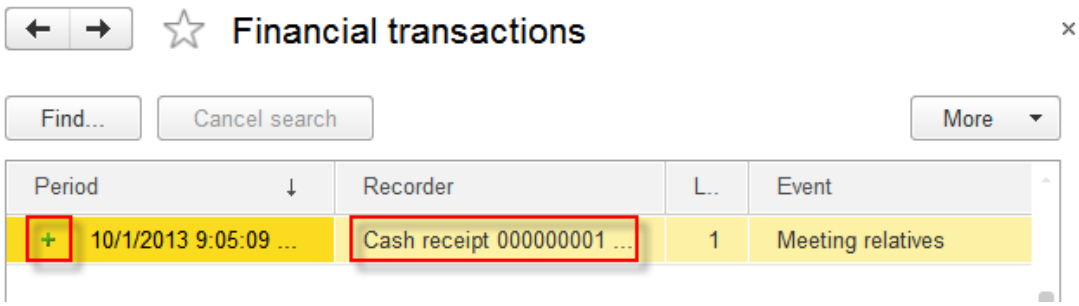

Figure 6-16. Accumulation register record

Switch to **Cash receipt** document list and post the second document. After the posting, return to the list of records of **Financial transactions** register. See that a new entry, belonging to the second document, appeared. You can update the form with F5 key.

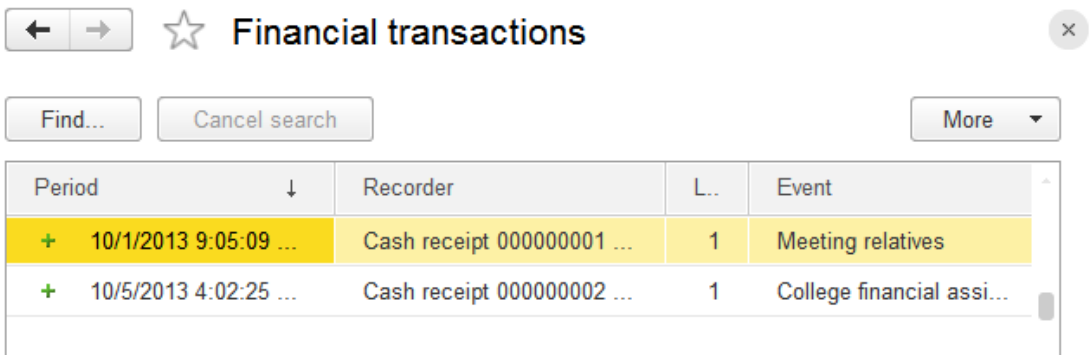

Figure 6-17. Accumulation register records

Finished with **Cash receipts**. However, users also need to track outgoings. Go back to the Designer mode. It seems obvious, that it is necessary to create the **Cash payment** document. You could easily create it manually in the same way as you did already with the existing **Cash receipt** document because compositions of two documents are identical. Another option, however, is to use the 1C:Enterprise 8 feature allowing you to create new configuration objects based on existing ones by simply copying them.

To do this, right-click the existing **Cash receipt** document in the list of metadata objects and click **Add by cloning** (F9).

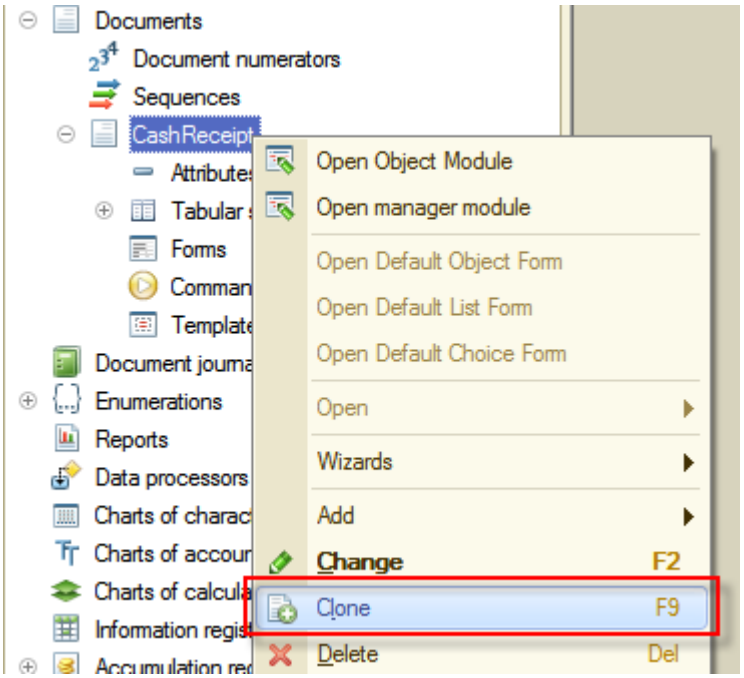

Figure 6-18. Adding by cloning an existing document

Once you click this button, a new **CashReceipt1** document appears. It is going to be an exact copy of **CashReceipt**.

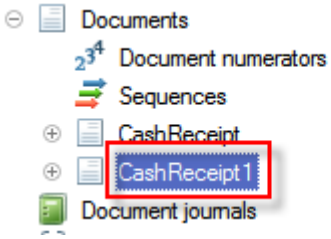

Figure 6-19. A copy of the Cash receipt document.

The only thing left to do is to rename the document and edit some of its properties. Do it.

First, open this document in the document editor editor and change the name of the document to **CashPayment**. Moreover, include the new document to the **Finances** subsystem.

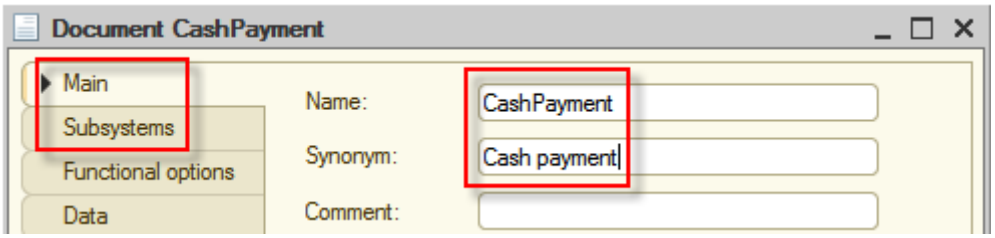

Figure 6-20. The Cash payment document

On the **Data** tab, using **Properties**, change the name of the tabular section to **Expenses**. Keep the remaining attributes unchanged as they meet project requirements.

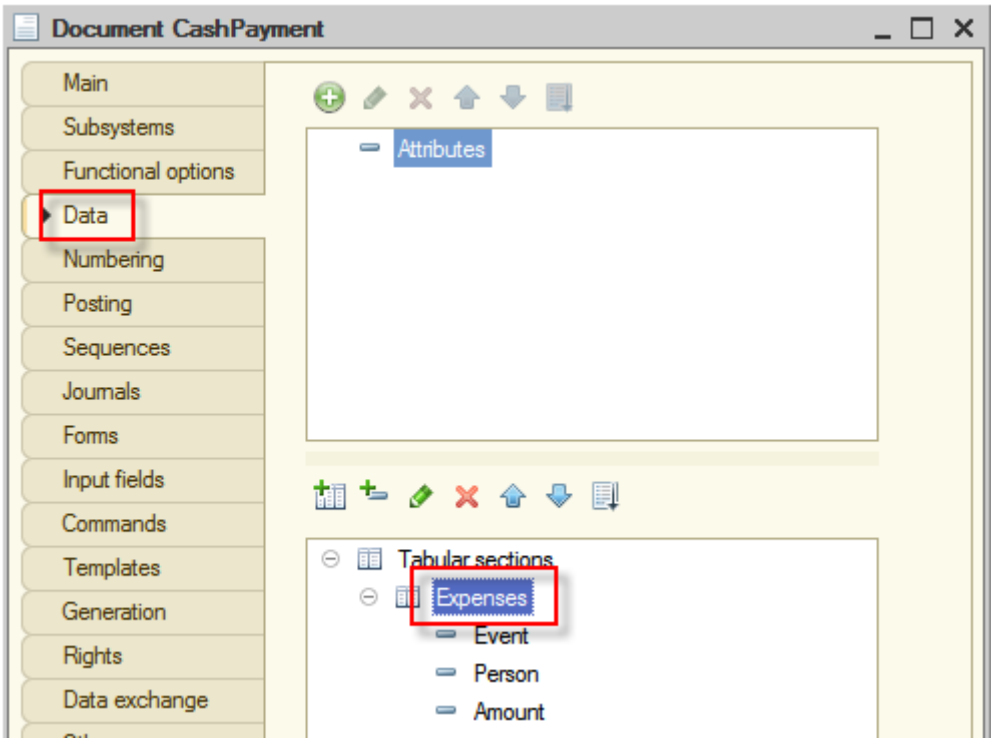

Figure. 6-21. Renaming the tabular section

Click the **Posting** tab. Here you will change the document posting. For now the posting algorithm is copied from the **Cash receipt** document. That document keeps income but in the newly created document is designed to keep outgoings.

Open **Register Records Wizard**. The platform will prompt you to confirm that the existing script will be replaced. Click **Yes**.

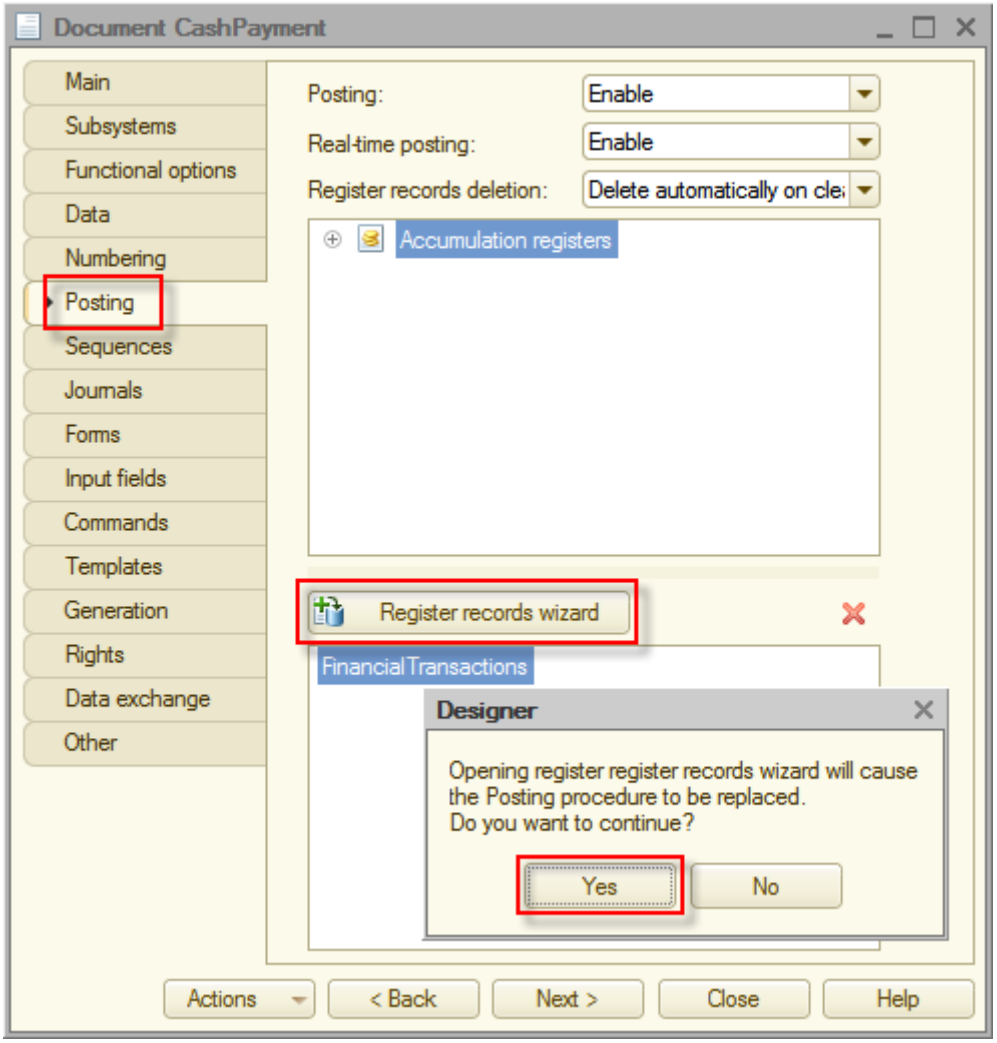

Figure 6-22. Opening Register records wizard

Here repeat all the steps that you performed for **Cash receipts** document. You only need to change the **Register Record Type** to **Expense** and select the tabular section **Expenses**. Then click **Fill Expressions**, and to update the module click **OK**.

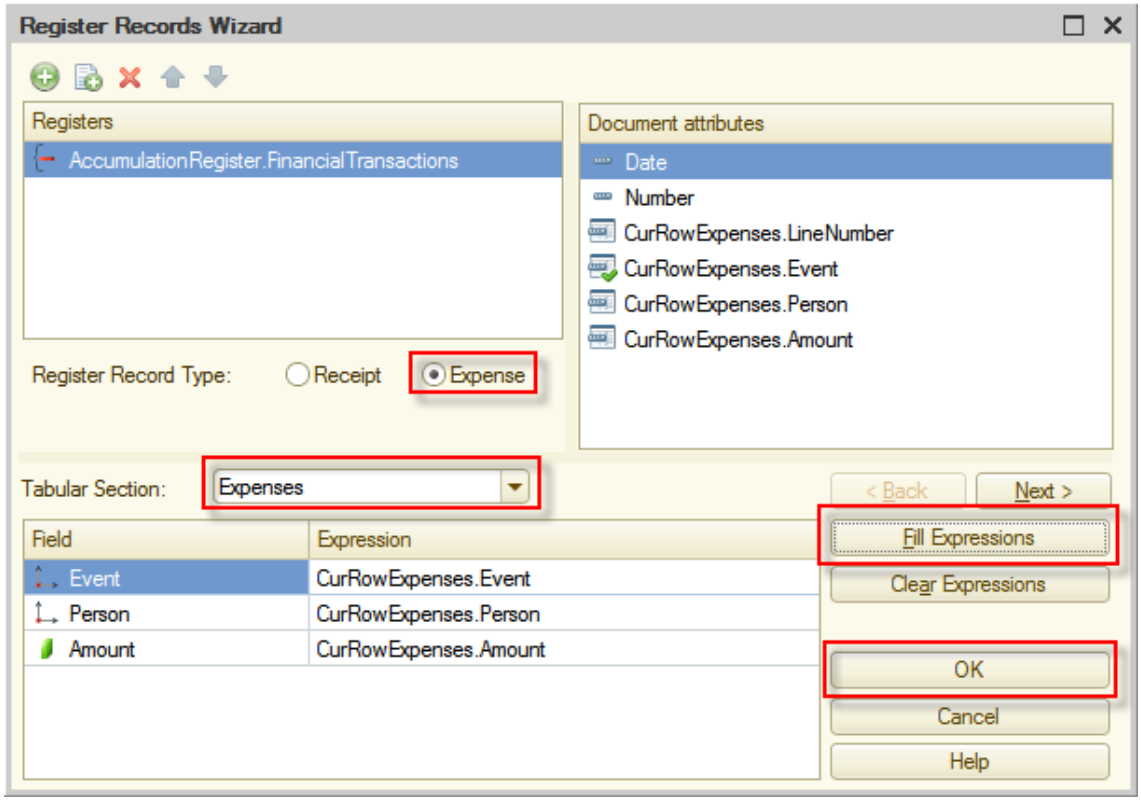

Figure 6-23. Register Records Wizard

You can see that there are only two differences between two documents, the name of the tabular section and the type of the accumulation register record.

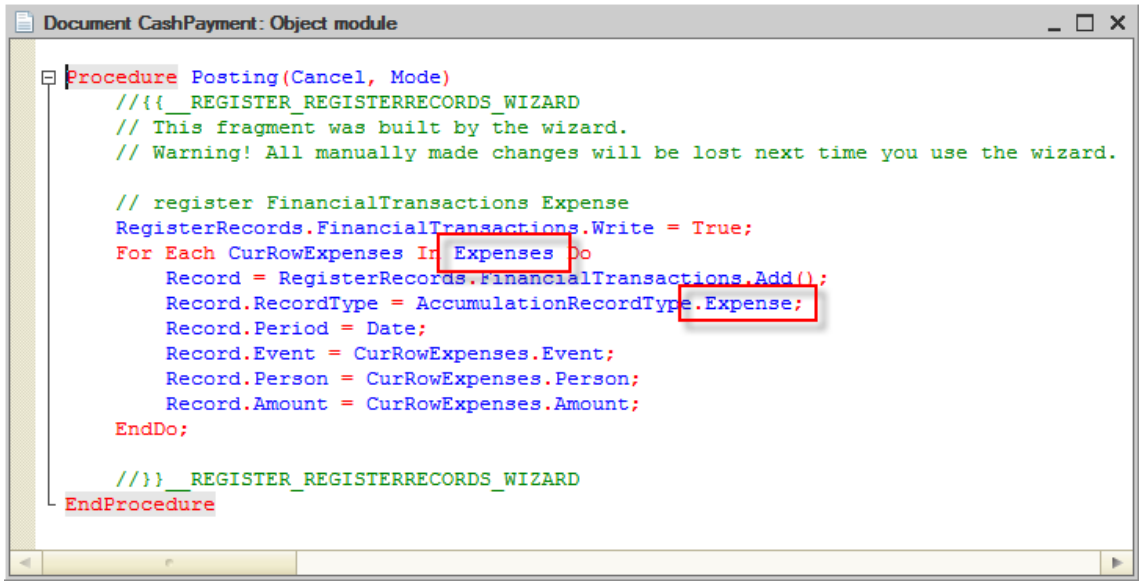

Figure 6-24. Posting procedure of the Cash payment document

Start the application in the 1C:Enterprise mode and, using the data processor, import the **Cash payment** data from the **07-CashPayment.xml** file. Then, switch to the **Finances** section and open the list of **Cash payment** documents.

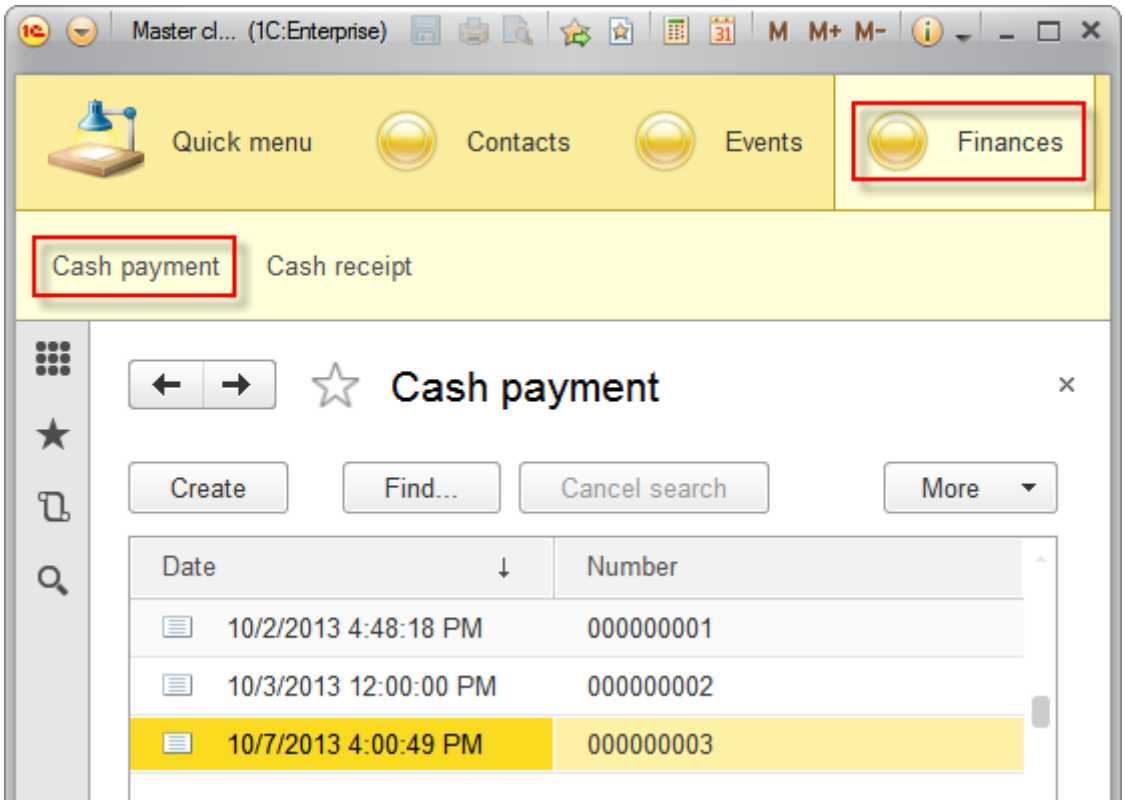

Figure 6-25. The Cash payment document list

Post these three documents and then verify that the data is recorded to the **Financial transactions** accumulation register.

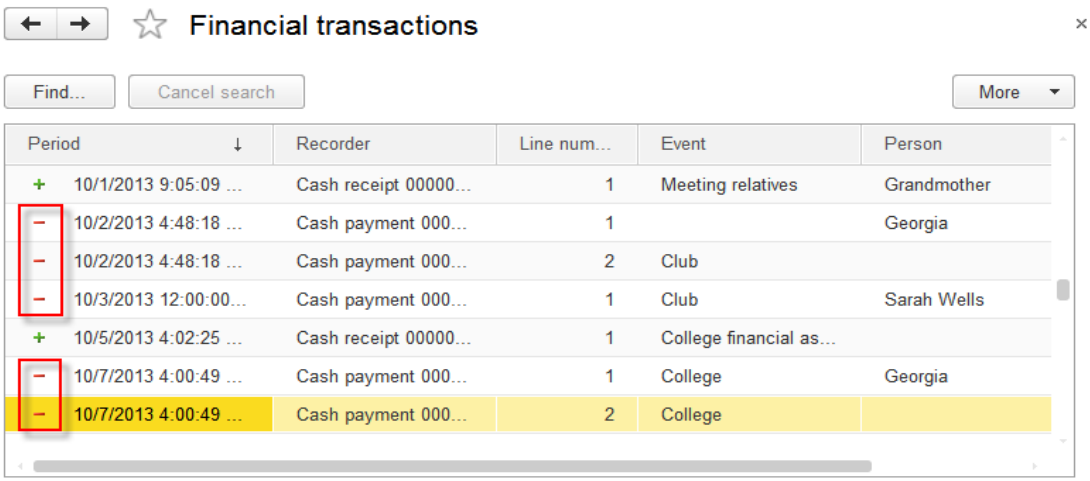

Figure 6-26. Records in the accumulation register

It is not difficult to understand that records in the register, belonging to **Cash payment** documents are marked with − icon. In addition, it is clear that a few lines of a document are recorded in the register in the form of individual entries. Note the **Line number** column.

At this point, the accounting part of the application is complete. Users are now able to fill a list of acquaintances and enter cash receipts and payments.

However, simple accounting is not enough. Users would also like to receive the information from the application in a convenient form. For this purpose, there are reports.

Reports

The next step in the development of the applied solution will be creation of convenient reports on activities. Reports as well will be created without any programming, using only visual design tools.

Reports are located in the **Reports** branch of the **Configuration** object tree. Let us add a new report.

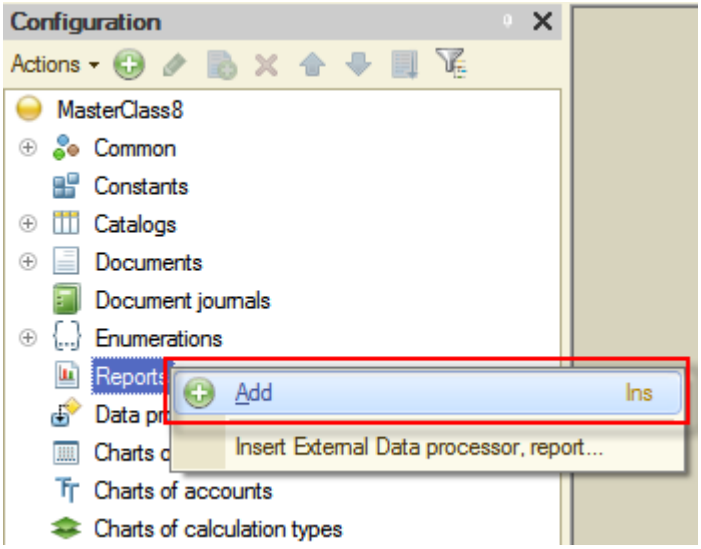

Figure 7-1. Creating a new report

Name the new report **Expenses** and then click **Open data composition schema** of this report.

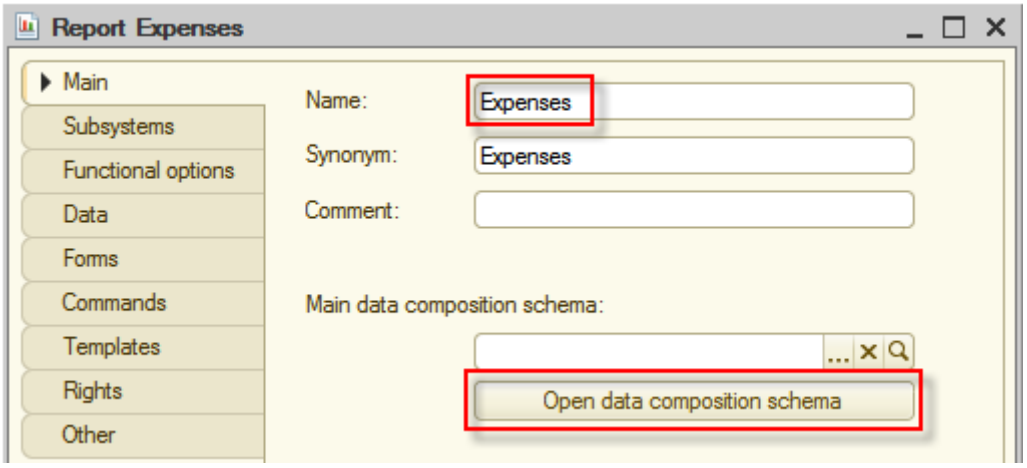

Figure 7-2. Opening the data composition schema of the Expenses report

Since this is a new report, it does not contain any data composition schema, the platform will open **Template Wizard** and suggest you to create a new template. That template will contain a data composition schema. Agree by clicking **Finish**.

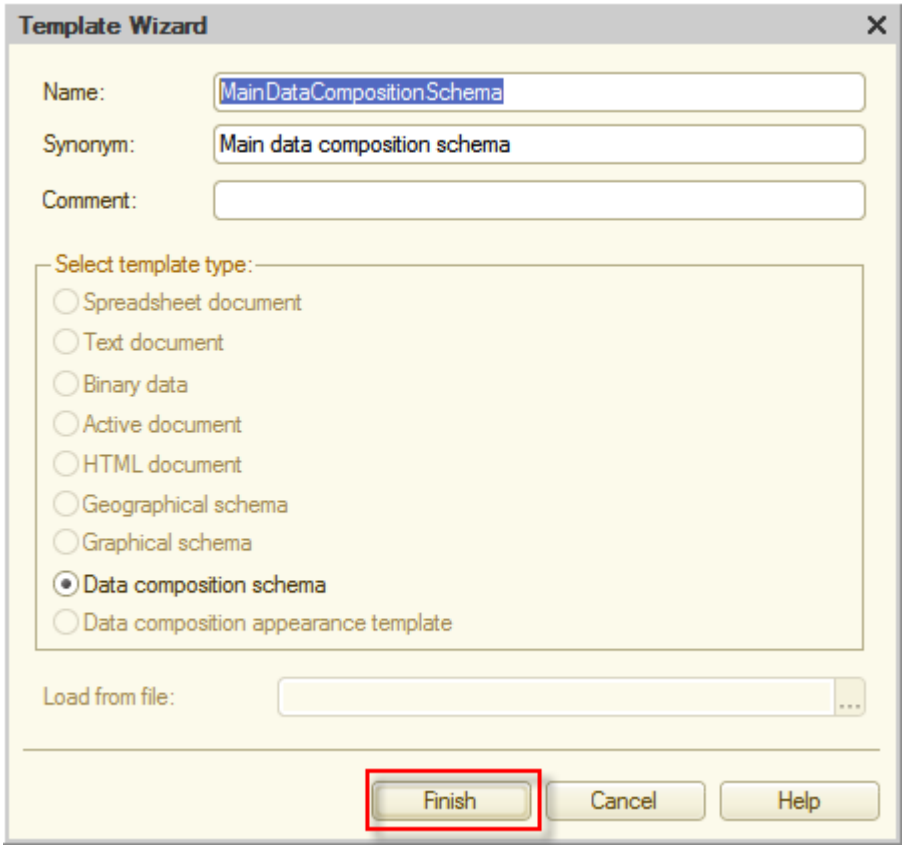

Figure 7-3. Template Wizard

After that, the platform will open the **Data composition schema** editor. For now, this composition schema is empty.

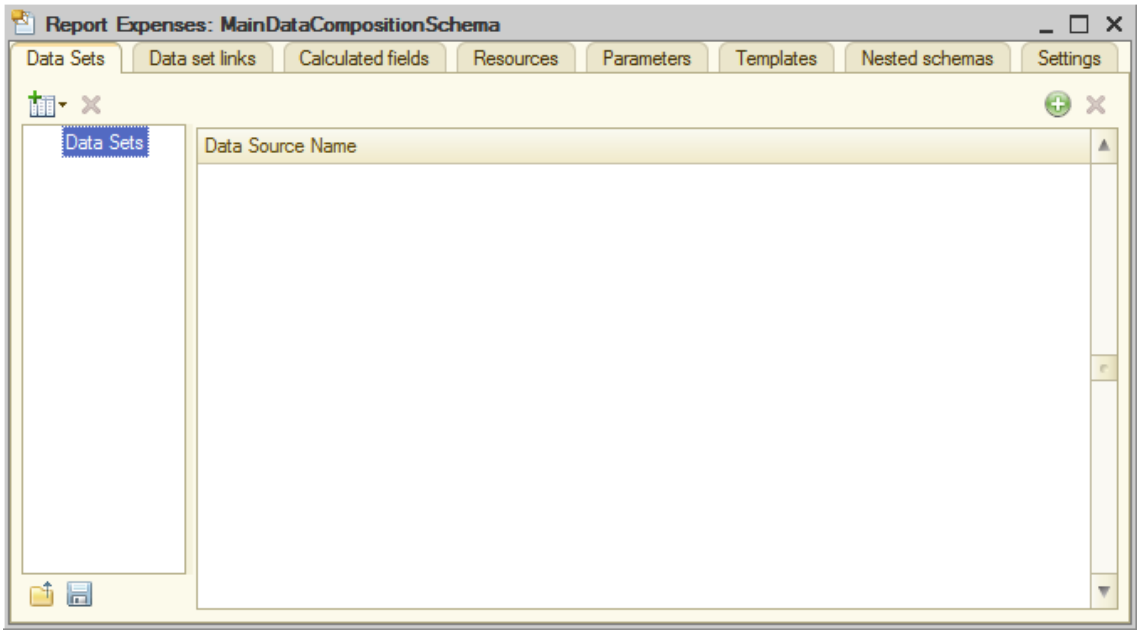

Figure 7-4. The empty data composition schema

Now you will define sources of data that the report will use to retrieve data, and define the structure of the report.

Add a **Data set** of type **Query**. In other words, the data for this report will be retrieved from the database, managed by 1C:Enterprise.

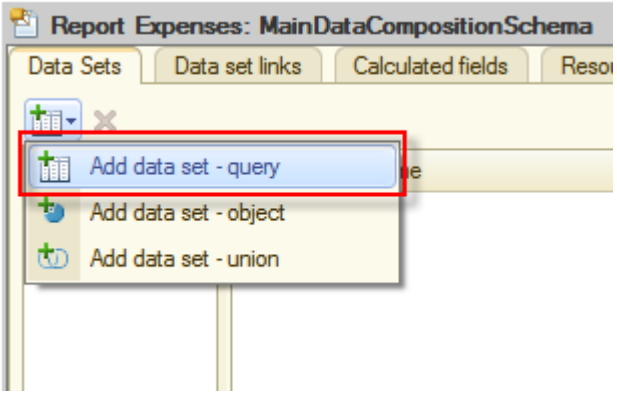

Figure 7-5. Creating a new query

You can enter a query manually or use **Query builder**. Let us use second way, for that click **Query Builder...**

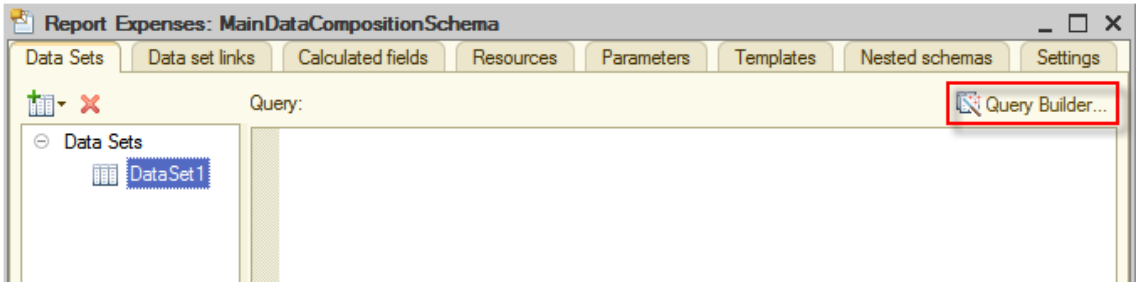

Figure 7-6. Opening Query builder

In the **Query builder** window on the left there is a list of tables that can be used for retrieving data. Expand the **AccumulationRegisters** branch and double-click to select **FinancialTransactions.BalanceAndTurnovers** table.

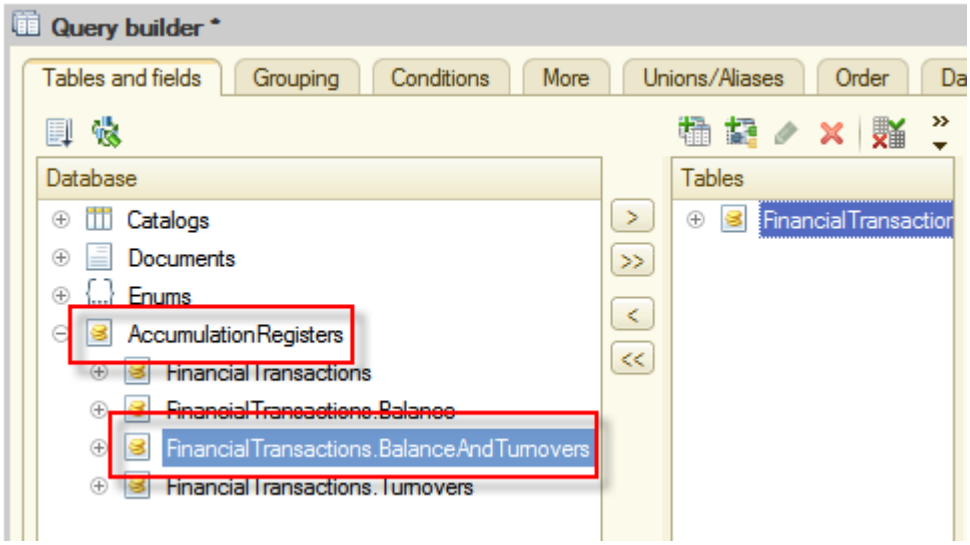

Figure 7-7. Selecting the FinancialTransactions.BalanceAndTurnovers table

In the **Tables** panel that is in the middle, select the **FinancialTransactionsBalanceAndTurnovers** table and click **Add all fields** >>. All fields of this table will be selected as query fields.

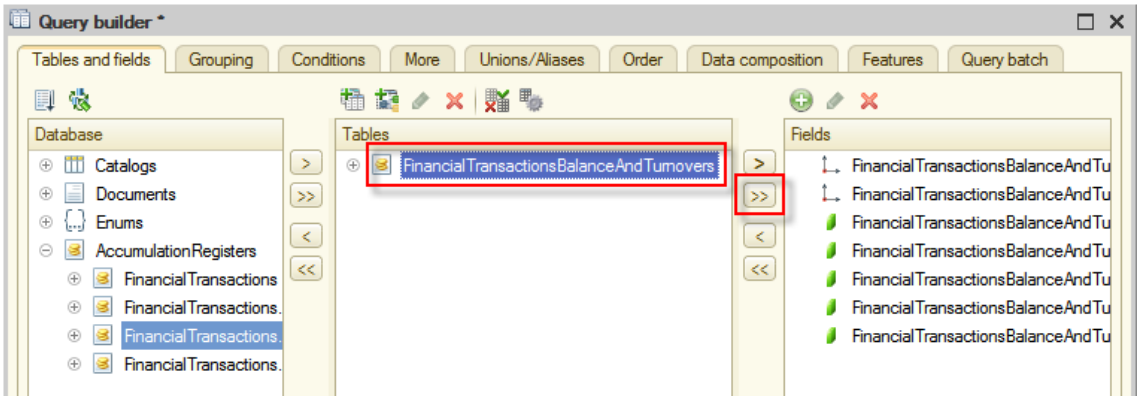

Figure 7-8. Selecting data for query

The query is configured, click **OK**.

The wizard will generate the query text and automatically populate fields of the data composition schema.

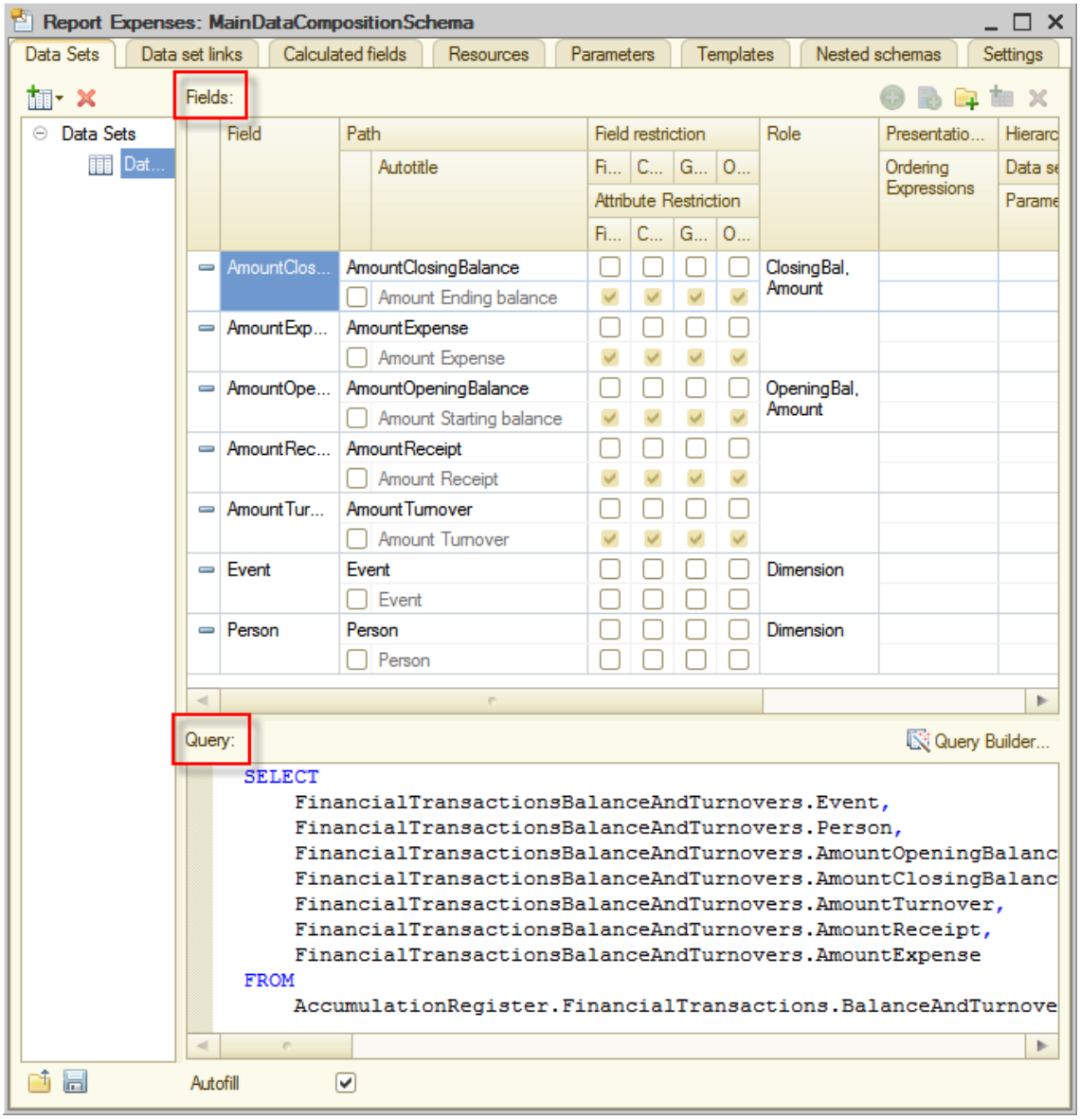

Figure 7-9. Data composition schema fields

Click the **Resources** tab. Here, using double-clicks select following fields:

- **AmountClosingBalance**
- **AmountOpeningBalance**
- **AmountTurnover**
- **AmountReceipt**
- **AmountExpense**

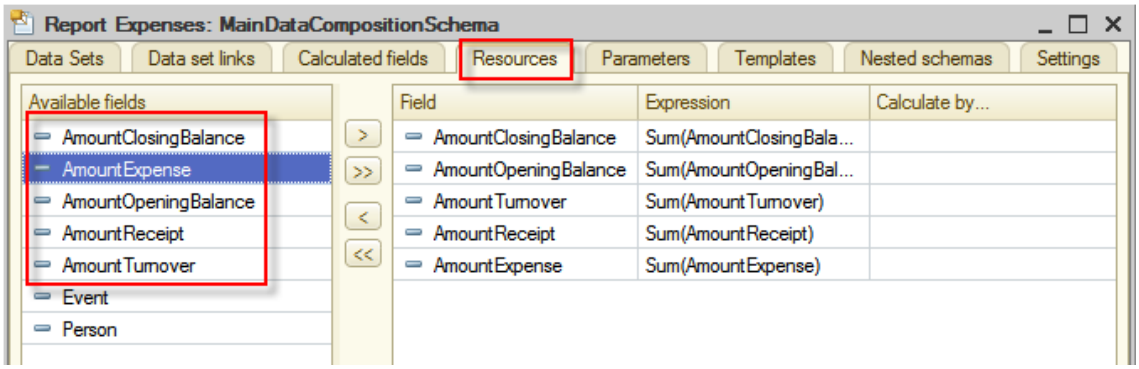

Figure 7-10. Data composition schema resources

Click the **Settings** tab. Here, in order to create the structure of the report, you will use another wizard. To do this, click **Open settings wizard** .

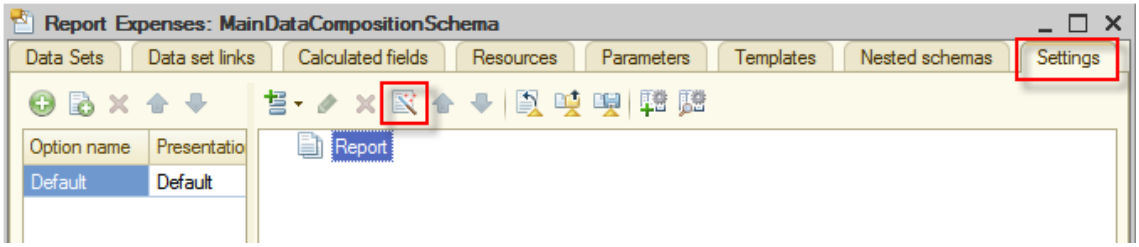

Figure 7-11. Opening Data Composition Settings Wizard

Data Composition Settings Wizard allows you to quickly create several simple variants of the report. For this tutorial purposes, **List** is a good choice. Therefore, do not change anything, and just click **Next**.

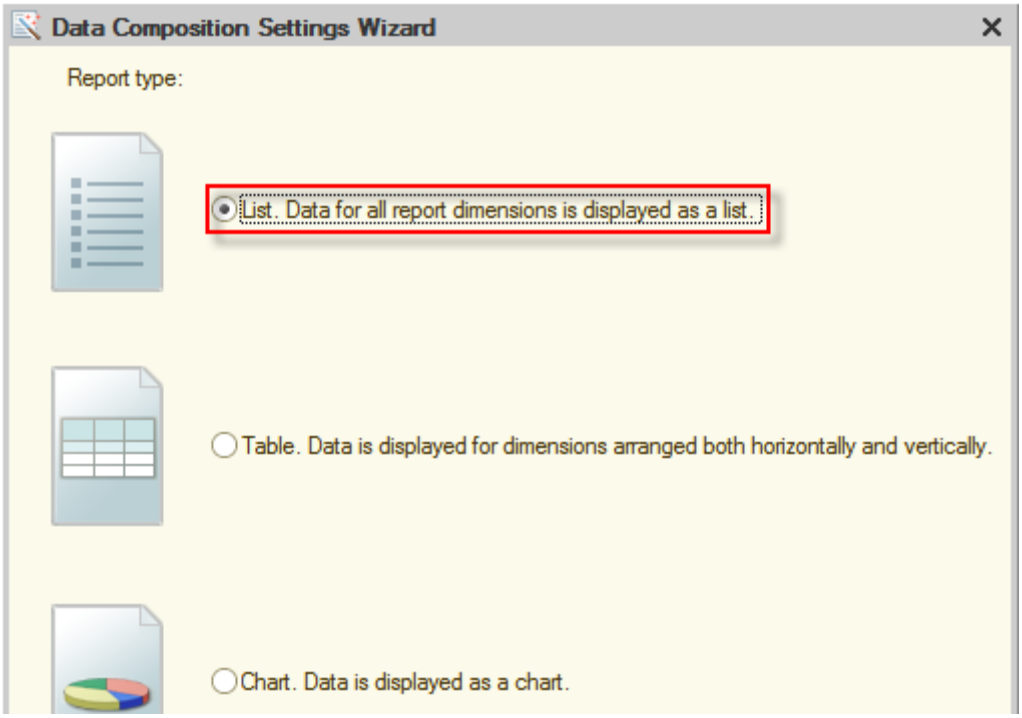

Figure 7-12. Creating a report in the form of list

On the next step, select fields that will be displayed in the report. Double-click the following fields in **Available fields** list:

- **Person**
- **Event**
- **AmountTurnover**
- **AmountReceipt**
- **AmountExpense**

Then click **Next**.

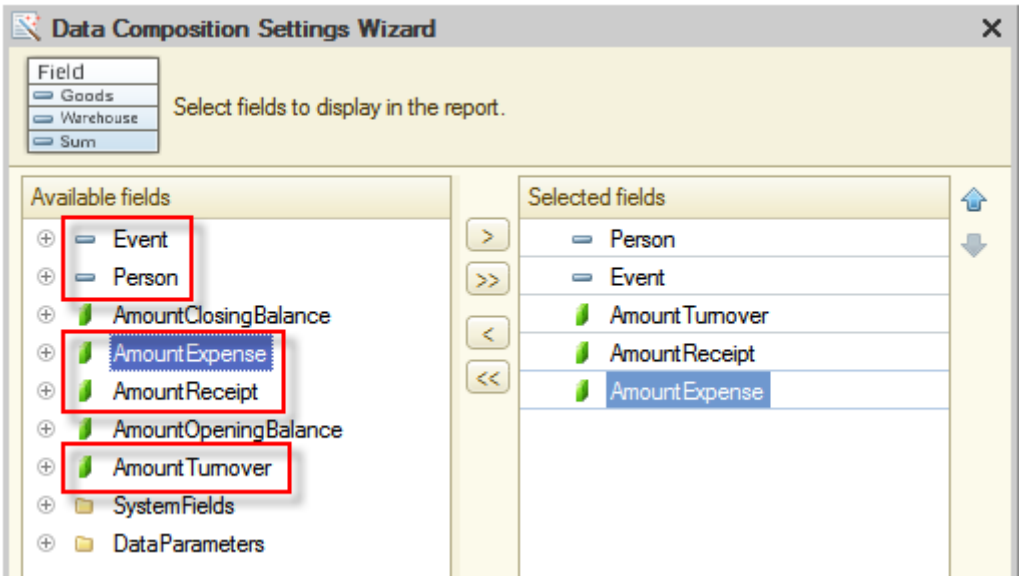

Figure 7-13. Selecting report fields

Now we need to select fields that will group the data in the report. Users would like to see reports on each acquaintance, and on each event associated with the acquaintance. To achieve this, the list of available fields double-click on both **Person** and **Event** fields to select them as grouping fields.

After that, click **OK**. You are now done with defining the structure of the report.

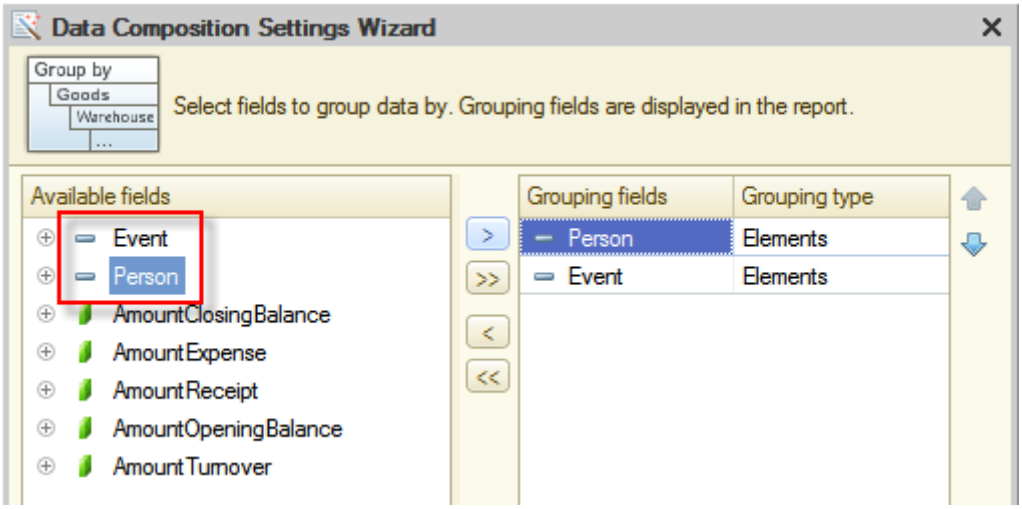

Figure 7-14. Selecting group fields of the report

The platform will now show the structure of the report.

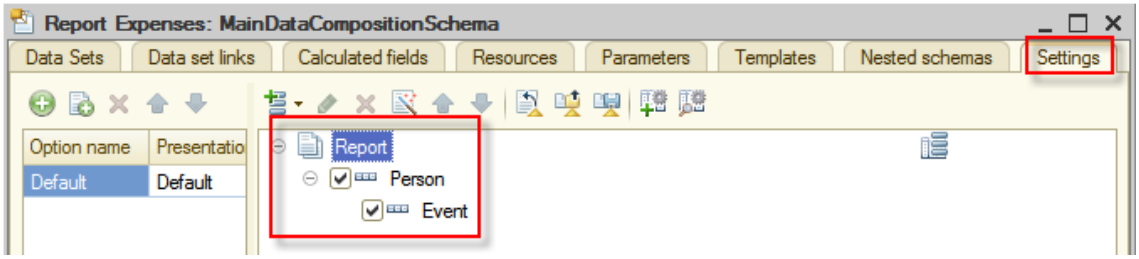

Figure 7-15. Completed structure of the report

It is almost done. Now, give a user an ability to arbitrarily set the report period. This will be useful when the register will contain large amount of records accumulated after a long period of using this application. This is easy to do as under the structure of the report, there are two parameters available, named **Begin of period** and **End of period**.

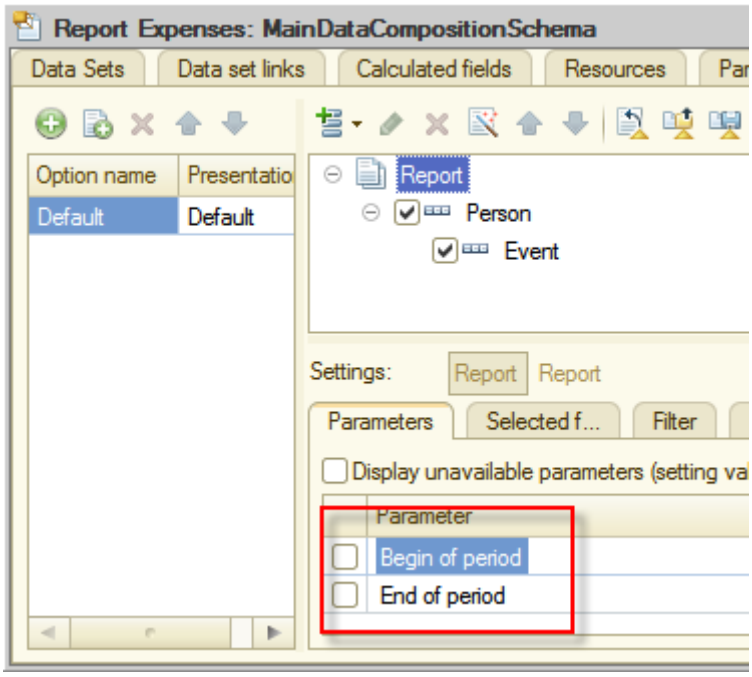

Figure 7-16. Report parameters

Right-click each of parameters and then select **Custom settings item property**.

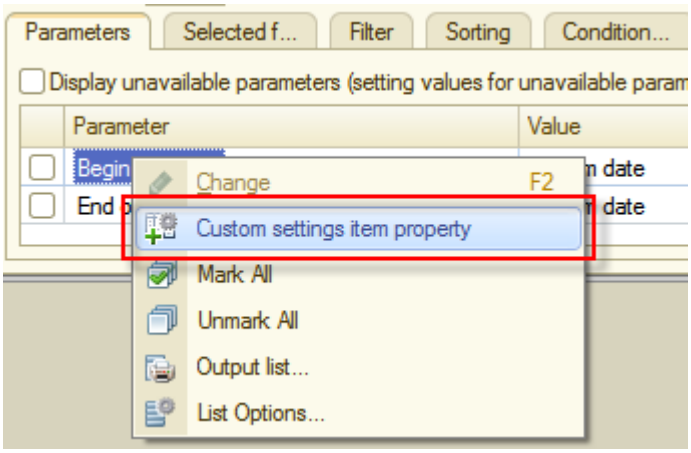

Figure 7-17. Opening Custom settings item property

In the opened window, select the **Include in custom settings** check box and click **OK**.

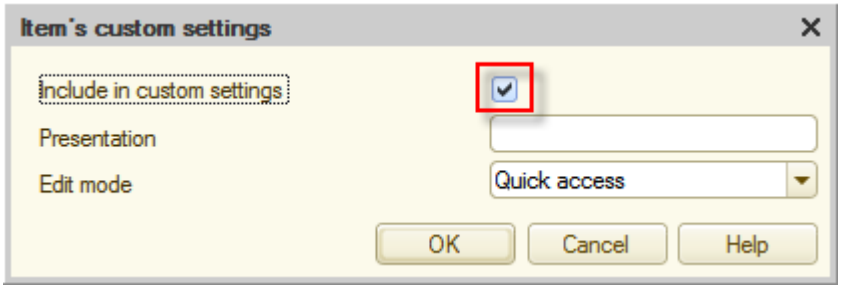

Figure 7-18. Include in custom settings

After repeating this procedure for both **Begin of period** and **End of period** parameters, continue configuring the report. To do this, close the data composition schema editor and return to **Expenses** report editor. Click the **Subsystems** tab and include this report in the **Finances** subsystem.

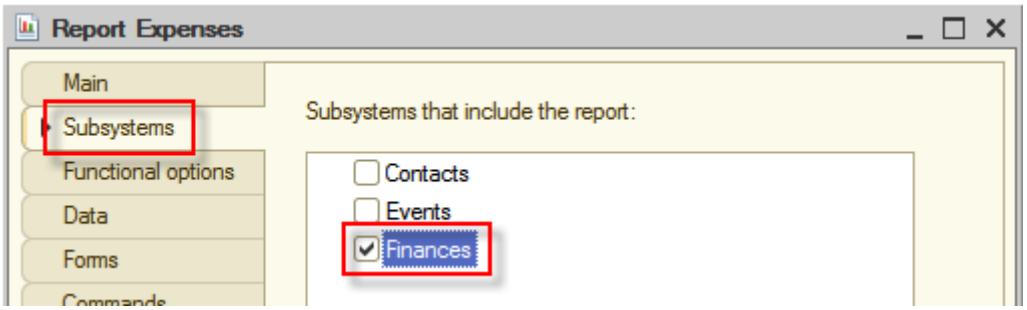

Figure 7-19 Including Expenses report in the Finances subsystem

Start the application in the 1C:Enterprise mode and take a look at the report. Notice that this report is located in the **Finances** section. Also, notice that the platform automatically placed the report in the **Reports** menu group.

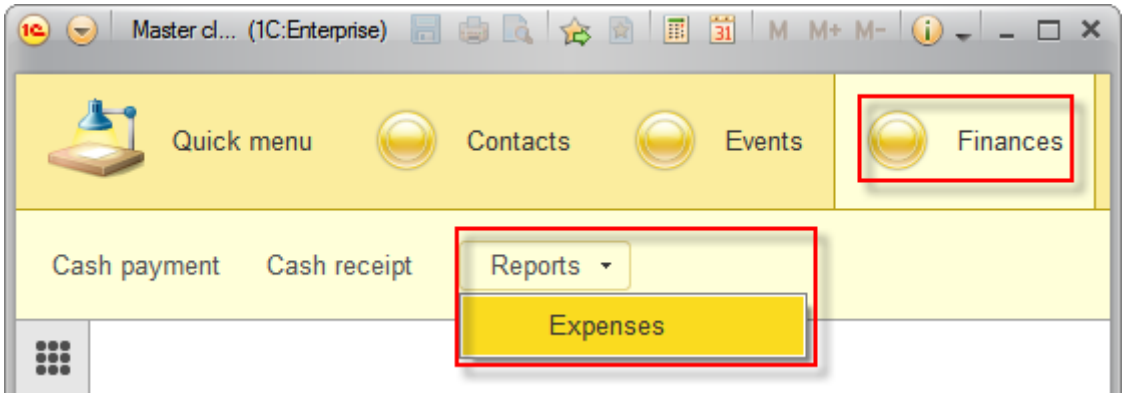

Figure 7-20. Opening Expenses report

After the report is opened, click **Create**.

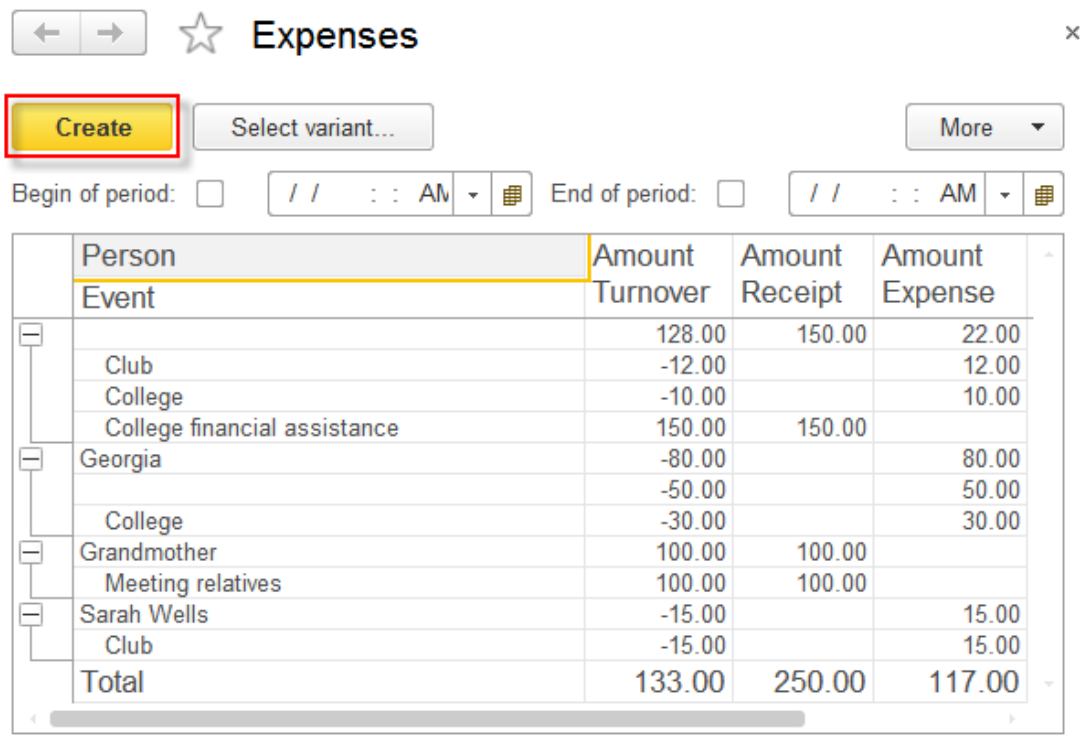

Figure 7-21. The Expenses report

Having confirmed that the report works properly and displays records of the **Cash receipt** and the **Cash expense** documents according to dimensions that were defined for the register, return to the Designer mode.

Create one more simple report that can show the current balance of available funds. Name it **CashBalance** and include it in the **Finances** subsystem. Similarly, to the **Expenses** report, click **Open data composition schema** for the **Cash balance** report on the **Main** tab and create the data composition schema. Add the data set of the **Query** type on the **Data sets** tab.

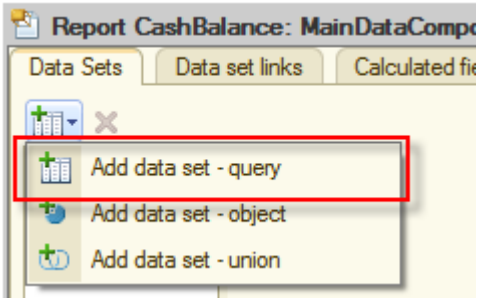

Figure 7-22. Adding the query for the Cash balance report

In the opened window, use **Query builder** as previously. However, for the new report, select the **FinancialTransactions.Balance** table.

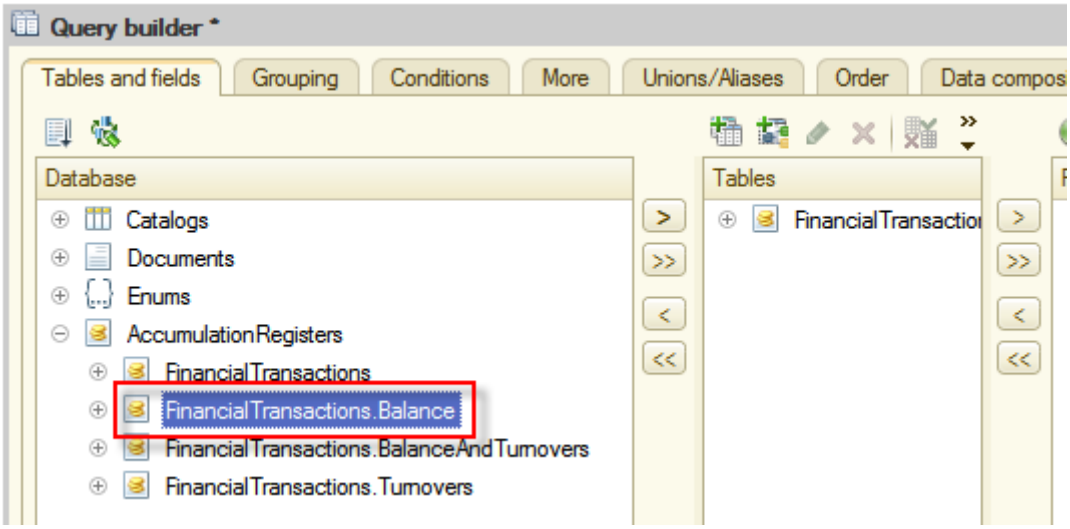

Figure 7-23. Selecting tables for the Cash balance report

Next, expand **FinancialTransactionsBalance** in the middle panel, click the **AmountBalance** field, and then double-click or click **Add selected** to select it.

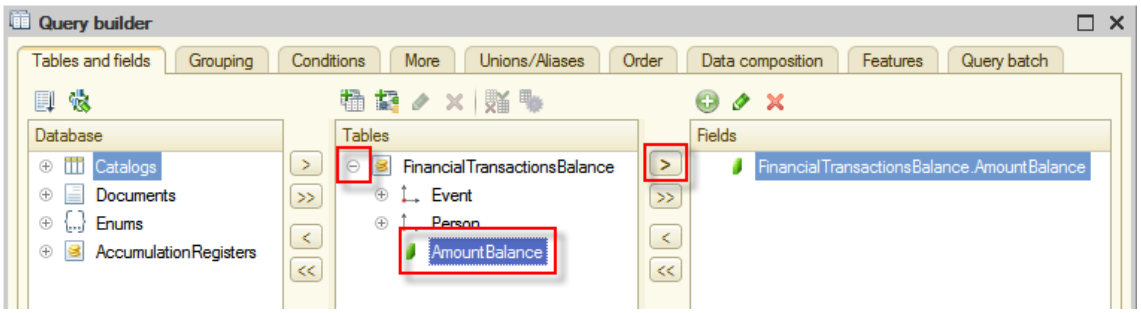

Figure 7-24. Selecting the AmountBalance field

Click **OK** to close **Query builder**. After this, again you will see the **Data Sets** tab of the data composition schema editor. No changes need to be made here. Click the **Resources** tab, then double-click the **AmountBalance** resource.

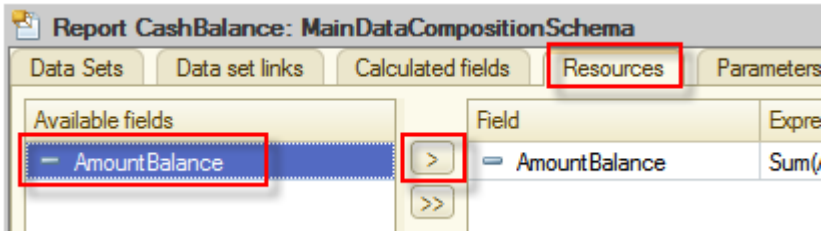

Figure 7-25. Selecting the AmountBalance resource

Open the **Settings** tab. Here, use the **Data Composition Settings Wizard** once again, confirming the default type, **List**.

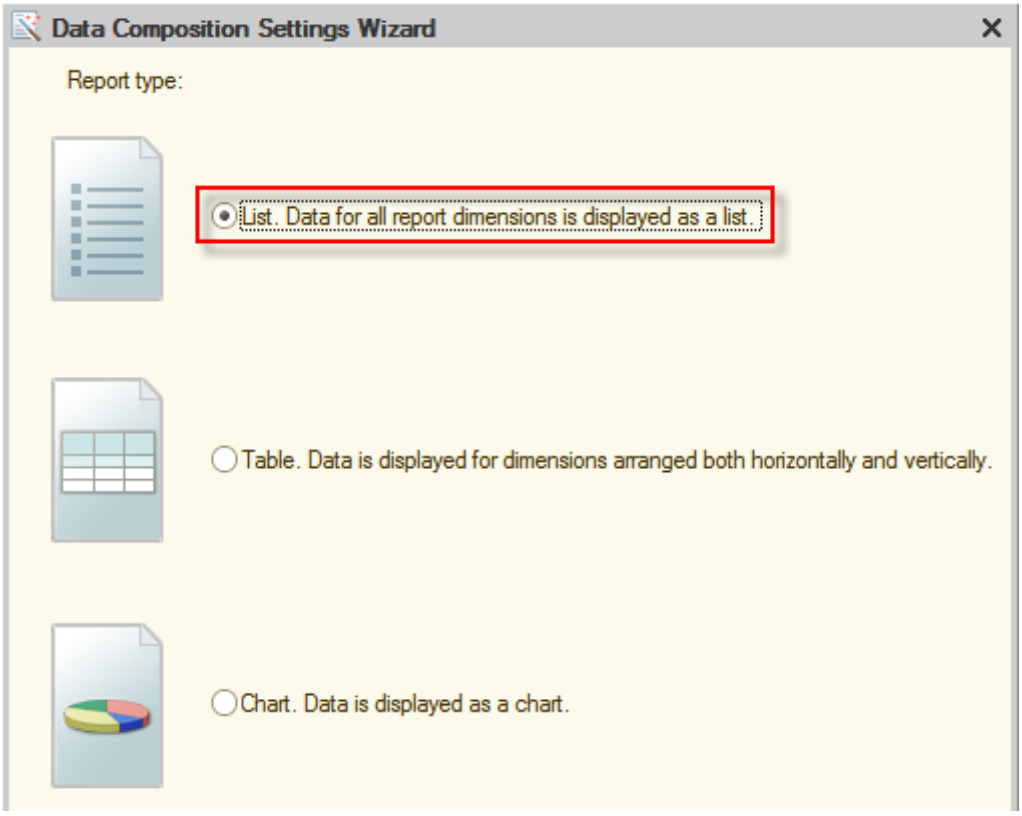

Figure 7-26. Data Composition Settings Wizard

On the next page, double-click the **Period** standard field in the **DataParameters** group. This field will allow seeing the date, which the report is created for. Then add the **AmountBalance** field.

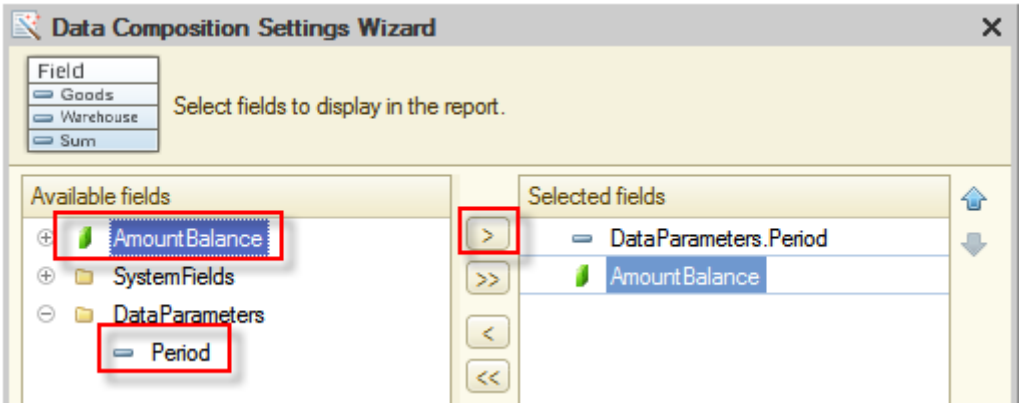

Figure 7-27. Selecting report fields

After selecting fields, click **OK** because there is nothing else that this report is required to display. The next thing to do is to include the **Period** parameter to **Custom settings** as you did when configuring the **Expenses** report.

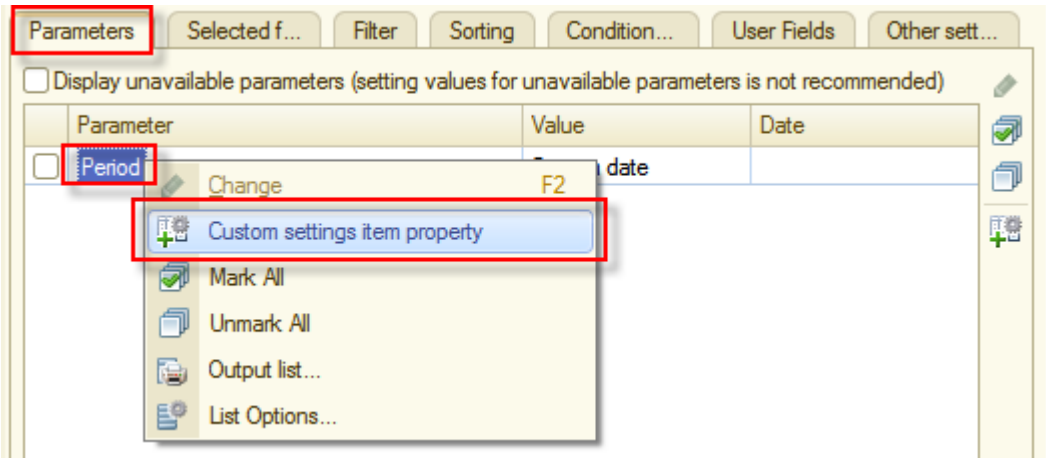

Figure 7-28. Including Period to Custom settings

Set the default value of the **Period** parameter to **Beginning of this day** so that the report will automatically display the information for the current day.

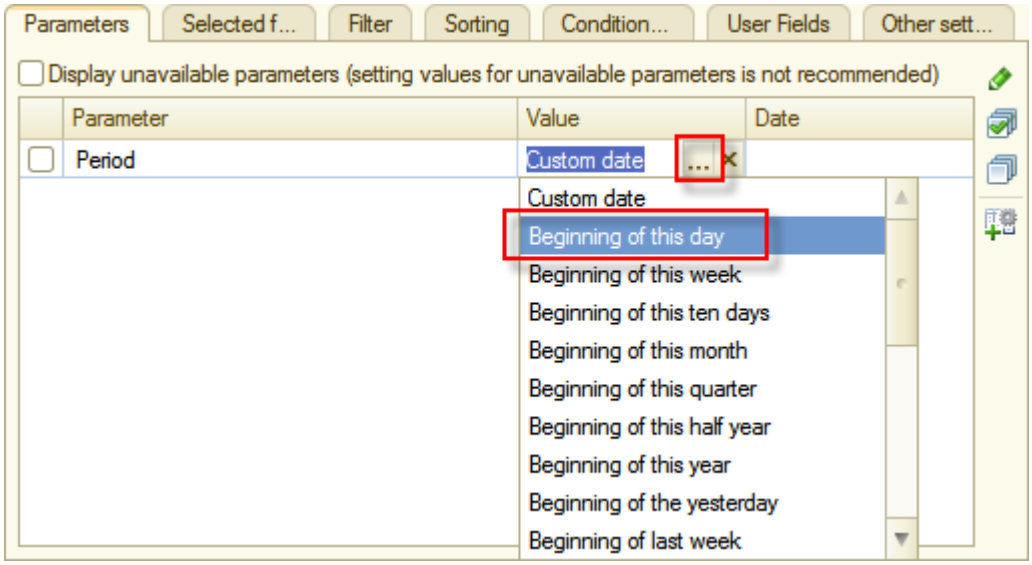

Figure 7-30. Selecting Beginning of this day as the default value of the Period parameter

After that changes, the report settings will look as follows:

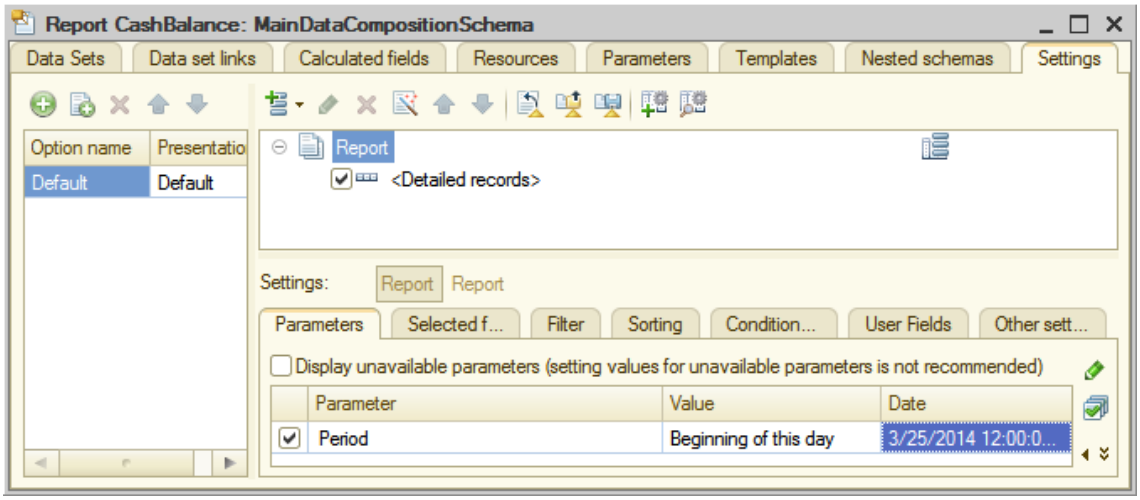

Figure 7-30. Settings of the Cash balance report

Now start the application in the 1C:Enterprise mode and check the report.

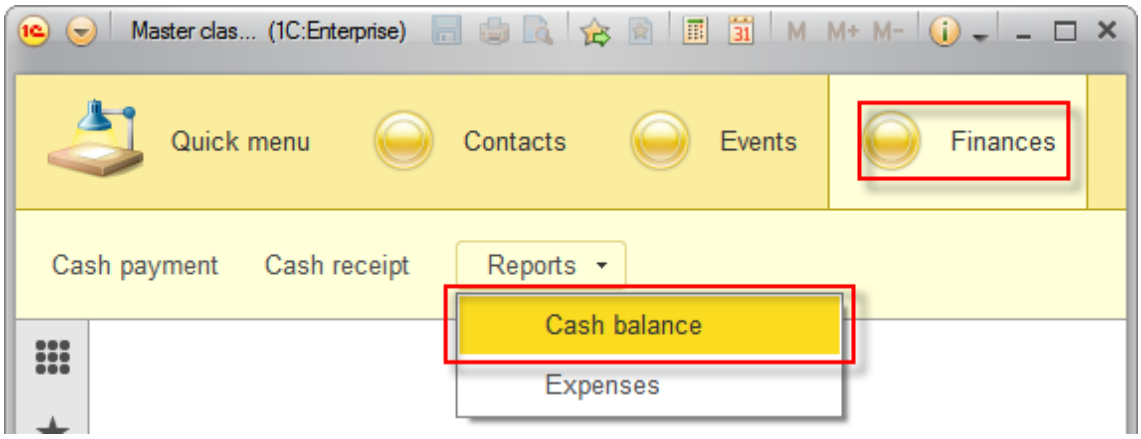

Figure 7-31. Opening the Cash balance report in the 1C:Enterprise mode

Information presented in the report is extremely simple but useful. Now the user can always check how much money is available now.

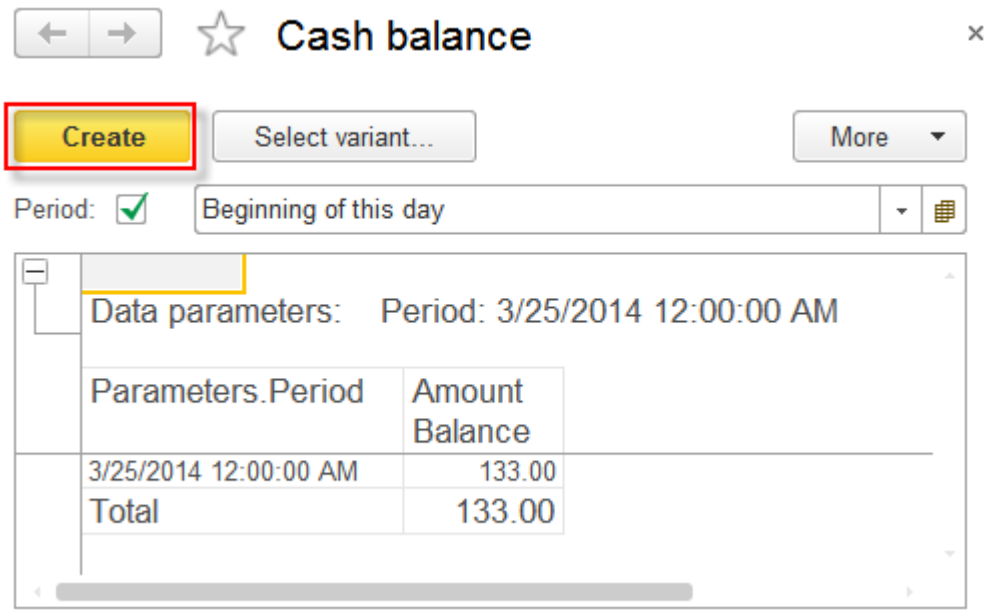

Figure 7-32. The data of the Cash balance report

Excellent! Now, create another report. After this report will be finished the CRM application will be complete.

Return to the Designer mode, create a new report, and name it **DailyChart**, then include it in the **Finances** subsystem, and finally click **Open data composition schema**. Add the **Data set** of the **Query** type, and then open **Query builder**.

In the left panel select the **FinancialTransactions.BalanceAndTurnovers** table. Up to this point, the process has no difference from what you did during configuring the **Expenses** report.

Seeing that the report is named **Daily chart** you might have guessed that a new report will display not numeric data but a chart. And now the trick. To present the data from the register in the suitable for this report format, select **FinancialTransactionsBalanceAndTurnovers** table in the middle panel of **Query builder** and click **Virtual table parameters** .

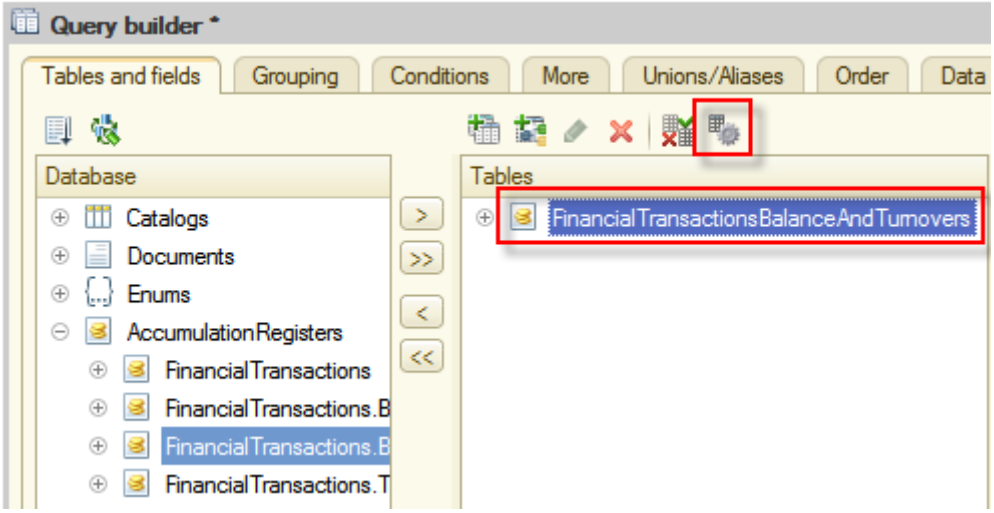

Figure 7-33. Opening Virtual table parameters

In the opened window, in the the **Periodicity** field, select **Day** and then click **OK** to save and close the window.

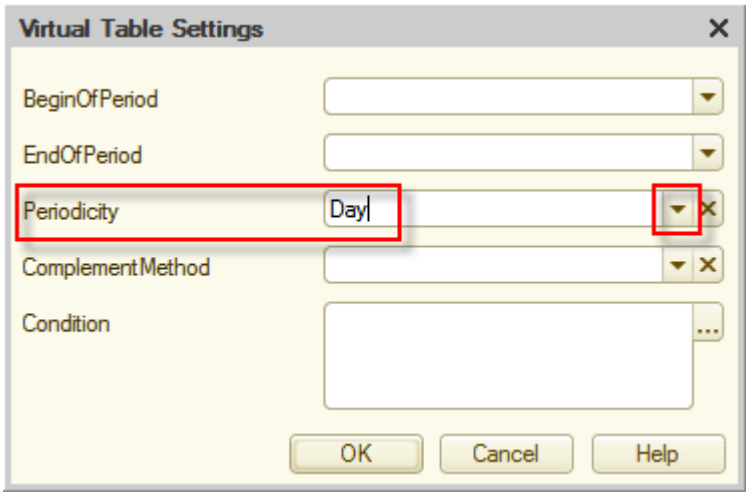

Figure 7-34. Adjusting Virtual table parameters

Then, in the middle panel, select the **Period** and the **AmountClosingBalance** fields from the **FinancialTransactionsBalanceAndTurnovers** table.

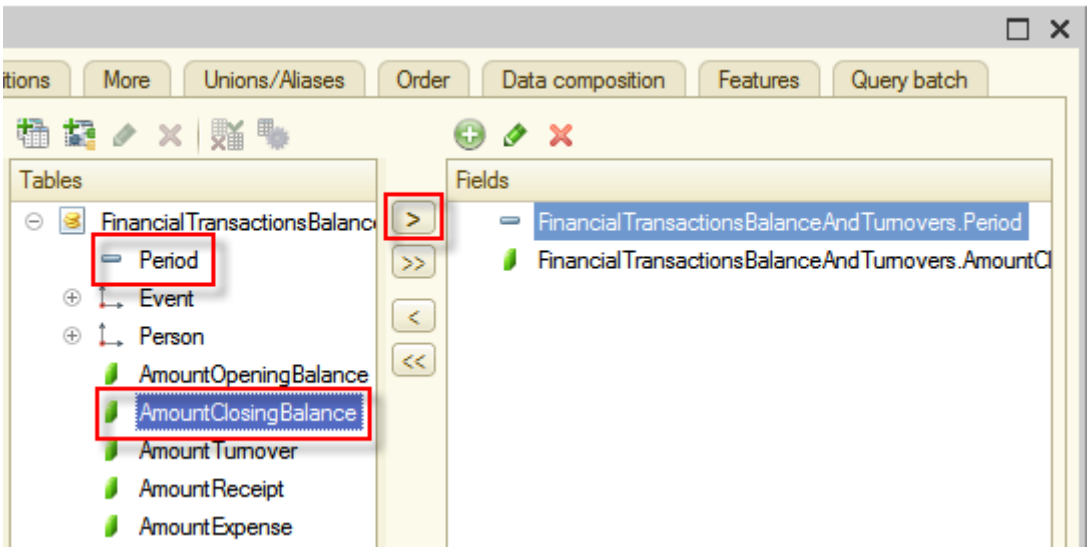

Figure 7-35. The Period and the AmountClosingBalance fields

Click **OK** to close **Query builder**. Then, click the **Resources** tab and select the **AmountClosingBalance** field as a resource.

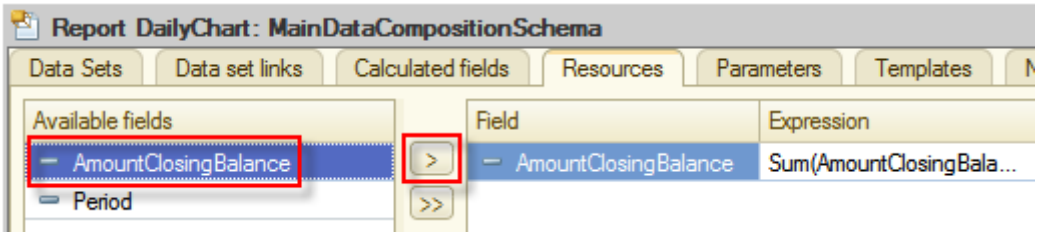

Figure 7-36. The Resources tab of the Daily chart report

Click the **Settings** tab and open **Data Composition Settings Wizard**. This time, click **Chart. Data is displayed as a chart**, and then click **Next**.

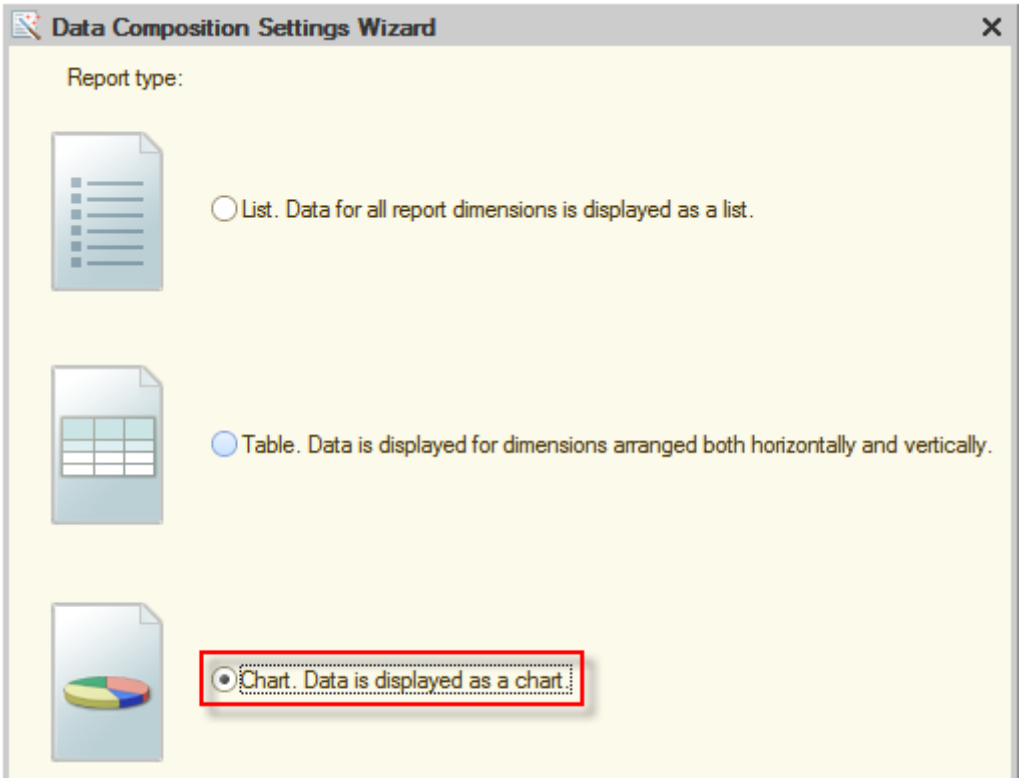

Figure 7-37. The Chart report type

After adding **Period** and **AmountClosingBalance** to selected fields, click **Next**.

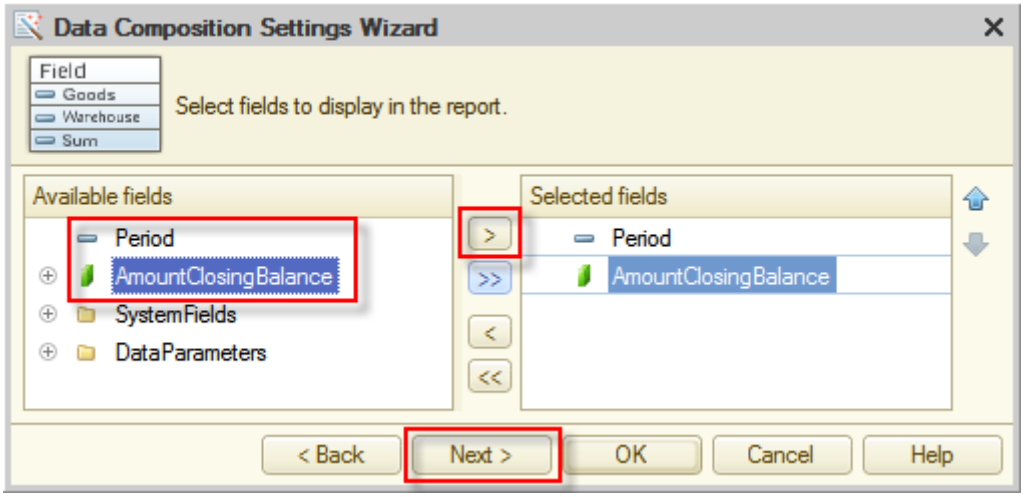

Figure 7-38. Adding chart fields

On the next page, add the **Period** field to the list in the **Points** panel on the right side. Click **Next**.

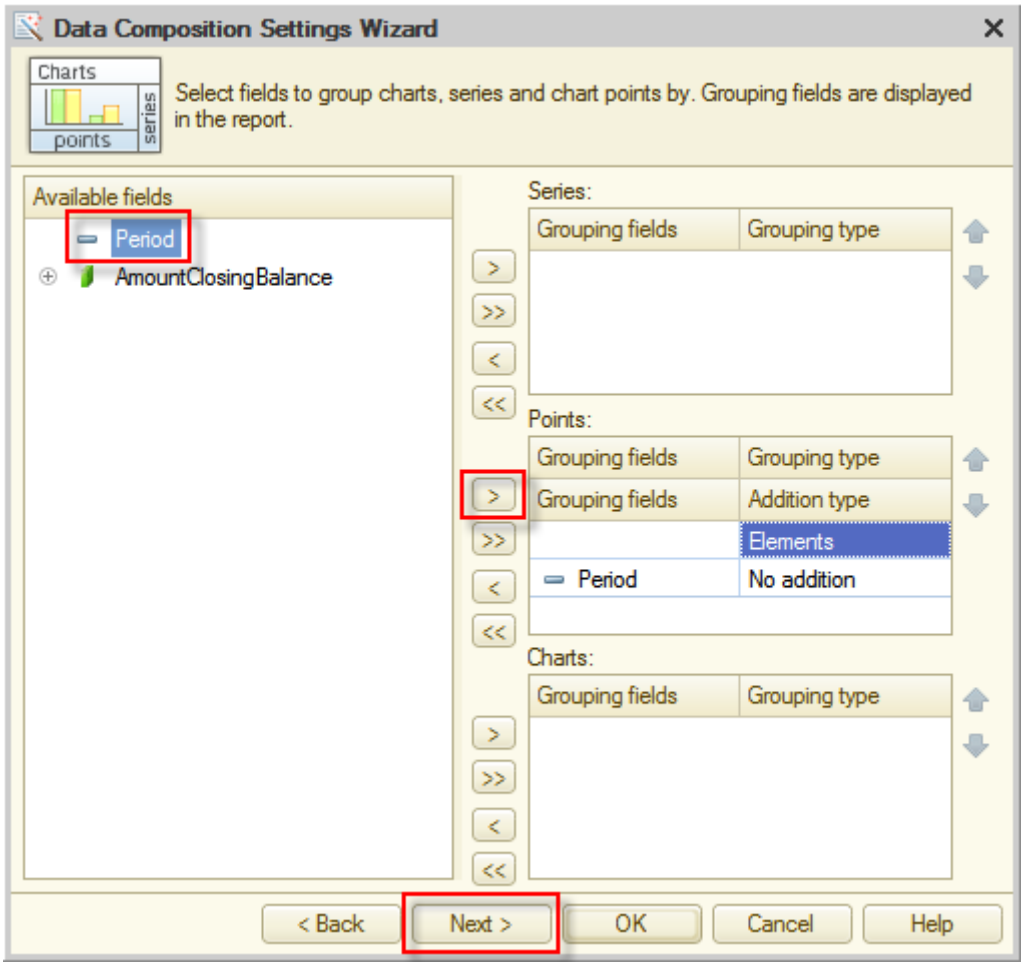

Figure 7-39. Configuring Points of the chart

On the next page, add **Period** to order fields on the right panel. Click **Next**.

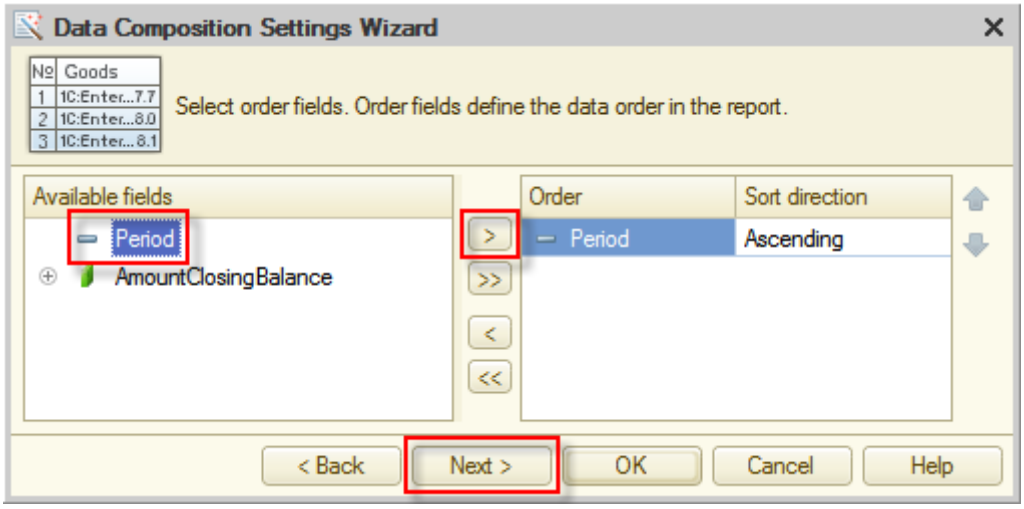

Figure 7-40. Data order fields of the chart

On the next page, in the **Chart type** field select **Line**. At this point, the design of chart complete, and you can click **OK**.

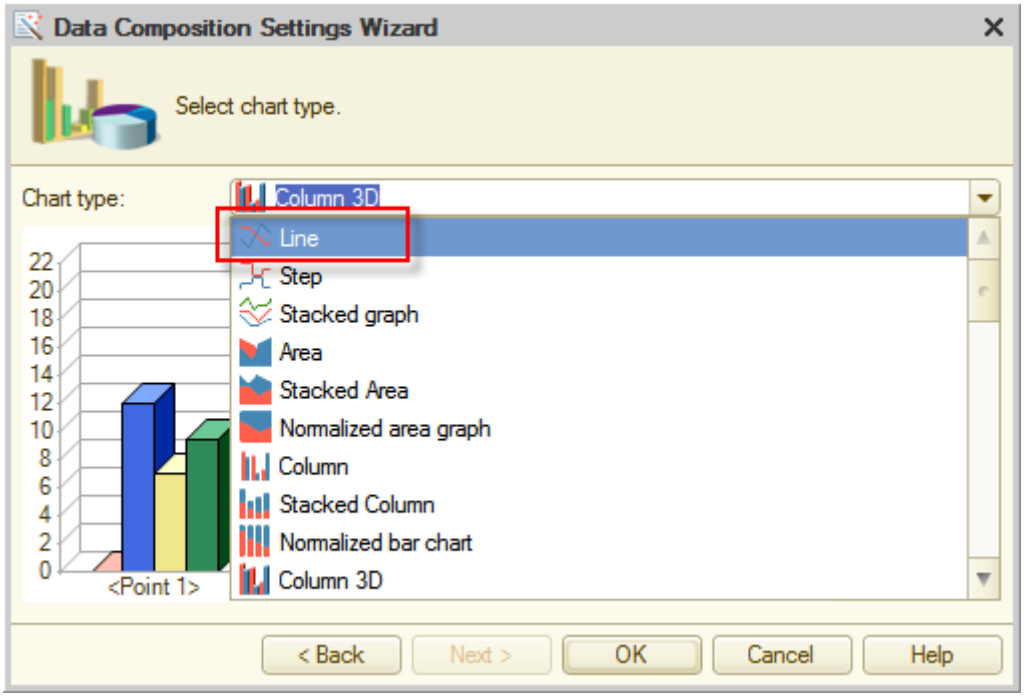

Figure 7-41. Chart types

In the same way as you did for previous reports, include the **Begin of period** and the **End of period** parameters in **Custom settings** using **Custom settings item property**.

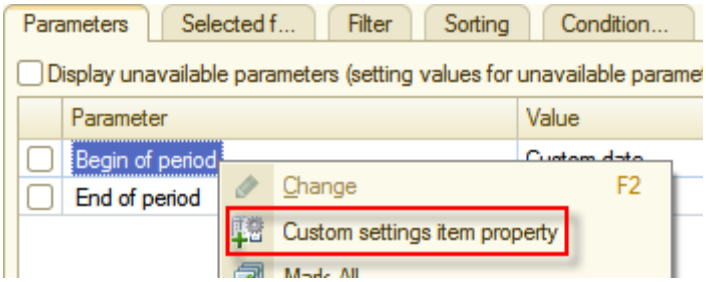

Figure 7-42. Adding properties to custom settings

Now you can see what you have created. Start the application in the 1C:Enterprise mode, open the **Daily chart** report, and then click **Create**.

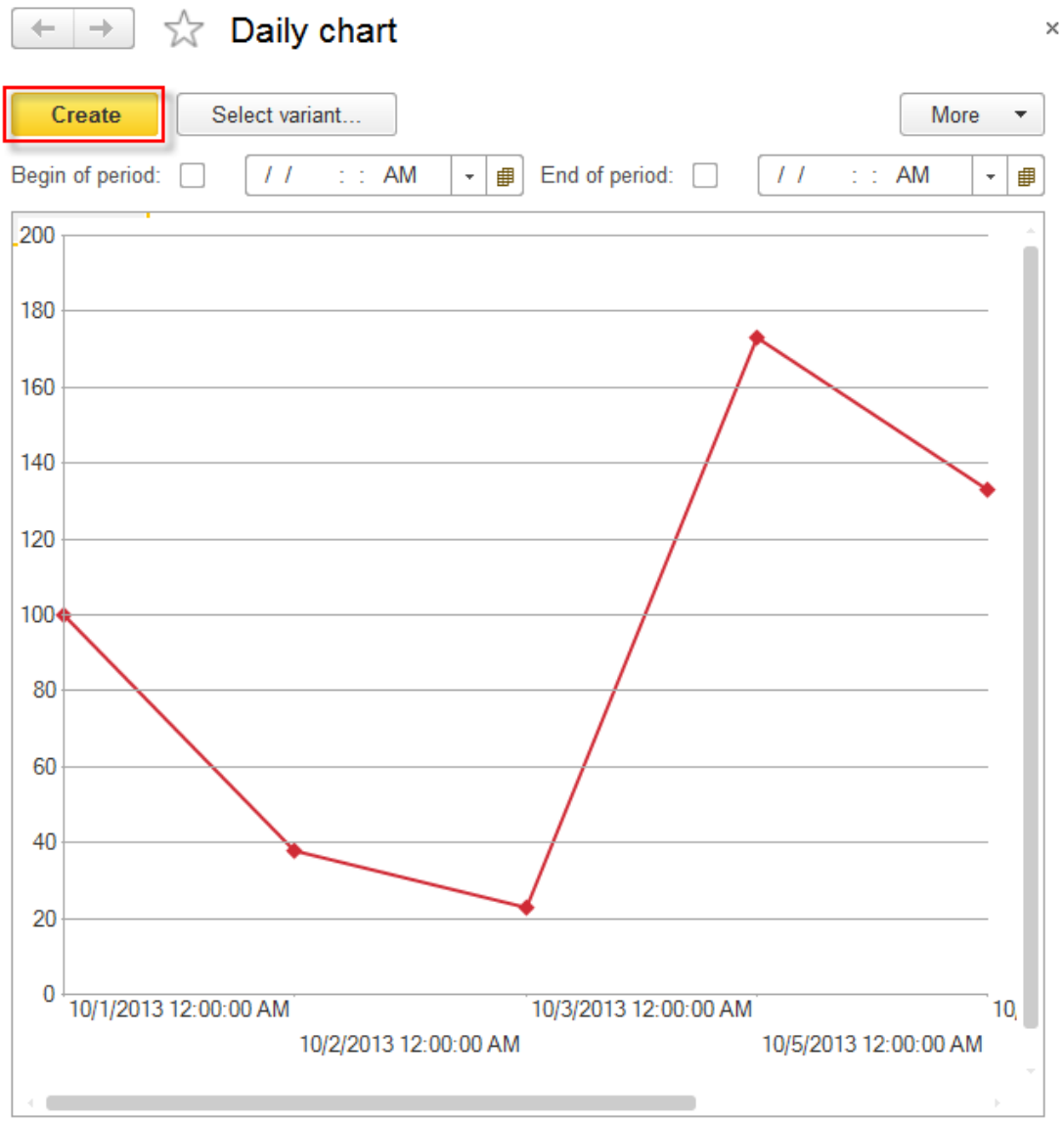

Figure 7-43. The Daily chart report

From this chart users can clearly see the changes in balance of money, made by **Cash receipts** and **Cash expenses**.

Improving the interface

So far, you have created a CRM application that is able to perform all actions, mentioned in project requirements. Except for one objective. At this point, the application is simple but not user-friendly.

Look at the main application window.

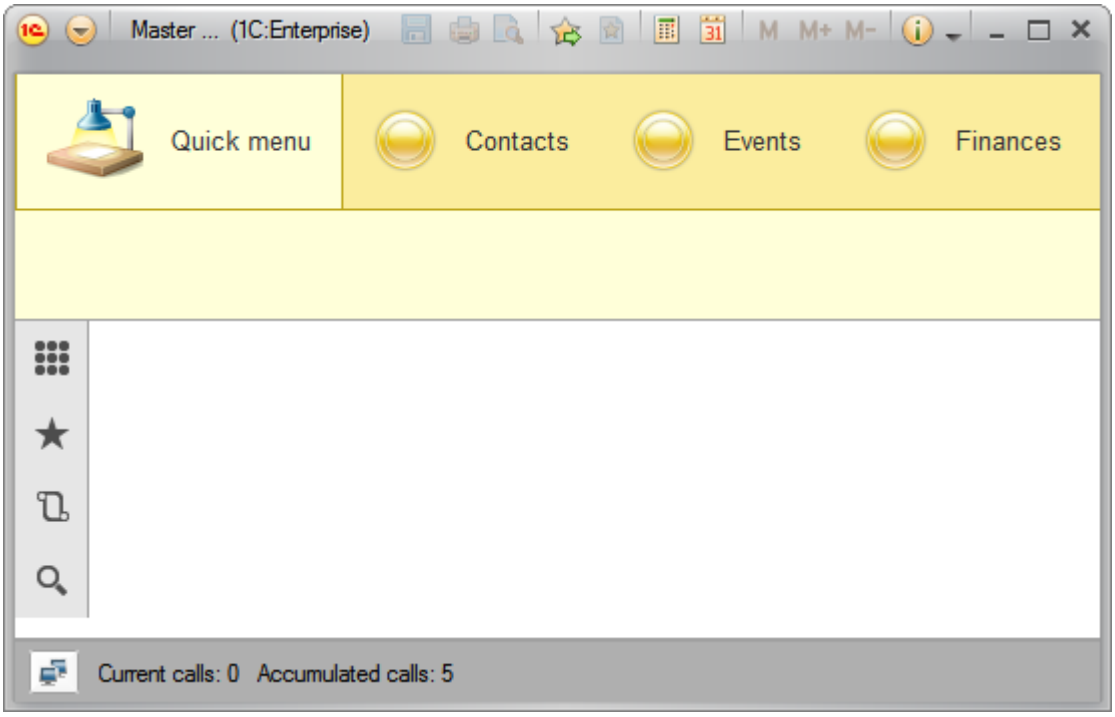

Figure 8-1. Start page

Start page is empty. All sections look the same and contain only the minimum that was automatically generated by the platform. In next chapters, you are going to improve this workspace.

Improving subsystems

As the first step, you will set icons for subsystems. This is not a difficult task. In the Designer mode open the subsystem editor and select the desired image in the **Picture** field for each subsystem. Start with the **Contacts** subsystem.

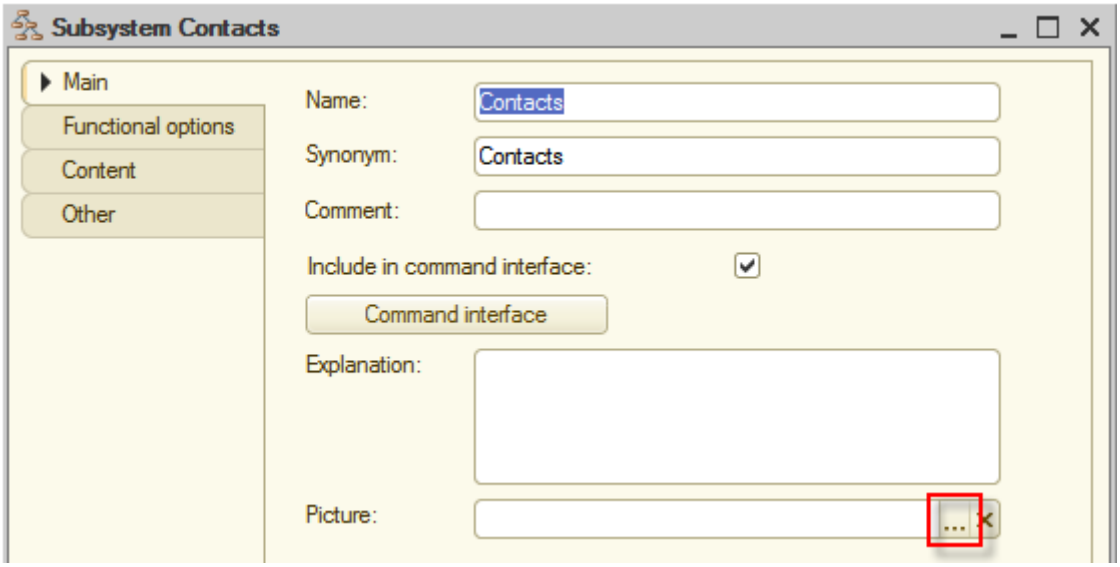

Figure 8-2. Selecting an icon for the subsystem

There are no pictures in the applied solution yet, so the list is blank. To have them available, import images from files. In the opened window, click **Add**. You can find images in the current distribution, for this, run **setup.exe**, click **Custom setup** and then click **Hello, 1C additional files**. In opened folder, you can find images. Copy them from that folder to your computer. And then add to the applied solution.

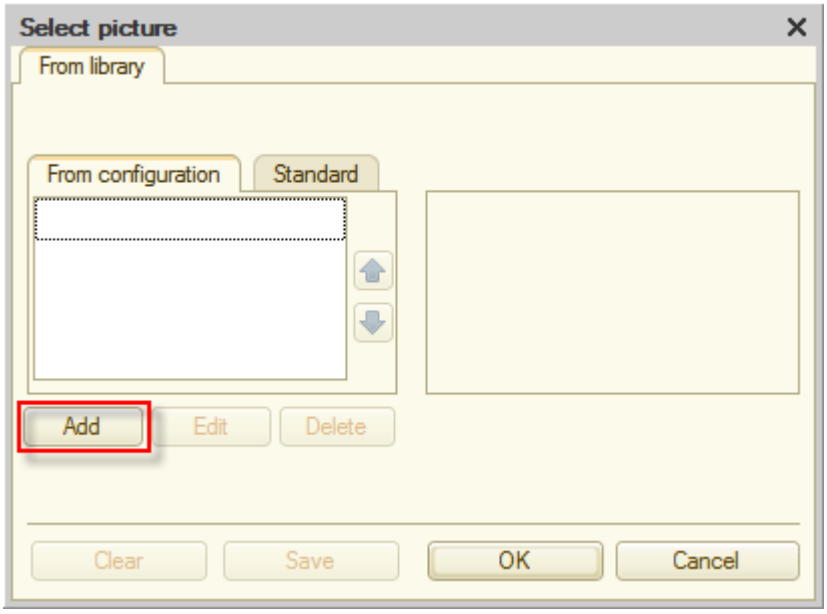

Figure 8-3. Adding an image

Click **Select from file**.

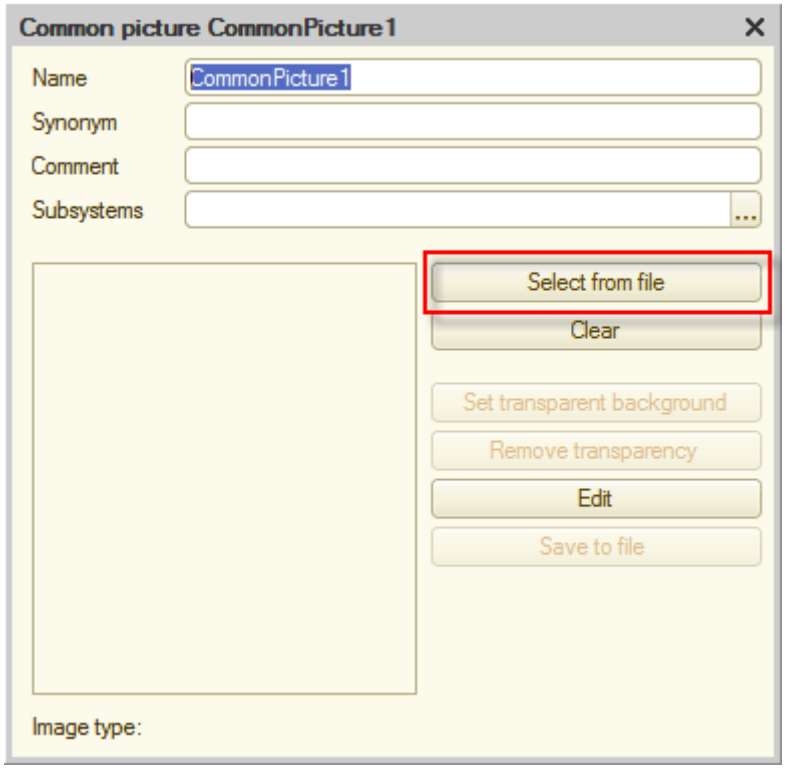

Figure 8-4. Selecting a file

For the **Contacts** subsystem select **ContactsSubsystem.png**, and then click **Open**.

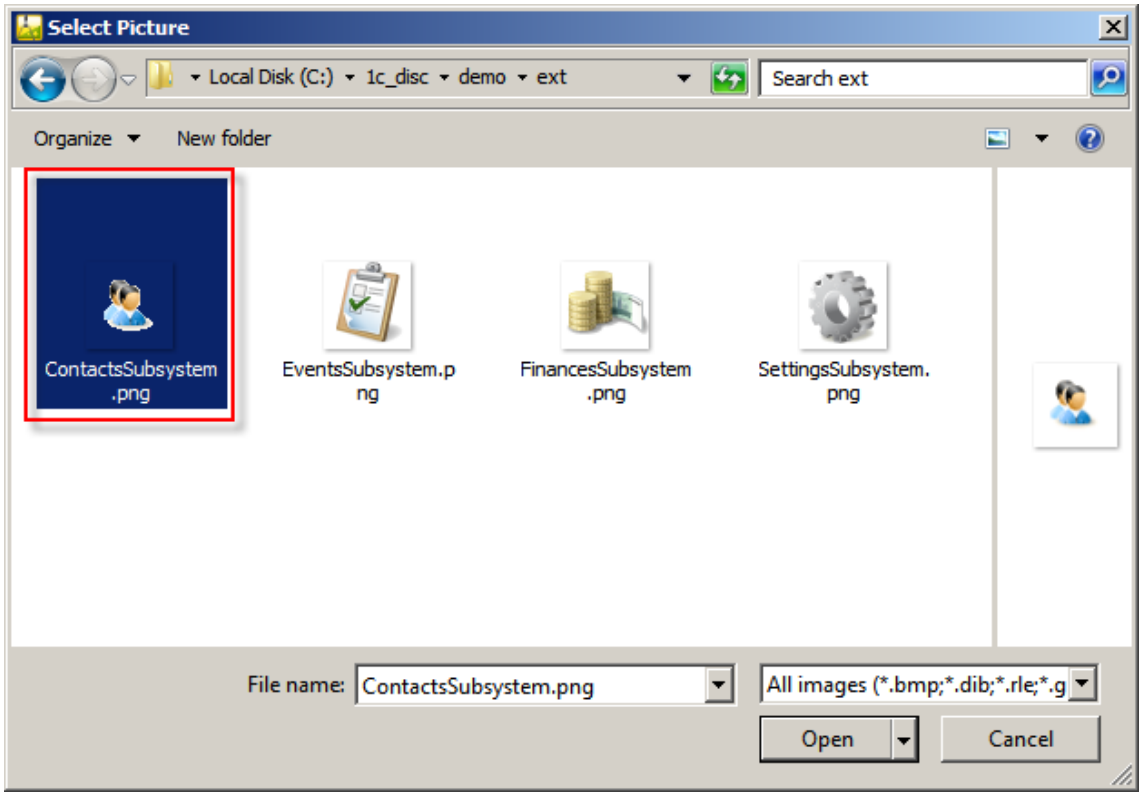

Figure 8-5. Selecting an image

Give the name to the picture, for example, **ContactsSubsystem**.

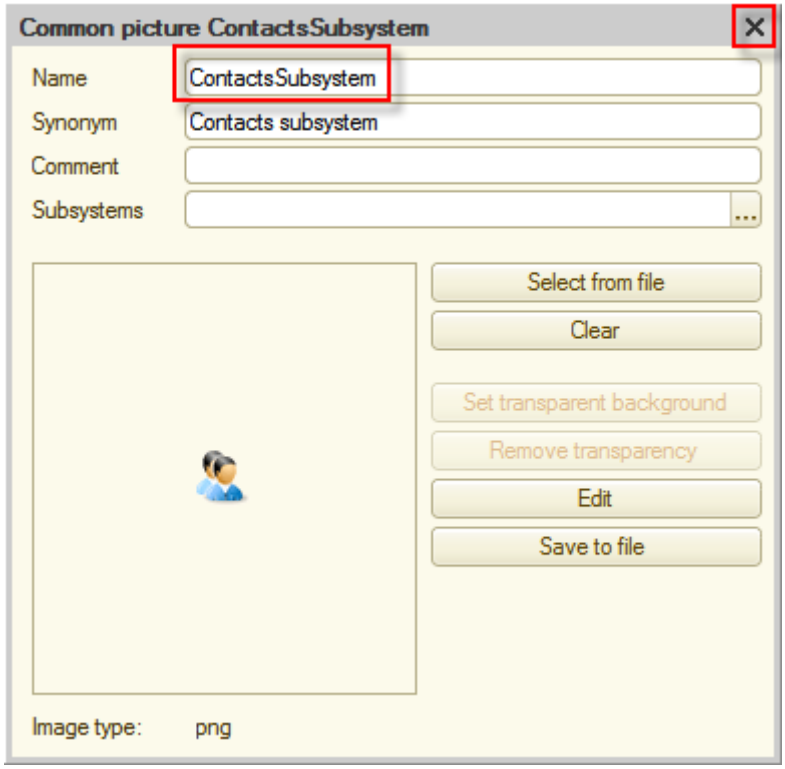

Figure 8-6. Naming the common picture

Now you can close the window with the picture. In the **Select picture** window select the picture that you have just imported, and then click **OK**.

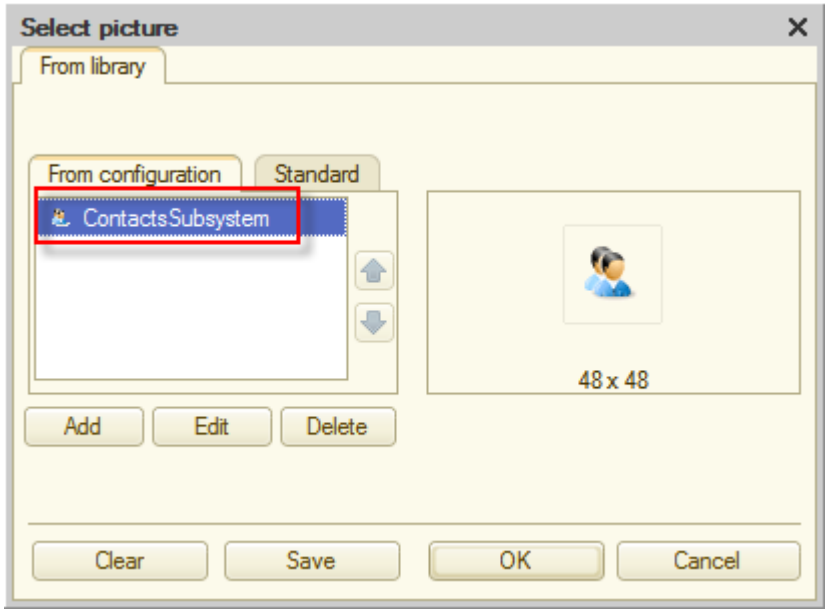

Figure 8-7. Selecting a picture

Now the **Contacts** subsystem has a unique appearance. Repeating the above steps, select images for the remaining subsystems:

- **EventsSubsystem.png** for the **Events** subsystem;
- **FinancesSubsystem.png** for the **Finances** subsystem.

As a result, pictures for subsystems will look as follows:

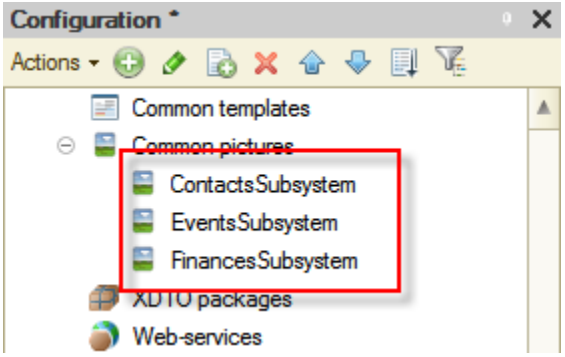

Figure 8-8. Common pictures for subsystems

Start the appliation in the 1C:Enterprise mode and look what you have now.

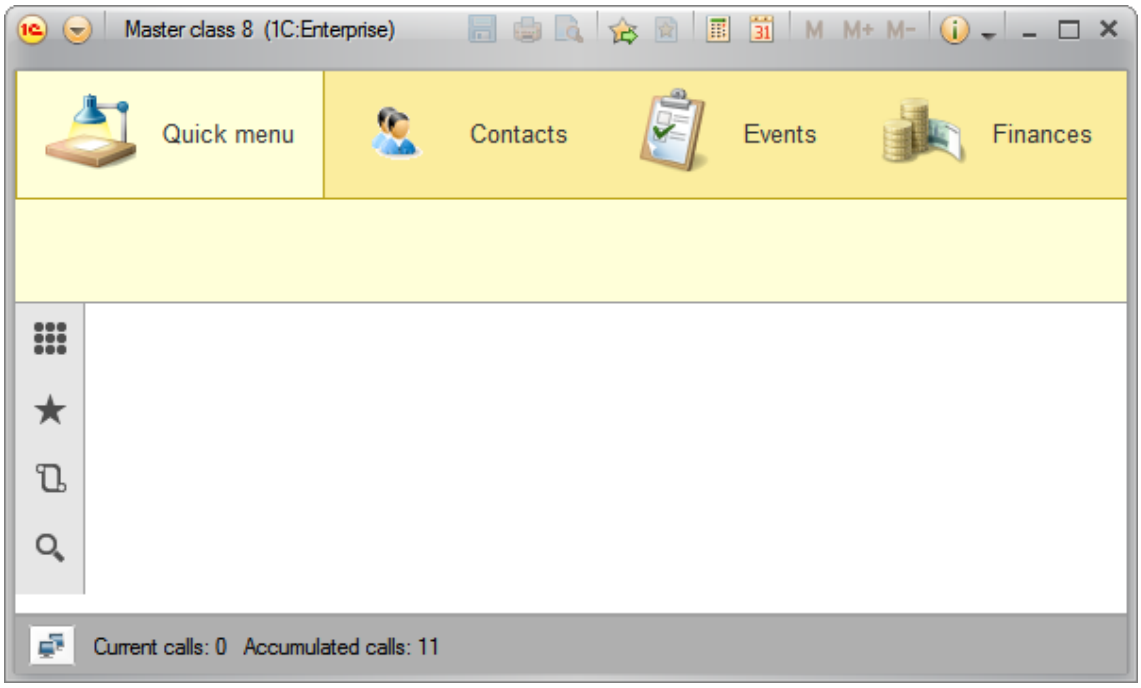

Figure 8-9. Subsystem icons in the 1C:Enterprise mode

Adjusting the subsystem content

Sections panel with newly added icons looks more user-friendly. Now, look at what is inside of sections. Their appearance is still quite blind. In **Contacts** there are only three links to catalogs, and in **Events** only two. The **Finances** section looks a bit more pleasant. Here also included reports besides documents.

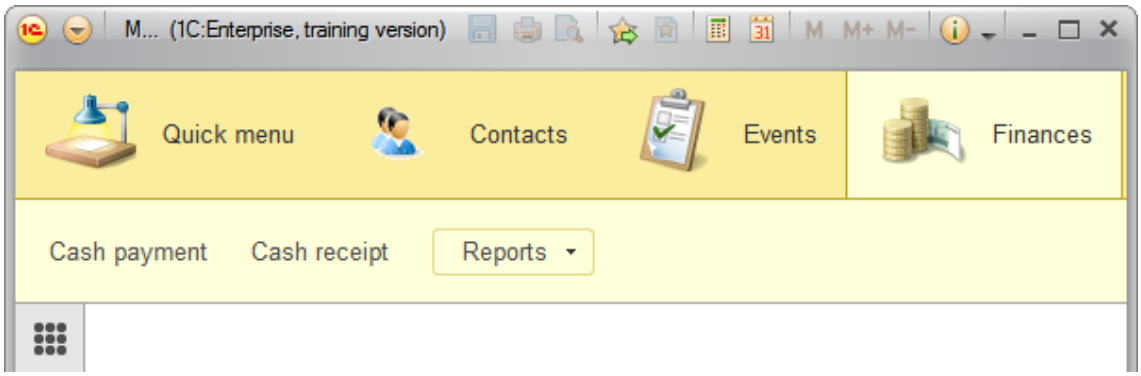

Figure 8-10. Content of sections

Add some diversity to the content of sections. Open the list of available commands for subsystems, and activate those commands that may be useful to users. Start with the **Contacts** subsystem. Open the subsystem editor, and click **Command interface**.

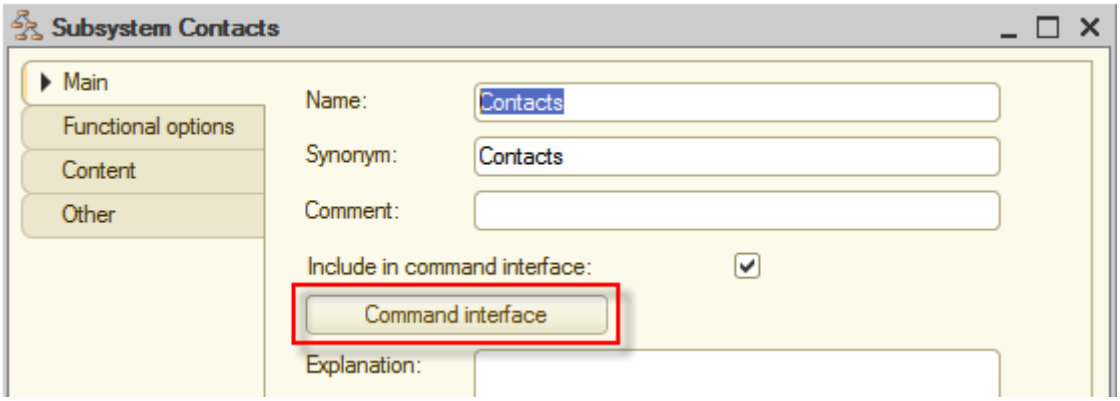

Figure 8-11. Opening Command interface

Place the often used **People** catalog in the **Navigation panel.Important** command interface group either by dragging it there with a mouse or clicking **Move a command** . Add the ability to create new records for all three catalogs in the **Actions panel.Create** command interface group by selecting **Visibility** check boxes. Then change the order of the **People: create** command by clicking **Move up** .

As the result, you will have following command interface settings:

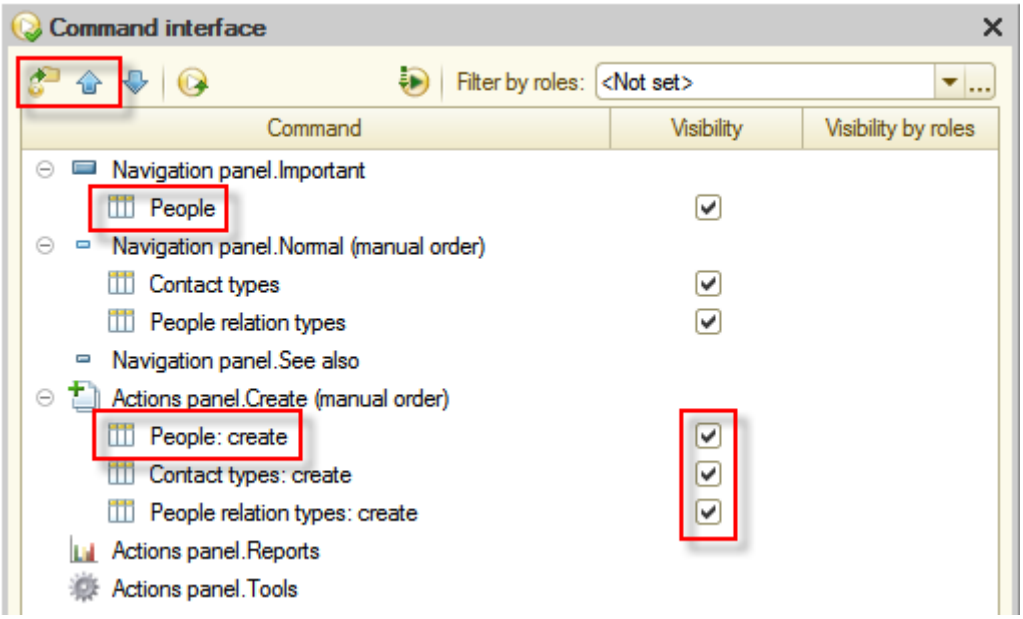

Figure 8-12. Adjusting Command interface of the Contacts subsystem

Make the similar changes to the **Events** subsystem.

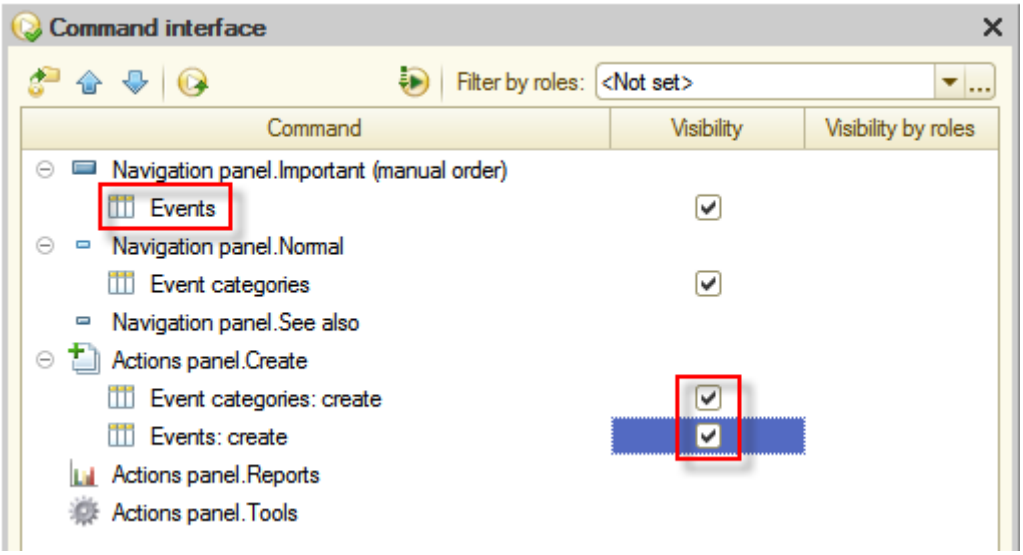

Figure 8-13. Adjusting Command interface of the Events subsystem

Also, check **Finances** subsystem. Here set **Financial transactions** command as an **Important** command.

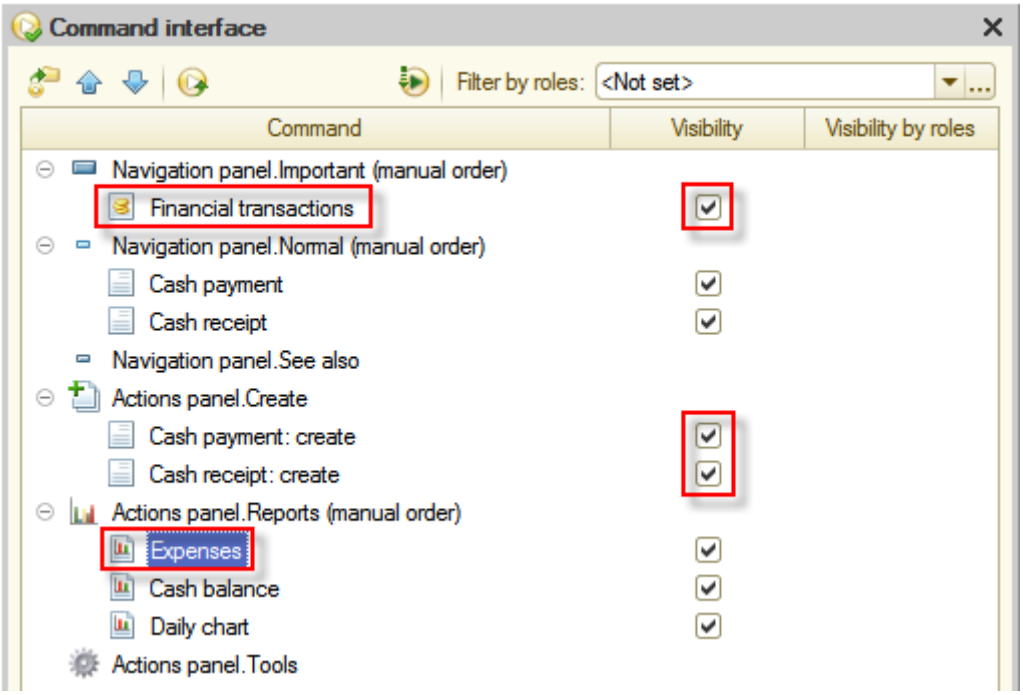

Figure 8-14. Adjusting Command interface of the Finances subsystem

Start the application in the 1C:Enterprise mode and take a look at the interface now. Well, links in the **Contacts** section look more orderly. The **People** catalog can be easily found now, which is convenient, considering that users will work with this catalog more often than with others. In addition, there is a new command group for creating new catalog items in this section.

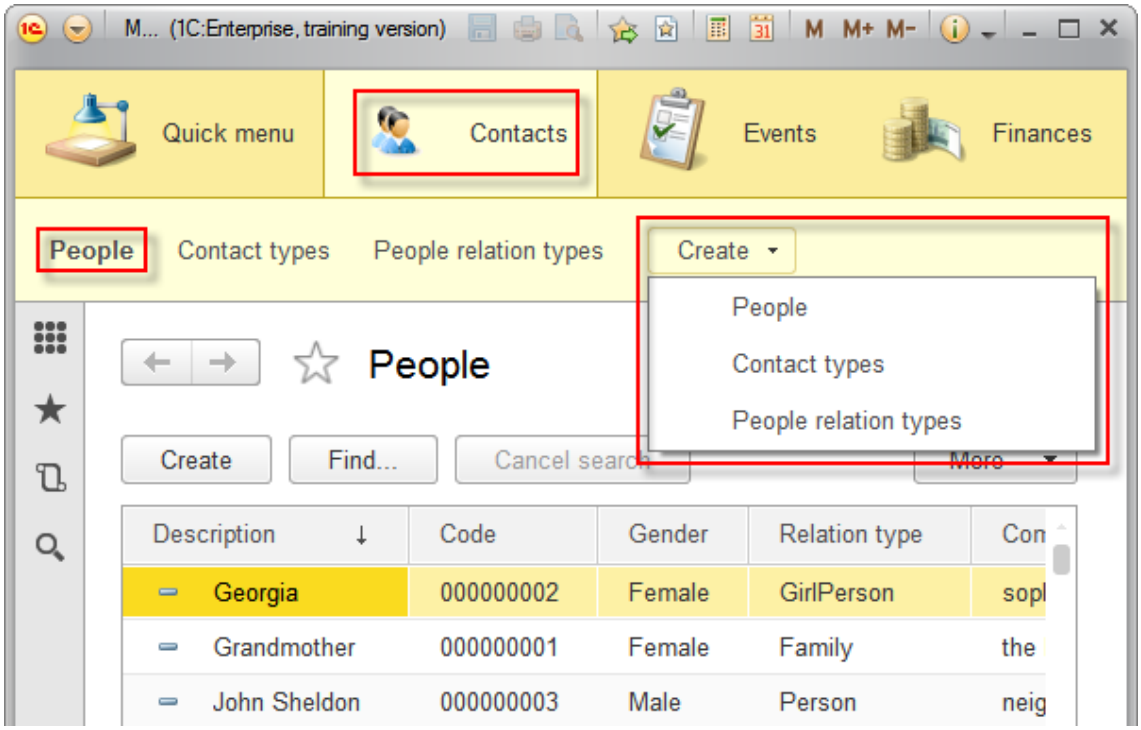

Figure 8-15. The Contacts section

Look at the interface of the **Events** section, which now looks more complete.

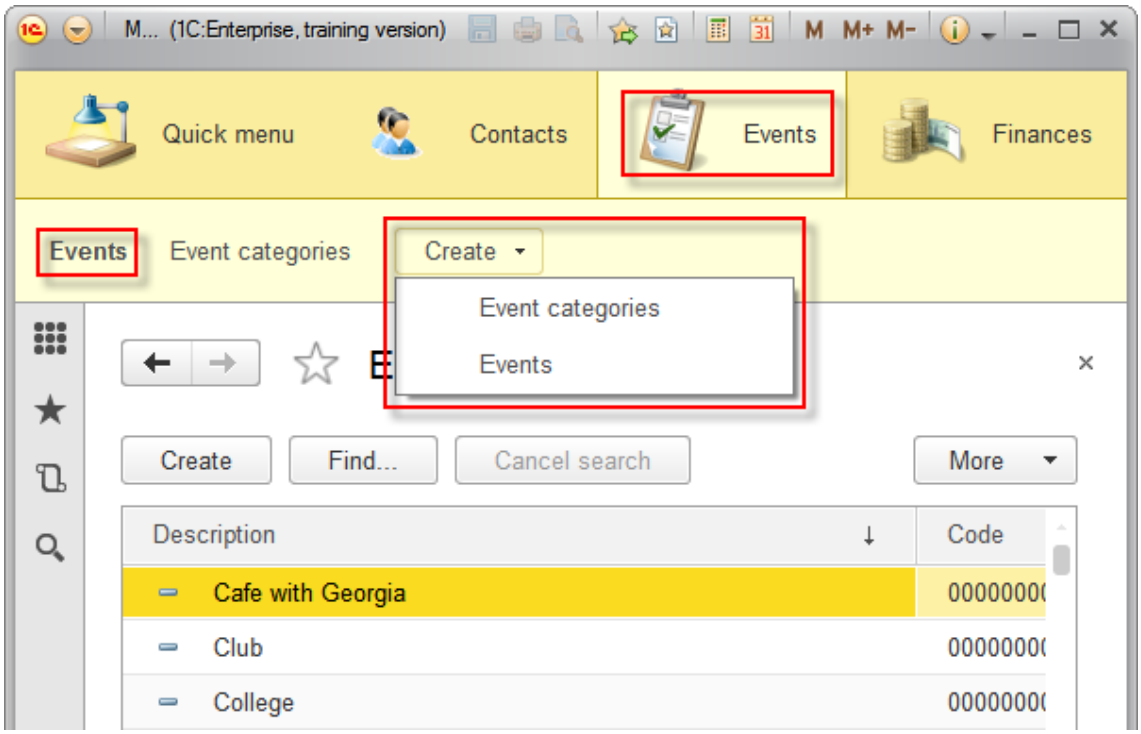

Figure 8-16. The Events section

Now, look at the **Finances** section. This time all objects that the user might need are easily accessible. For example, you no longer need to open **All functions** in **Main menu** each time when you want to view records of the **Financial transactions** register.

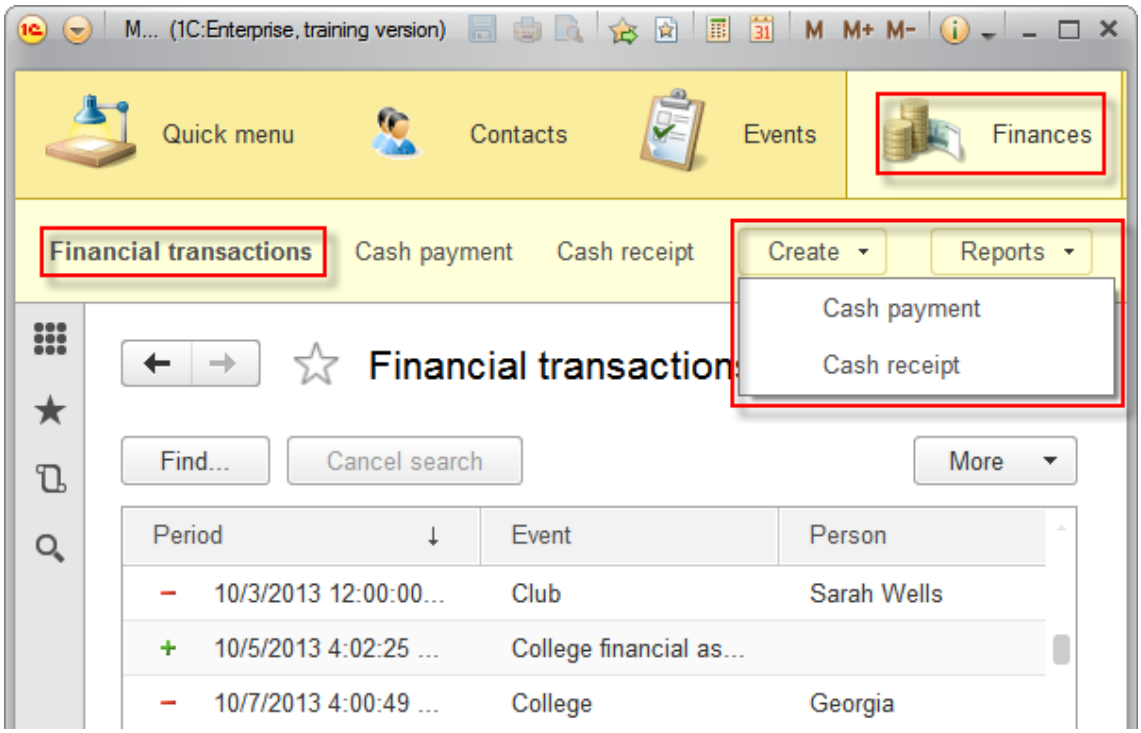

Figure 8-17. The Finances section

Start page

You have managed the lack of navigation links in subsystems. But **Start page**, which is the first page that the user sees when open an application in the 1C:Enterprise mode, is still empty.

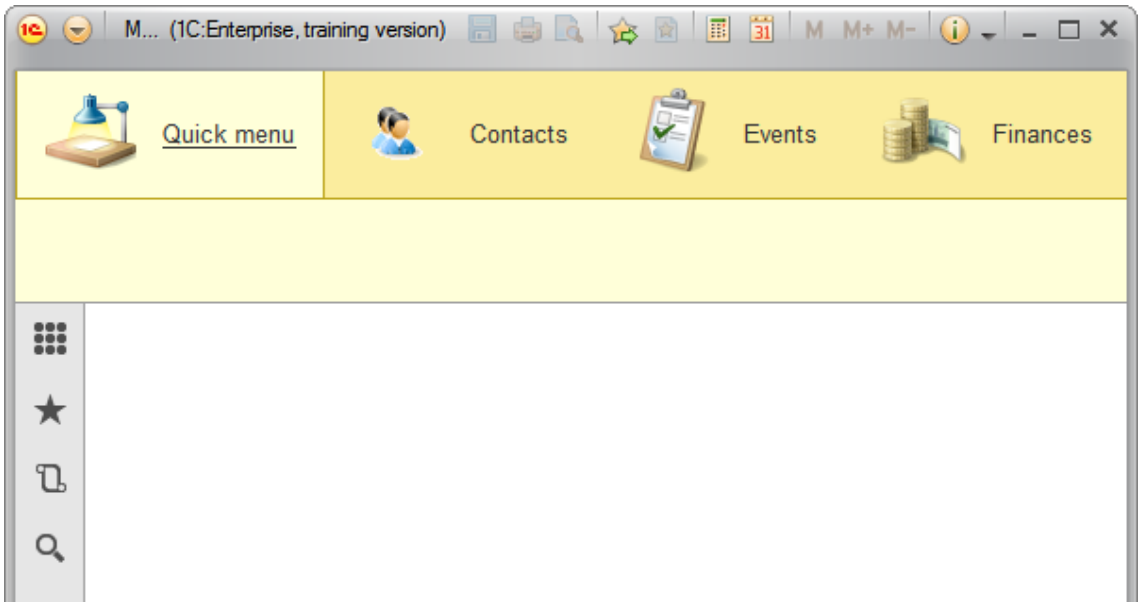

Figure 8-18. The empty Start page

You definitely want to make things right. To do this, return to the Designer mode, open **Properties** of the **Configuration** root node, and click on **Open** link of **Start page working area** field.

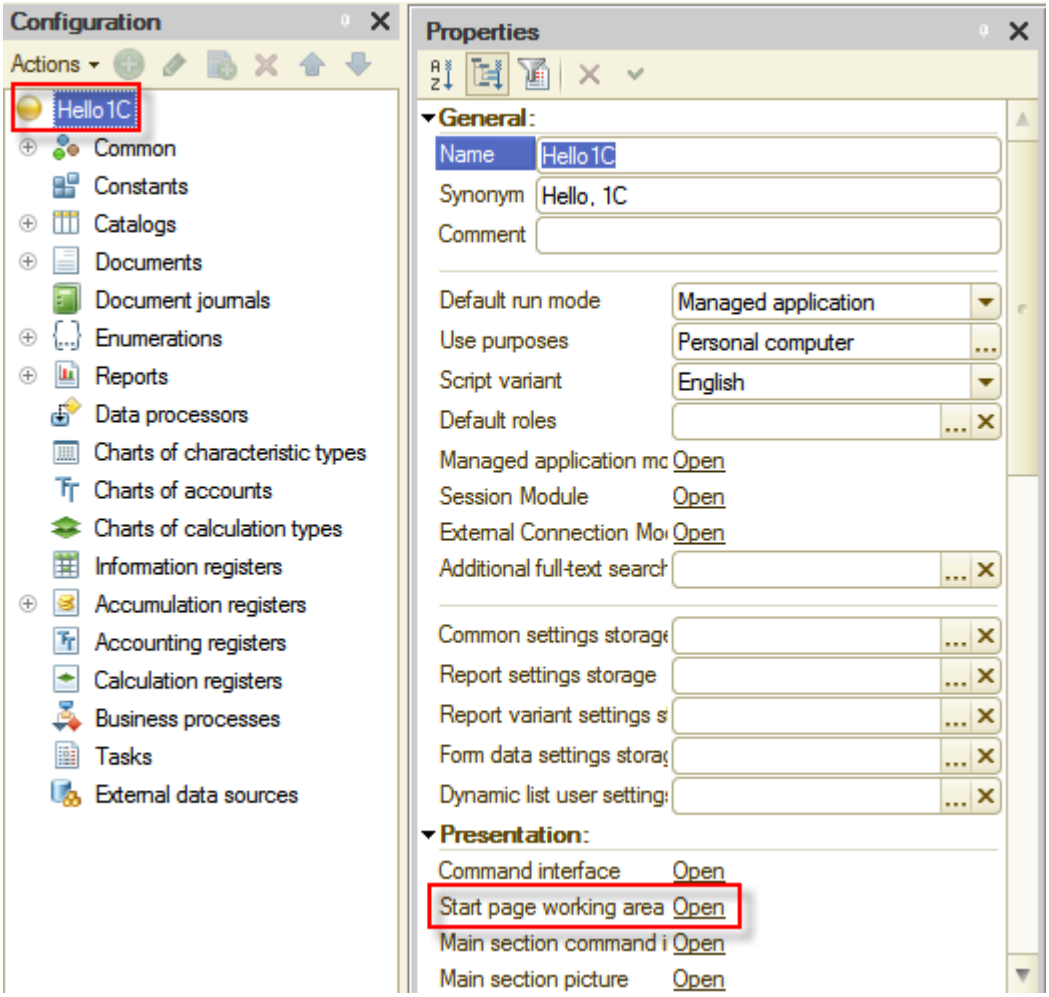

Figure 8-19. Opening Start page working area

There are a few things that you can change here for now, which is why simply select **Two columns, different width (2:1)** as the value of the **Starting page template** field.

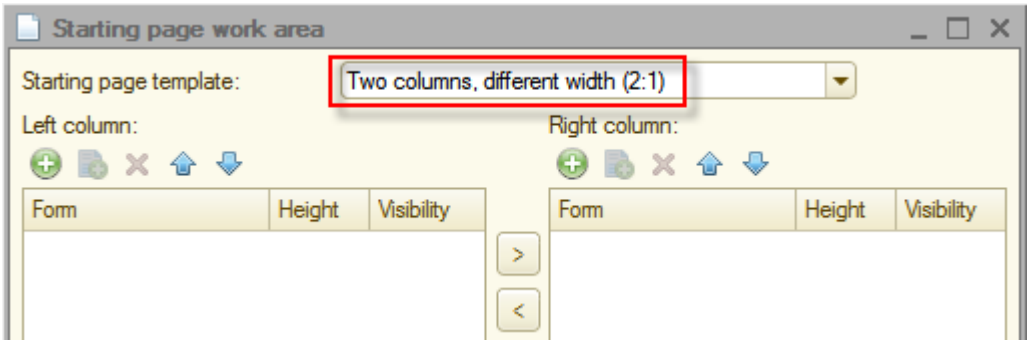

Figure 8-20. Starting page template

The feature of **Start page** is that you can place there only those forms that are explicitly created in the **Configuration** object tree. Thus, the first thing you need to do is to create forms using **Form Wizard**. Those forms are the list form of the **Financial transactions** accumulation register, the list form of the **People** catalog, and the report form of the **Cash balance** report.

It is easy to do. In the **Configuration** object tree, select the configuration object, for which you are going to add a form. Expand this branch and right-click **Forms** node inside of it, then click **Add** (Ins).

Start with the register.

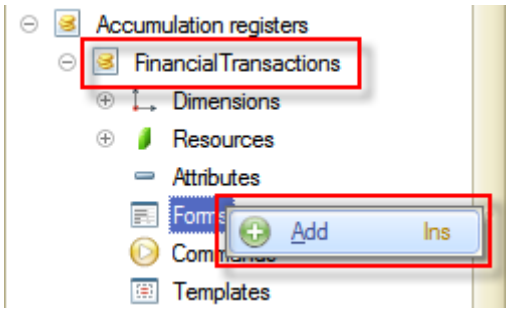

Figure 8-21. Adding the list form for the Financial transactions accumulation register

The **Accumulation Register Form Wizard** window will open. In this case, the default option **Accumulation register list form** is the required one to create the desired list form, so simply click **Finish**.

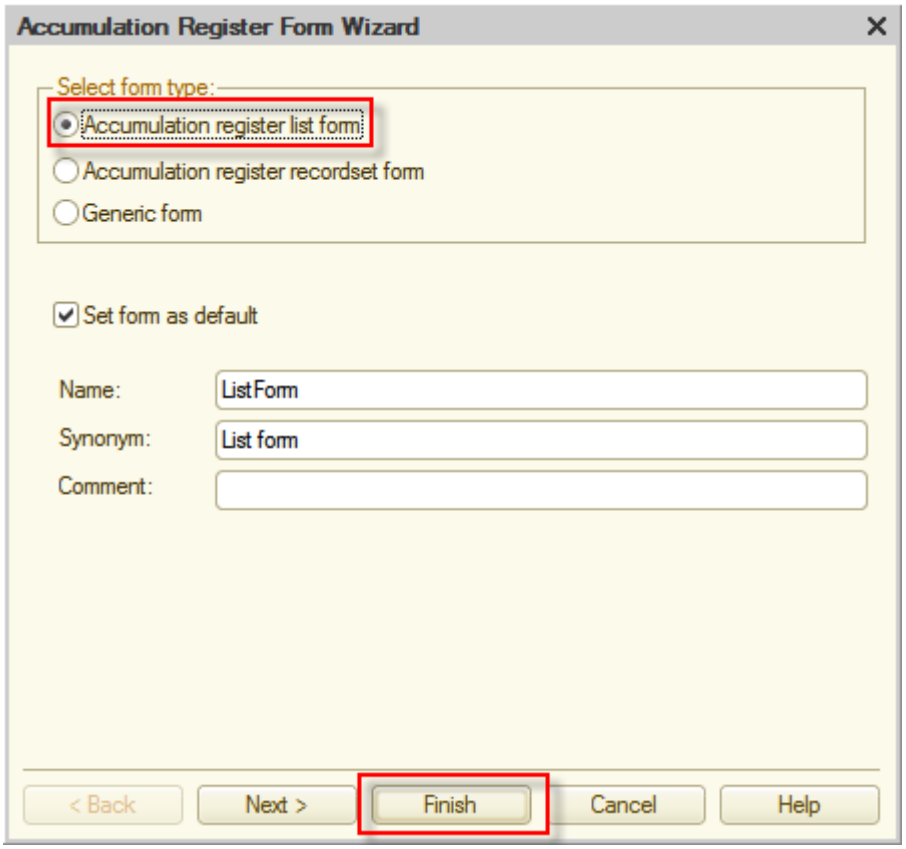

Figure 8-22. Adjusting the list form of the accumulation register

Find the **People** catalog in the **Configuration** object tree, and add a list form for this catalog as well.

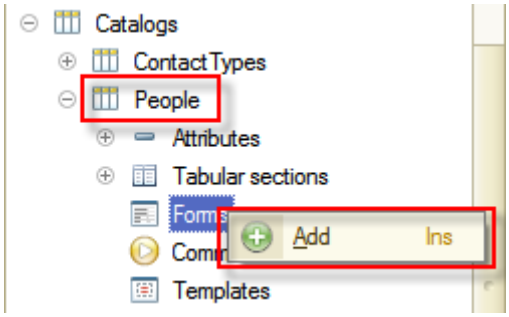

Figure 8-23. Adding a list form for the People catalog

Click **Catalog list form** in the **Select form type** panel, and click **Finish**.

Figure 8-24. Adjusting a list form of a catalog

Now create the report form for the **Cash balance** report.

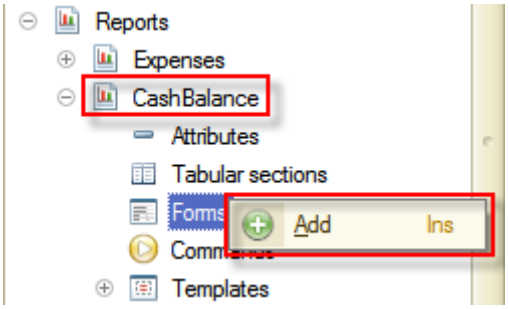

Figure 8-25. Adding a report form for the Cash balance report

Click **Report form** in the **Select form type** panel and click **Finish**.

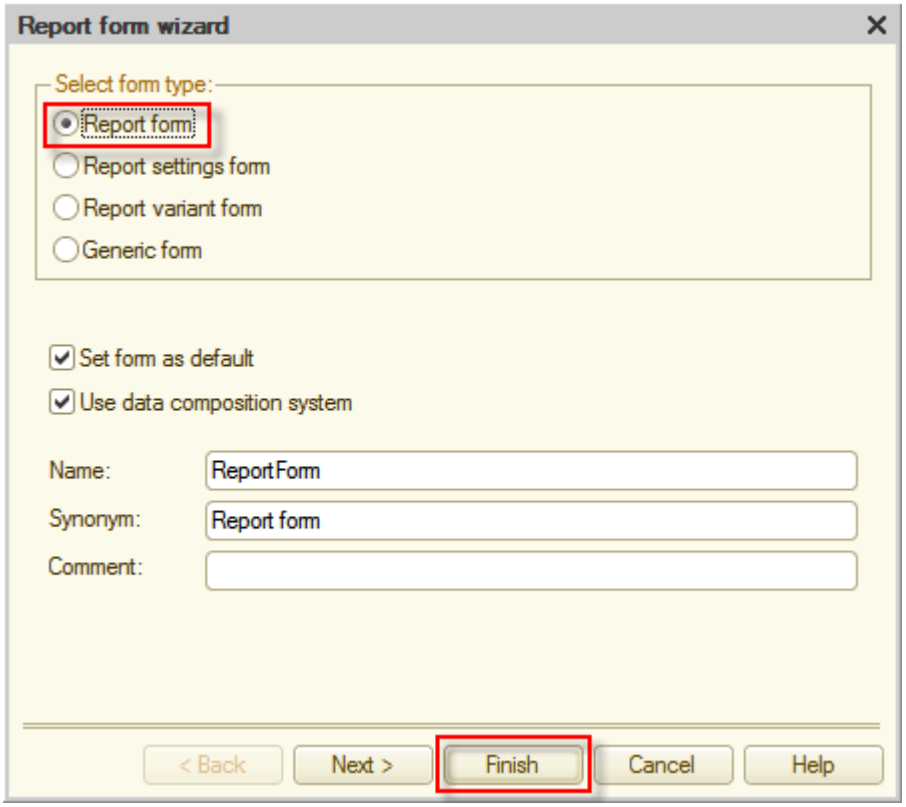

Figure 8-26. Adjusting a report form of a report

Return to the **Starting page work area** editor. This time you will be able to customize its appearance by adding to it newly created forms. Let it have an appearance with the **Financial transactions** register list form and the **Cash balance** report form on the left side, and the **People** catalog list form on the right side.

For implementing this, click **Add** above the **Left column** list and select the list form of the **Financial transactions** accumulation register, and then click **OK**.

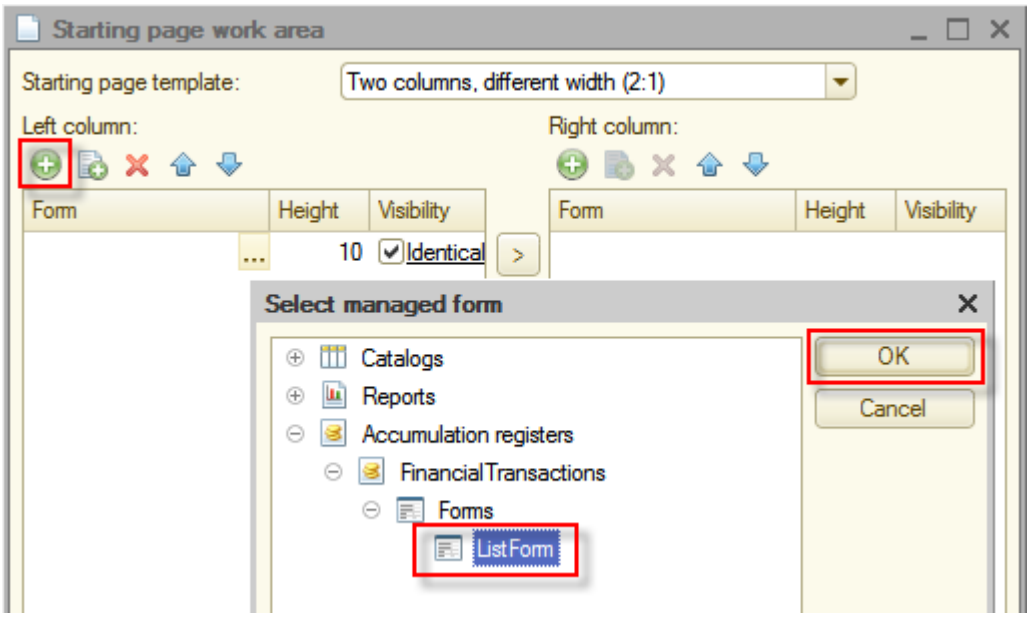

Figure 8-27. Adding the list form of the Financial transactions accumulation register to Start page

In the same way, add the report form of the **Cash balance** report.

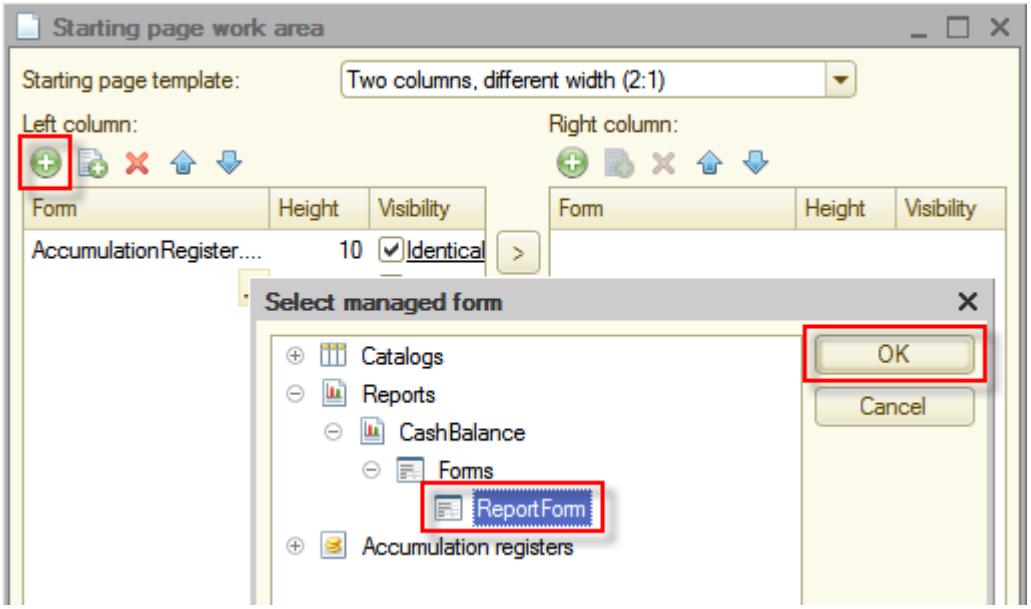

Figure 8-28. Adding the Cash balance report form to Start page

Repeat the same steps for **Right column** of **Start page**. Here, add the list form of the **People** catalog.

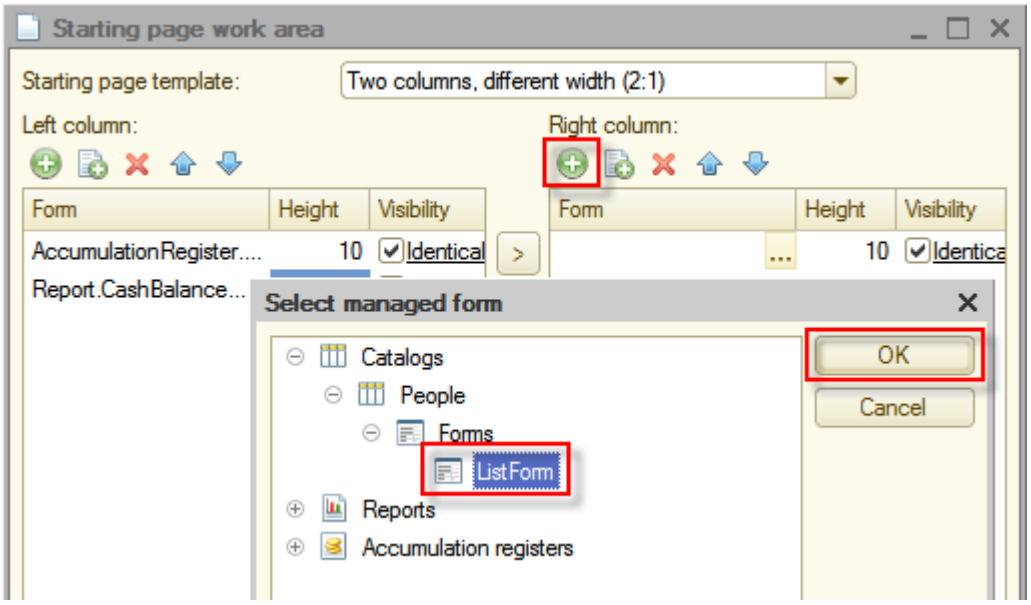

Figure 8-29. Adding the list form to the Start page

Start the application in the 1C:Enterprise mode and look at **Start page**.

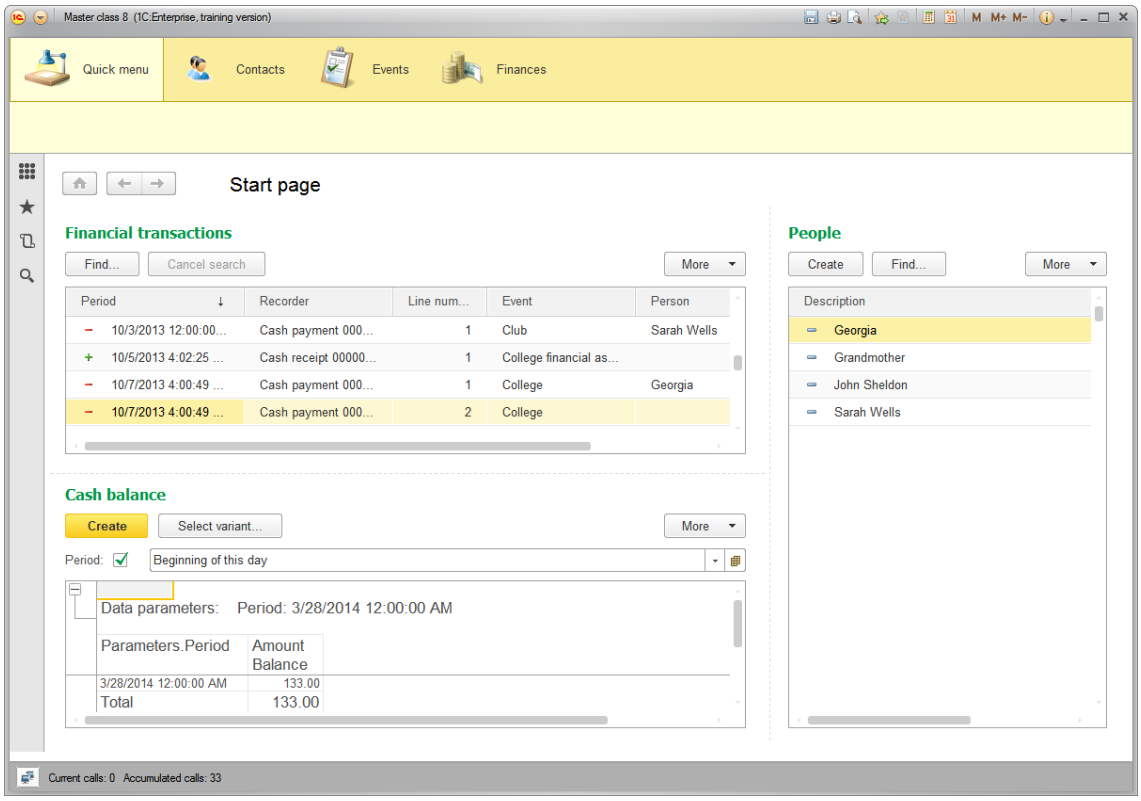

Figure 8-30. Start page in the 1C:Enterprise mode

Now **Start page** looks more useful. From here users can easily check financial transactions, manage the **People** catalog, and see how much cash they have available at a moment.

Command interface of Main section

At the current moment, only three objects are available on **Start page**. What if users would like to have more objects available? Is it necessary to create object forms for all of them and then manually place each one of them on the **Start page?** No, it is not necessary. If you place many forms on **Start page,** the interface will be overloaded and thus not user friendly. To make users able to access more objects from **Start page** without making the interface of it overloaded, you can adjust **Command interface** of **Main section** as you did with **Command interface** of subsystems.

In the Designer mode open **Properties** of root node in the **Configuration** object tree, and click **Open** link of **Main section command interface** property.

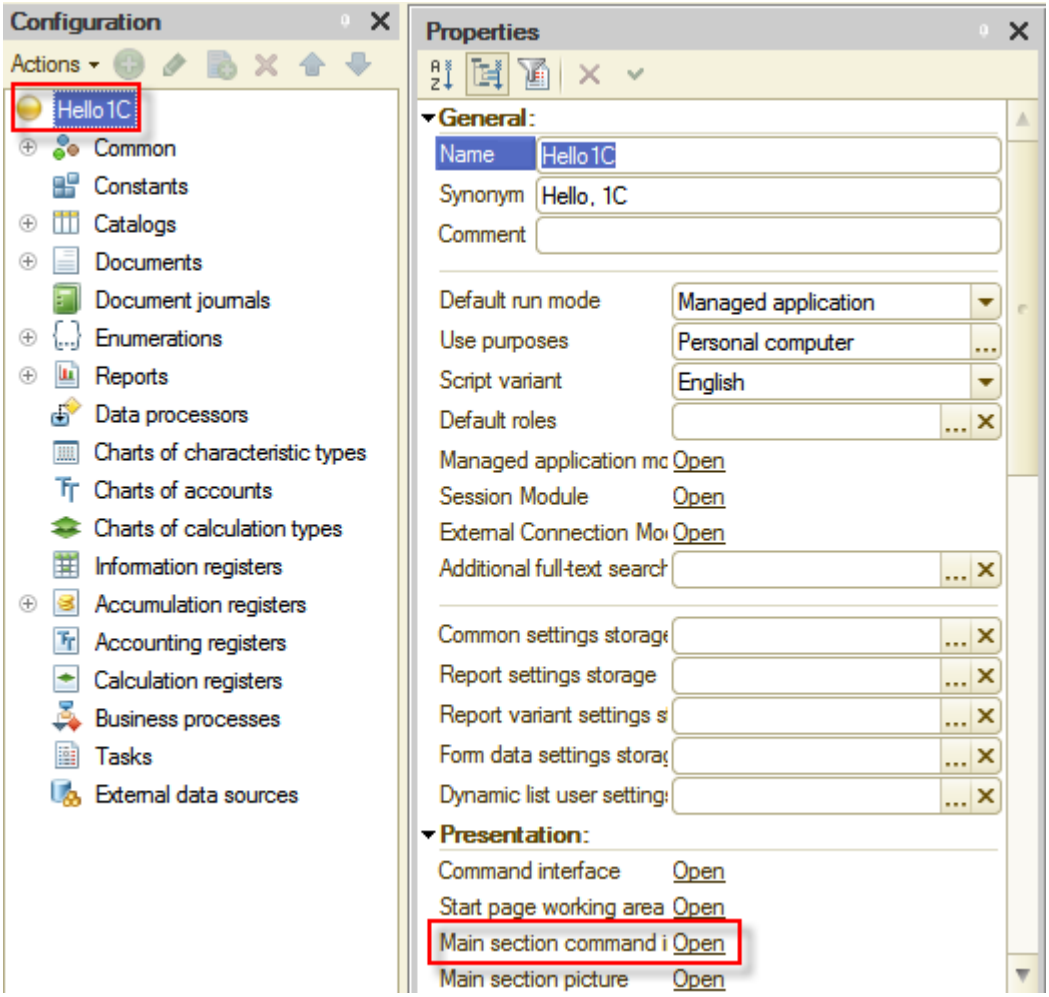

Figure 8-31. Opening Command interface of Main section

In the opened window select configuration objects and place them in **Command interface** of **Main section** using > button. Afterwards arrange objects in **Command interface** by dragging them as it is shown below:

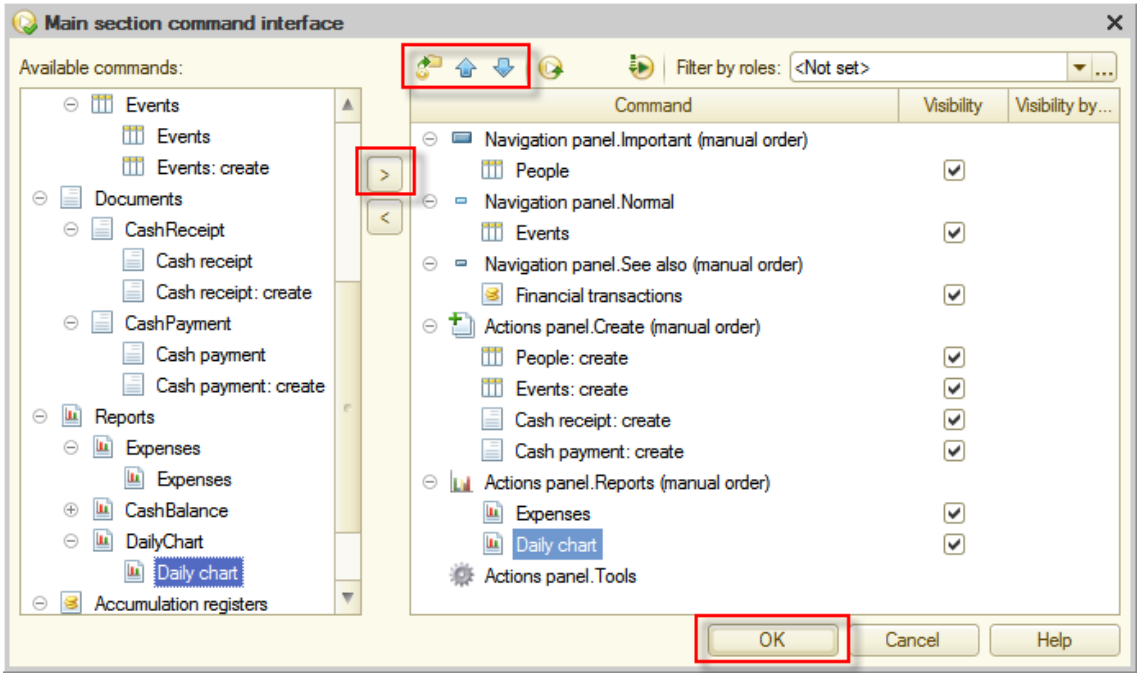

Figure 8-32. Adjusting Command interface of Main section

See how **Start page** looks now. Start the application in the 1C:Enterprise mode.

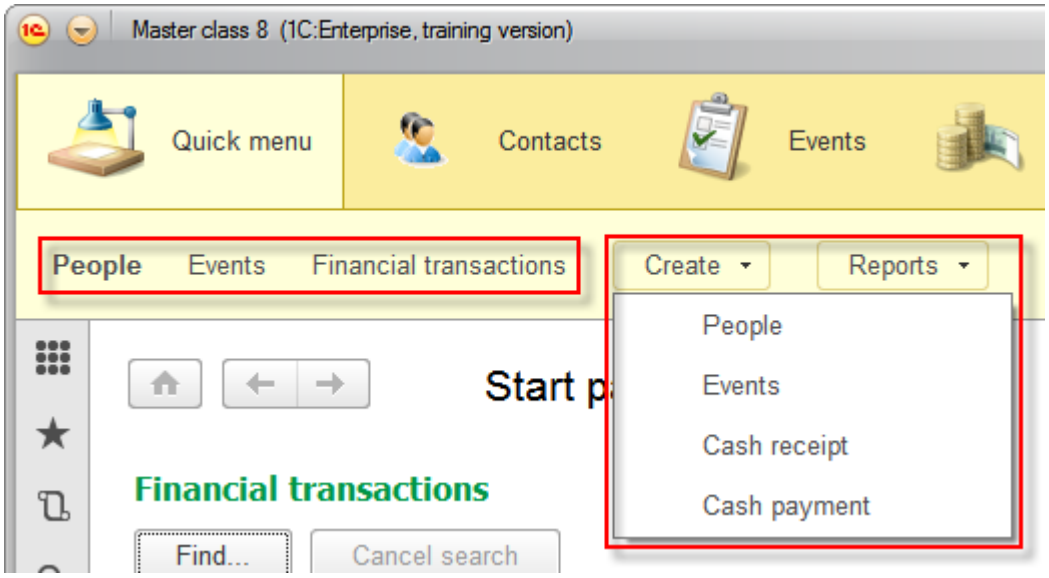

Figure 8-33. Adjusted Command interface of Start page in the 1C:Enterprise mode

Excellent! Now everything is at fingertips of users. By the way, using window separators, users can adjust the interface to increase more often used items, which is very convenient.

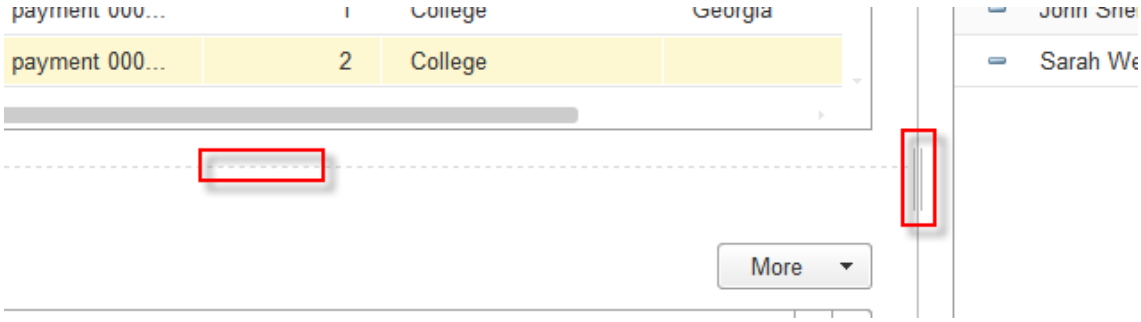

Figure 8-34. Separators

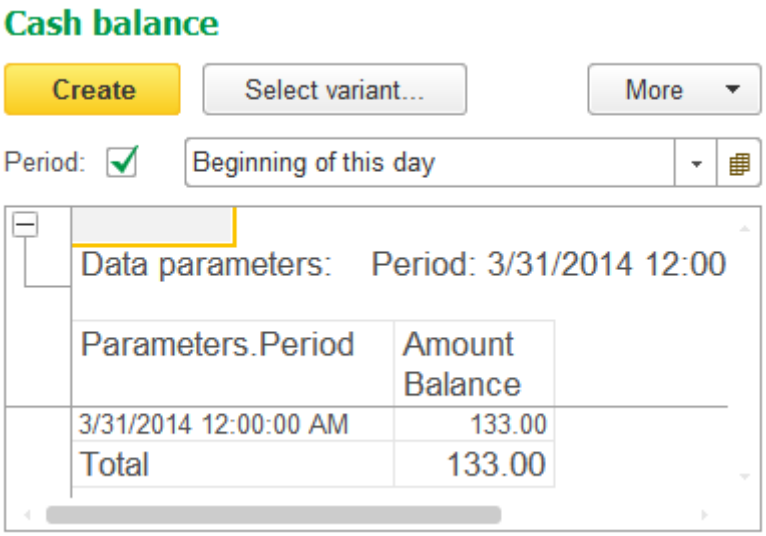

Figure 8-35. The report form with adjusted size

Managed forms

If you look more precisely at register and catalog lists on **Start page**, you will see that the information is not very conveniently displayed. In the register list, there is odd information, but on contrary, in the catalog list it is not enough.

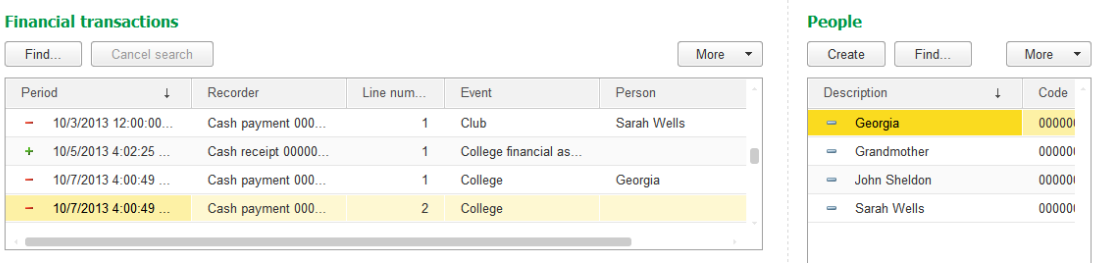

Figure 8-36. Start page

To amend the situation, return to the Designer mode. Start with the register and open the recently created list form of the **Financial transactions** accumulation register.

Until now, all forms that you saw were generated automatically, even those that were created with one click to be placed on **Start page**.

There is no need to manually draw forms in 1C:Enterprise 8. The developer only needs to configure the form composition in a hierarchical tree inside the top panel of the form editor, and the appearance of the resulting form is displayed as a preview inside the bottom panel of the form editor.

The platform automatically calculates positions and sizes of items on the form.

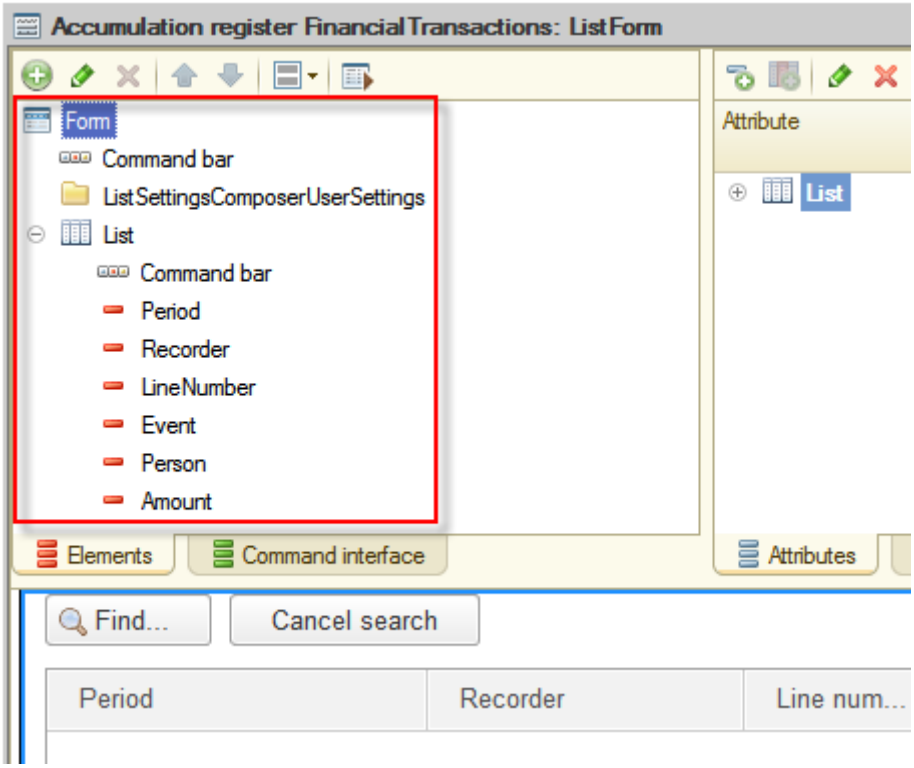

Figure 8-37. Managed form editor

Now, you will configure this list form. Remove odd the **Recorder** and the **LineNumber** attributes. Select the desired attribute in the list, and click **Delete current item** ✖(Del).

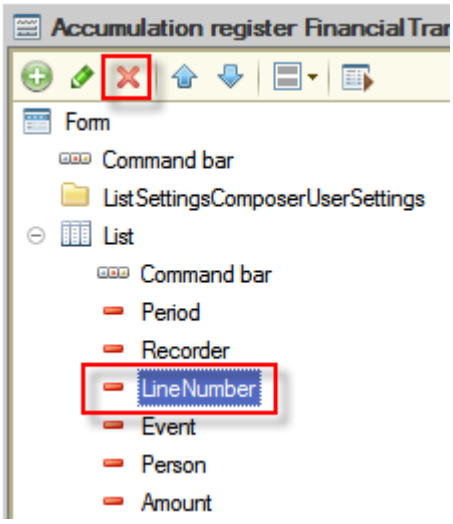

Figure 8-38. Deleting attributes of the managed form

After deletion of odd form items, the platform will redraw the form and display the preview as a user will see it in the 1C:Enterprise mode. The current form will look as follows:

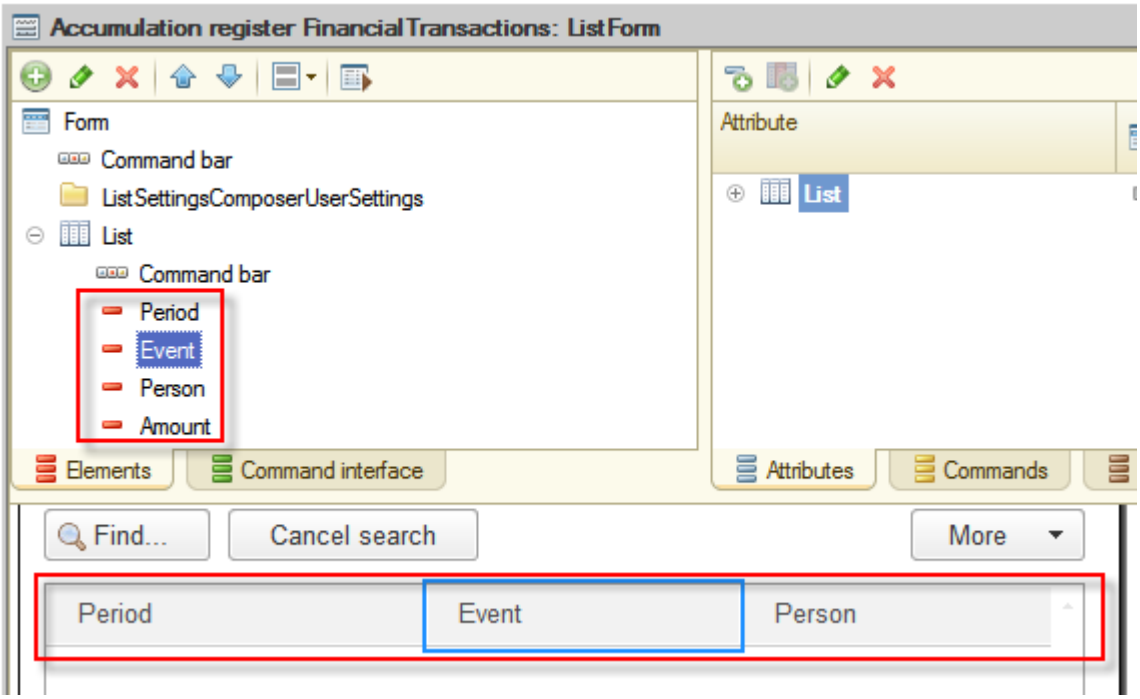

Figure 8-39. Configuring a managed form

In addition to deletion of odd form items, also adjust presentation of remaining form items.

In the 1C:Enterprise 8 platform, many properties that affect the data presentation in the interface are placed directly in properties of attributes of configuration objects. It is enough to describe attribute presentation settings in the object metadata itself. Then, the platform will automatically apply these properties when display attributes in all forms where they are placed.

Of course, if there is a need to change a presentation for a particular form, then those changes can be done in properties of the form item.

To begin, change the width to **10** for all form items in **List** table of list form of **Financial transactions** accumulation register.

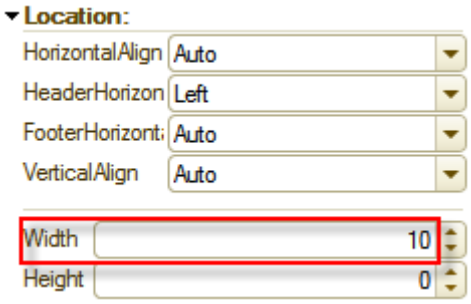

Figure 8-40. Adjusting the width of the managed form item

For the **Amount** form item select **Right** as a value of **HorizontalAlign** property so amounts could be observed easier.

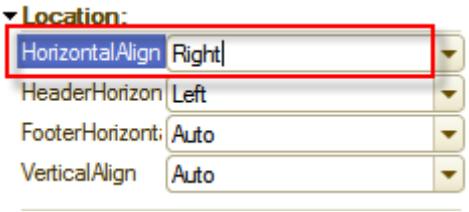

Figure 8-41. Adjusting alignment of the form item

Then give a more user-friendly name to the **Period** form item let it be **Date**.

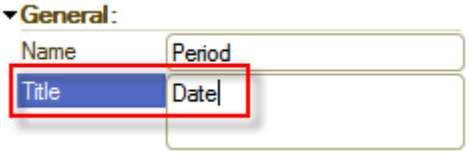

Figure 8-42. Adjusting a title to a form item

Now, start the application in 1C:Enterprise mode and look at the list form of the accumulation register.

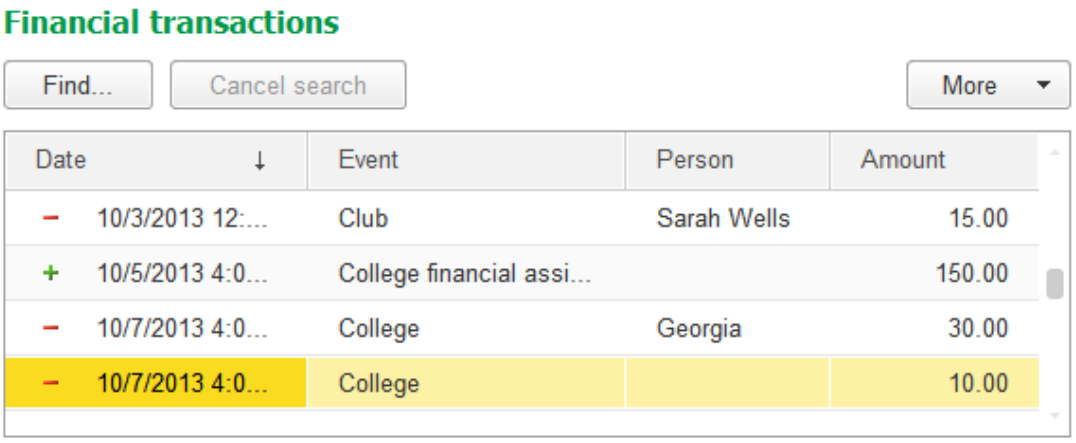

Figure 8-43. The list form of the Financial transactions accumulation register

You can verify on your own whether this form looks the same by opening it in the **Finances** section or by opening **Main menu**, then clicking **All**

functions... and opening this form in the form list. The form will look the same. Settings of the managed form are applied to all places of the applied solution where this form is used.

Standard and ordinary attributes

Continue to the **People** catalog. The **People** list on **Start page** currently contains less information. Besides, the **Description** word does not clearly present what is contained in the corresponding column. In addition, the list includes the **Code** column that seems not very useful to users.

As a start, you will change **Synonym** of the **Description** attribute to **Full name**. One way to do this is to open a list form and in **Properties** of the **Description** form item enter a synonym, as you did for the **Period** form item in the list form of the **Financial transactions** accumulation register.

However, this time, a different procedure is more suitable. There are several forms in the **People** catalog now, the list form and the catalog item form. The list form was created in the Designer mode, and the catalog item form is generated automatically by the platform. If you will change **Synonym** in the list form only, the catalog item form will not be changed. Therefore, it is better to change **Synonym** to **Full name** in one place, which is **Properties** of **Standard attribute**. To do this, find the **People** catalog in the **Configuration** object tree, right-click on it and then click **Standard attributes**.

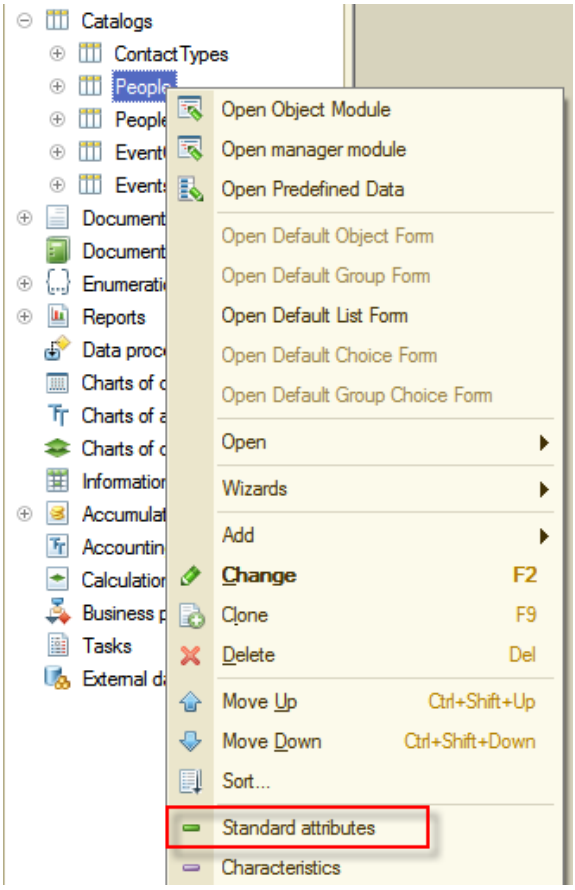

Figure 8-44. Opening Standard attributes

In the opened window find the **Description** attribute, and then in **Properties** specify **Full name** in the **Synonym** property.

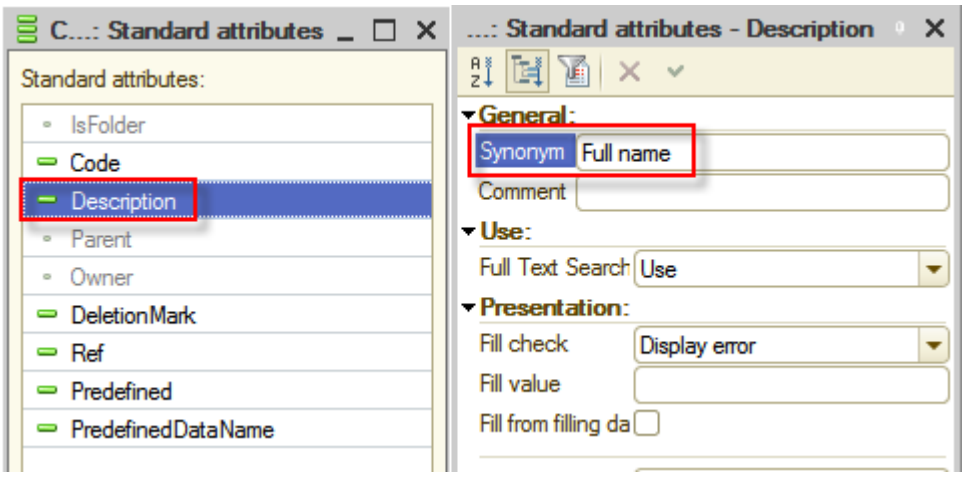

Figure 8-45. Standard attributes

After that, select **Comment** attribute, and in **Properties** select the **Multiline mode** and the **Extended edit** check boxes since in **Comment** is used to keep notes regarding the person. Enabling these options will make adding and editing of notes more convenient.

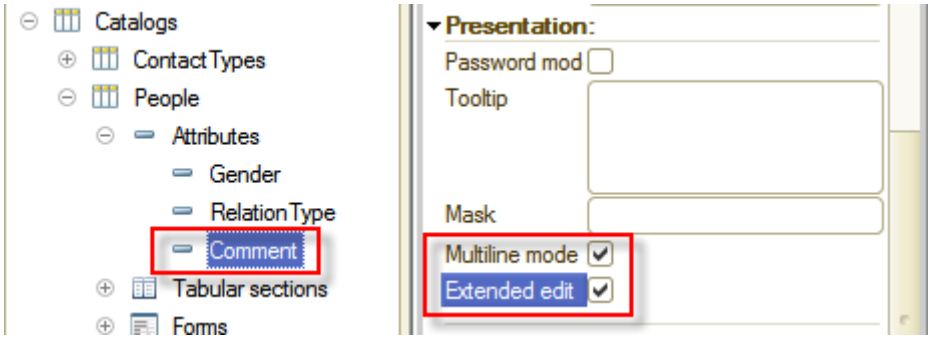

Figure 8-46. Enabling the Multiline mode and the Extended edit options

Next, open the list form of the **People** catalog and delete the **Code** form item. Then to add the **Comment** column to the list, expand the **List** form attribute in the right panel and drag the **Comment** attribute inside the **List** form table in the left panel.

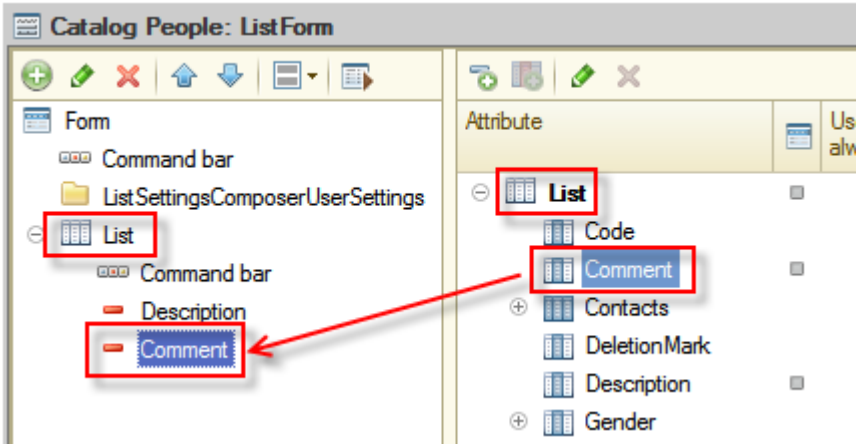

Figure 4-47. Adding the Comment attribute to the managed form

Set **Width** properties of the **Description** and the **Comment** form items to **10**. You can select more than one form item by holding down **Ctrl** key and

clicking on each attribute. When more than one form item is selected, you can change most properties for all of them in **Properties**.

Now, start the application in the **1C:Enterprise** mode and review the changes to the list form of the **People** catalog. This time the **People** list on **Start page** looks more useful.

Figure 8-48. The adjusted list form of the People catalog

The list form of the **People** catalog is also updated in the **Contacts** section where it is also placed. In addition, the item form of the catalog, which is generated by the platform automatically, also reflected all changes that you have recently made, such as **Synonym** of **Description** attribute and **Multiline mode** of **Comment** attribute.

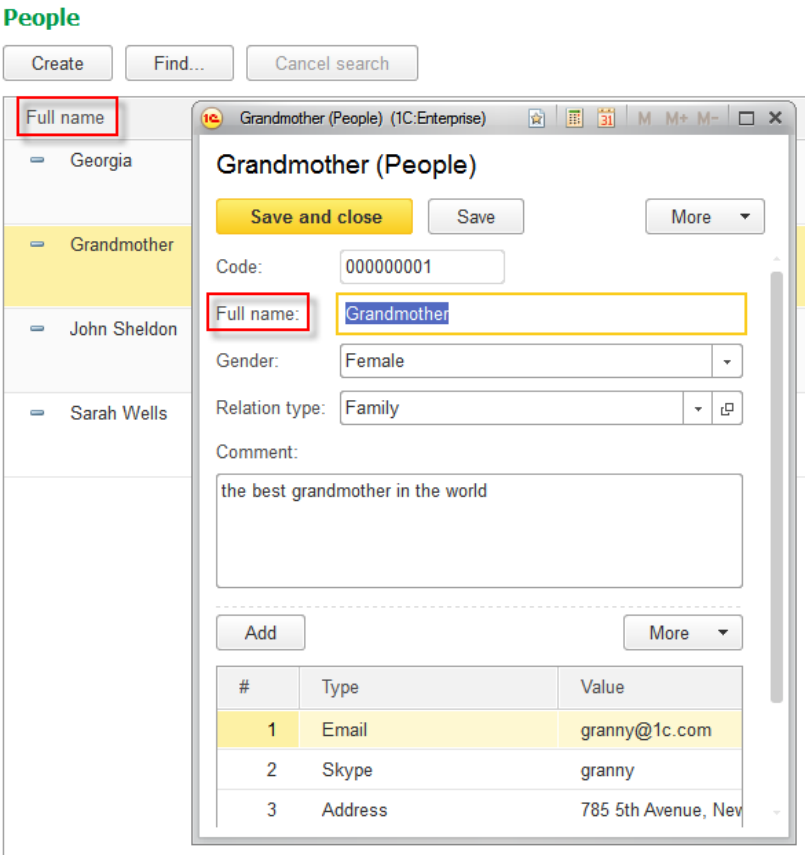

Figure 8-49. The adjusted item form of the People catalog

Make a couple more improvements to the application. Adjust the **Events** catalog to make it more user-friendly. To do so, set **Title** as **Synonym** for the **Description** form item in **Standard attributes**, and set **Attendee** as **Synonym** for the **Participant** attribute of the tabular section, and then select the **Multiline mode** and the **Extended edit** check boxes for the **Details** attribute.

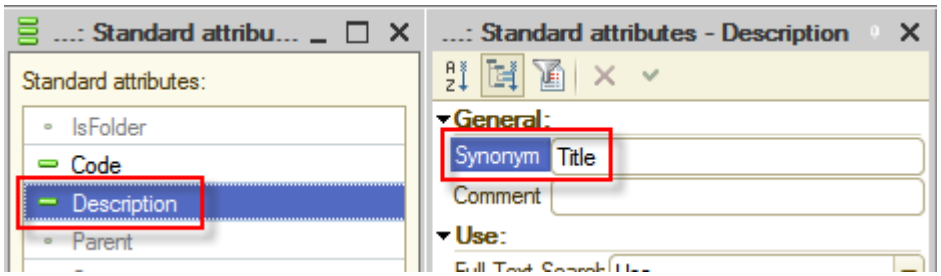

Figure 8-50. Setting Synonym for the Description Standard attribute

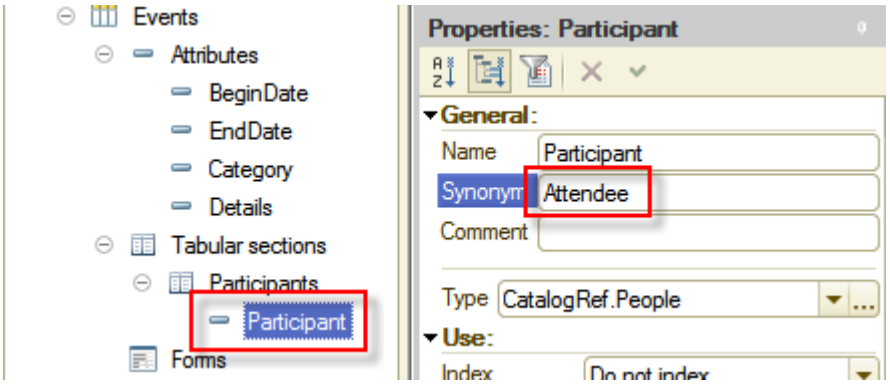

Figure 8-51. Setting Synonym for the Participant attribute of tabular section

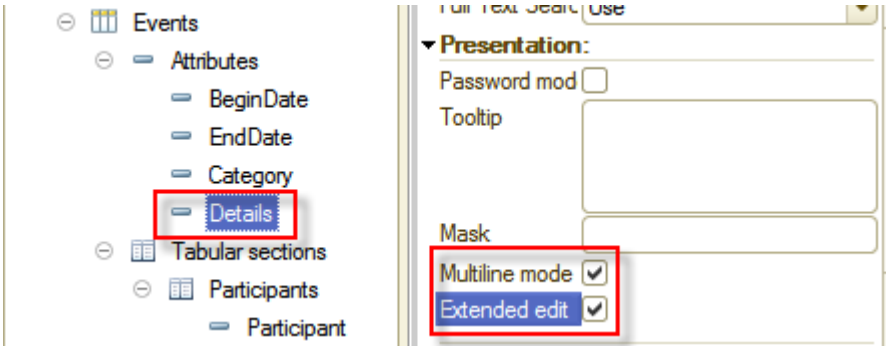

Figure 8-52. Enabling the Multiline mode and the Extended edit options

Now, you can review the changes. The **Events** catalog now looks much better.

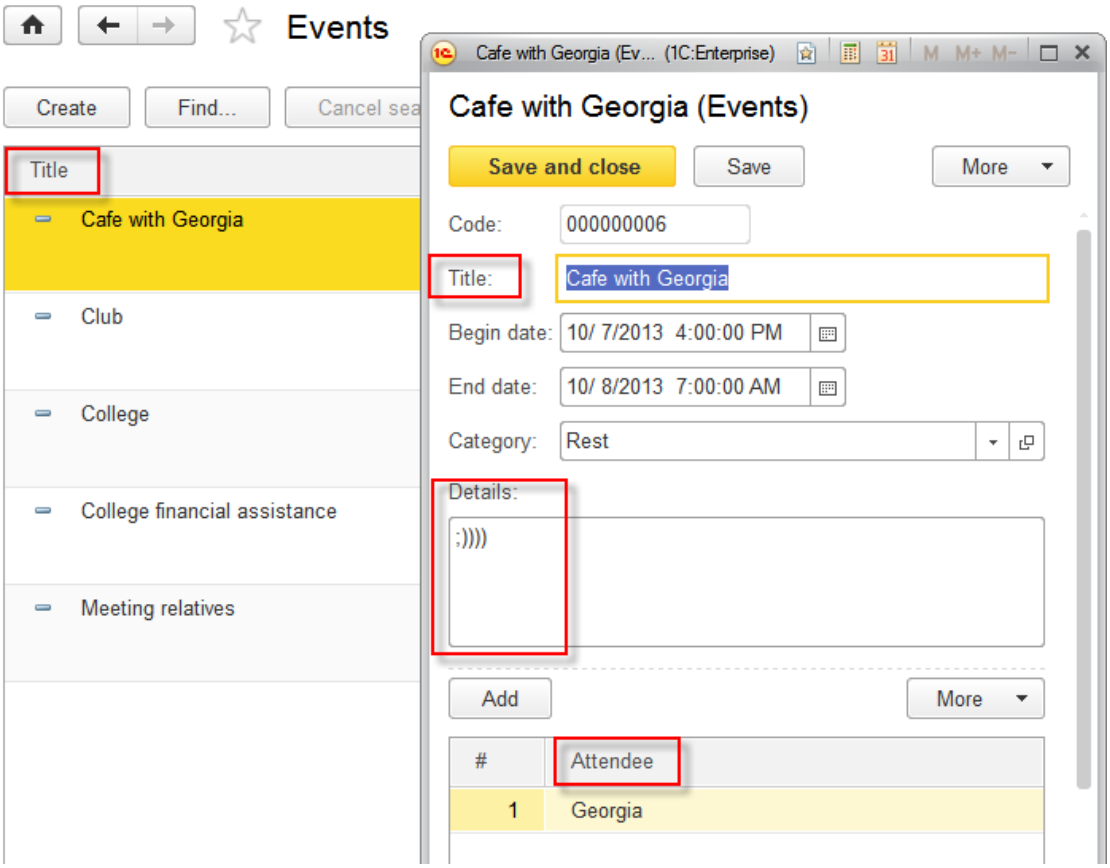

Figure 8-53. The adjusted Events catalog

Object presentations

You might already noticed that names of items in **Navigation panels** of sections and in **Create** groups of **Actions panels** are the same, both opening of list form and creating a new item are plural. A beginner user will be confused with that.

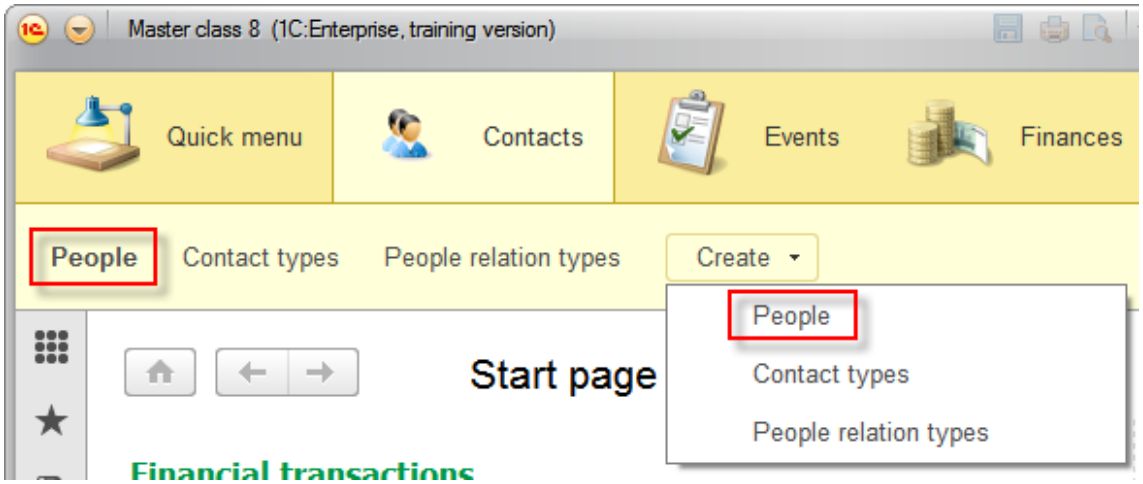

Figure 8-54. Same names for opening the list form and creating a new item

It is easy to fix. All that you need to do is to adjust **Object presentation** in **Properties** of each catalog. In other words to specify its singular name. Do this for each catalog as follows:

- For the **Contact types** catalog set **Object presentation** to **Contact type**
- For the **People relation types** catalog set **Object presentation** to **People relation type**
- For the **People** catalog set **Object presentation** to **Person**
- For the **Event categories** catalog set **Object presentation** to **Event category**
- For the **Events** catalog set **Object presentation** to **Event**

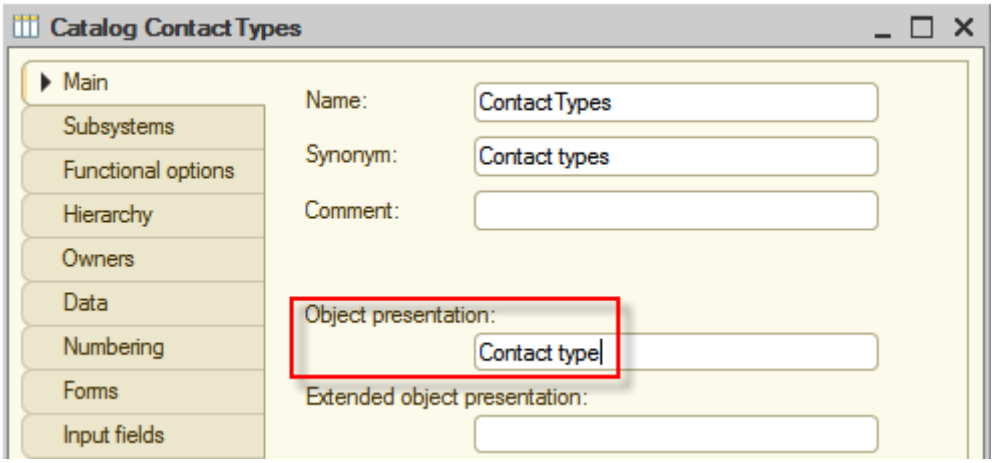

Figure 8-55. Adjusting Object presentation

Check how it looks like. Now names in the **Create** group are more grammatical.

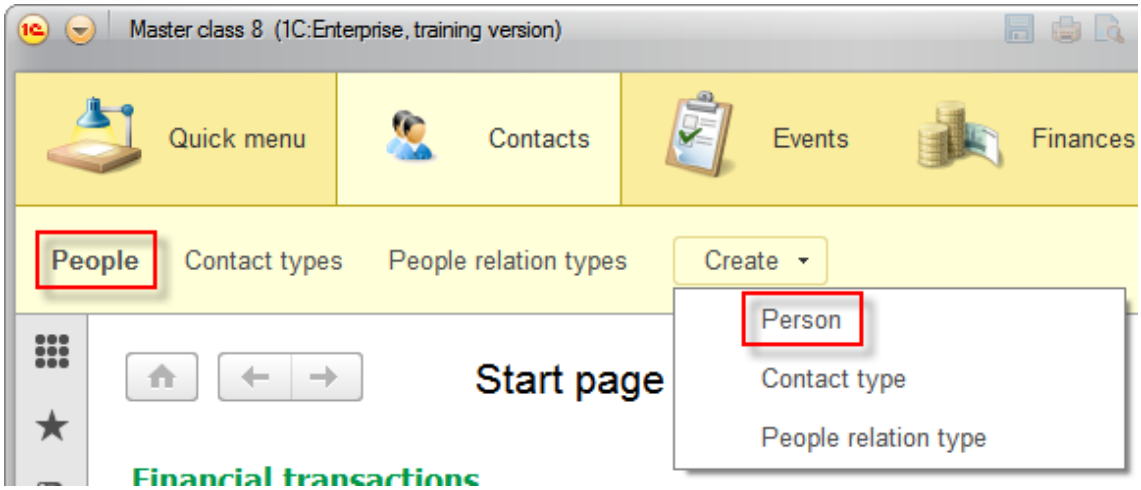

Figure 8-56. Difference between Synonyms and Object presentations

Quick selection of values

Now, try to change **Relation type** of the **People** catalog item or **Category** of **Event**, or **Type** in one of rows of contacts of a person. When you click a field that allows selecting values, to select a value you can either start typing first characters of a value description or click the **Show all** link and see the list of all items in a separate window. If you used the second method, to select a value, click the row with that value and then click **Select** or simply double-click the row.

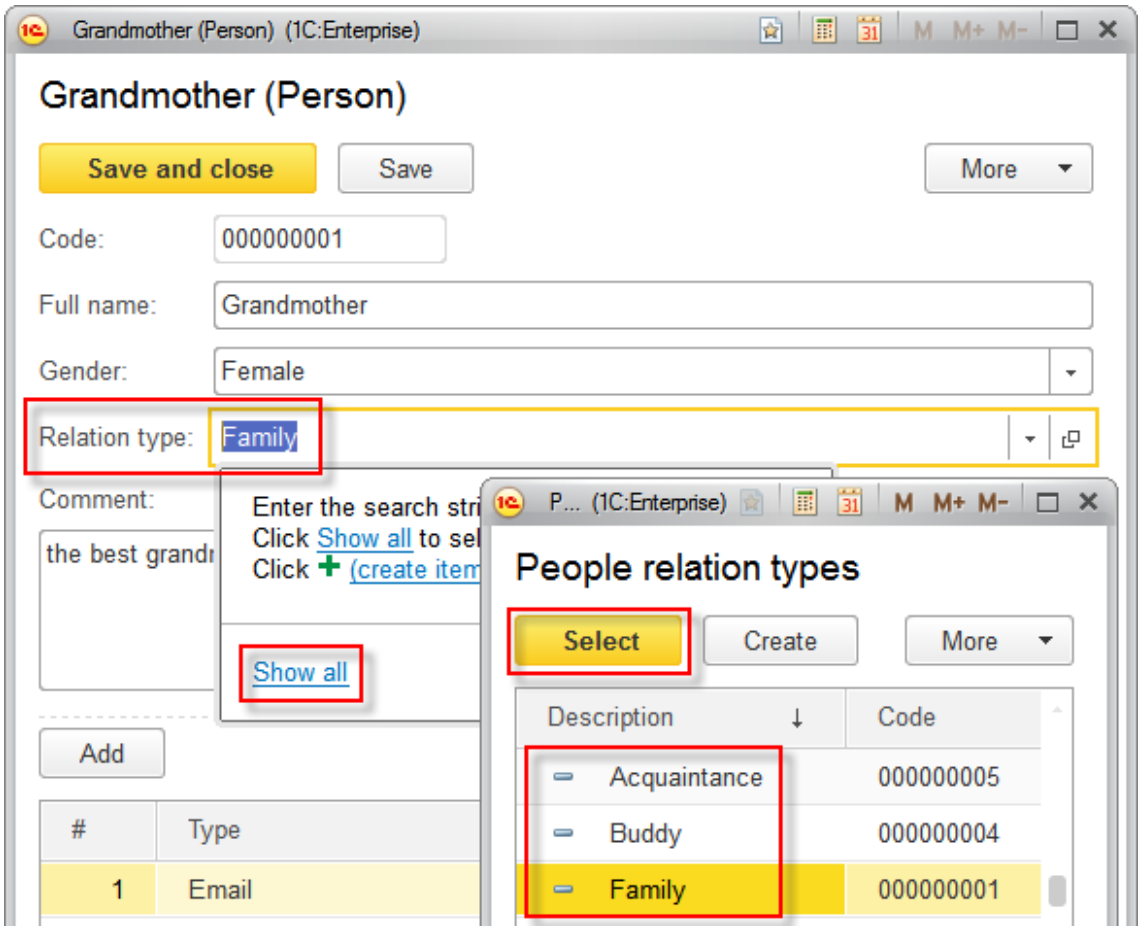

Figure 8-57. Selecting values from the list of catalog items

In cases when the list of items in catalog is short, opening an additional window is not necessary and slows down operation with an application. You can change the application behavior in **Properties** of the catalog to one when its items will be selected from a drop-down list. To do this, select the desired catalog in **Configuration** metadata object tree, and then select the **Quick choice** check box in its **Properties**. Select this check box for the **Contacts types**, the **People relation types**, and the **Event categories** catalogs.

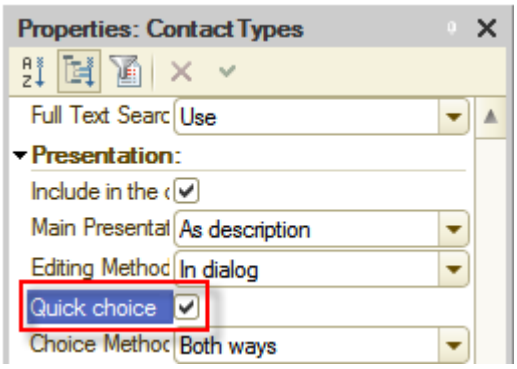

Figure 8-58. Enabling the Quick choice option

Now see it in the 1C:Enterprise mode. A drop-down list is displayed to select values.

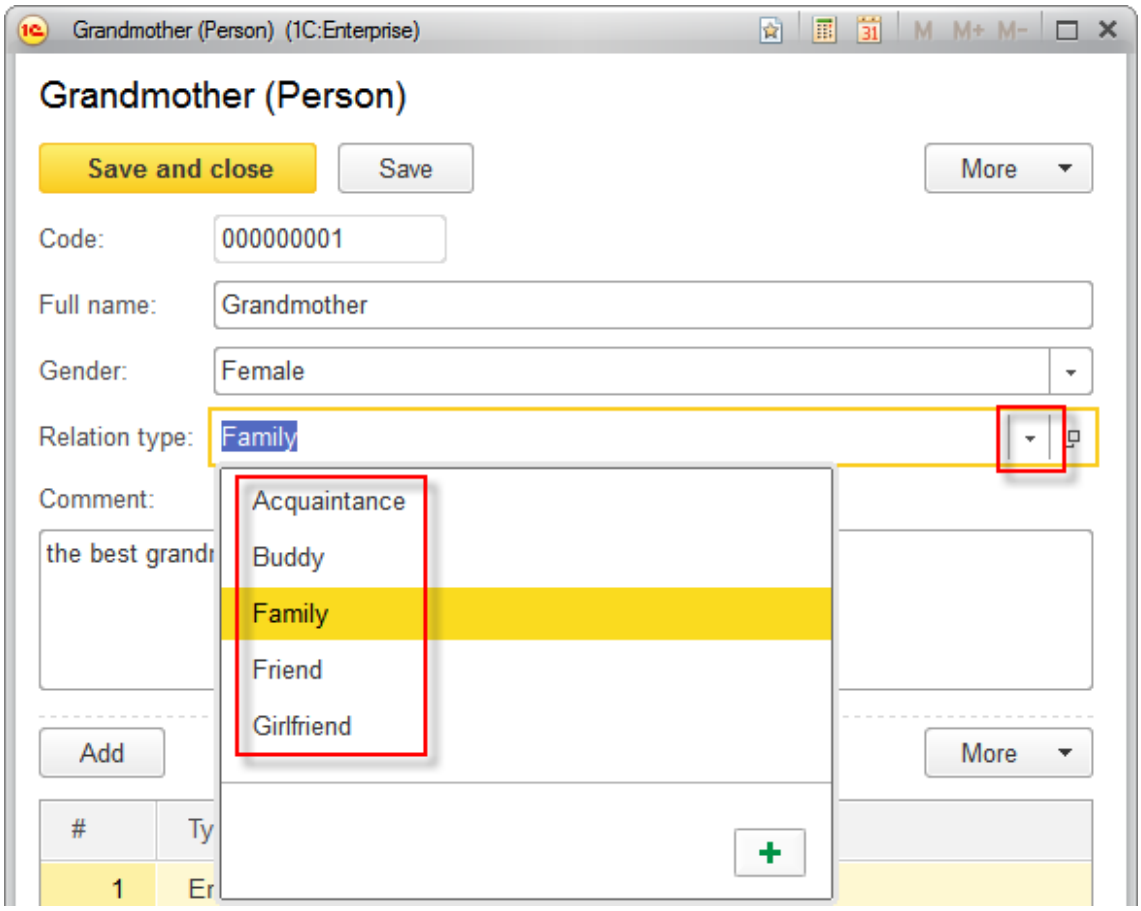

Figure 8-59. Drop-down list as Quick choice is enabled

Adjusting reports

Look at the **Daily chart** report. In general, it is fine, but a user may be confused about some technical information, the format of dates, and the legend. In addition, it would be much better to be able to see points of the plot directly on the chart, not only when hovering over them with the mouse.

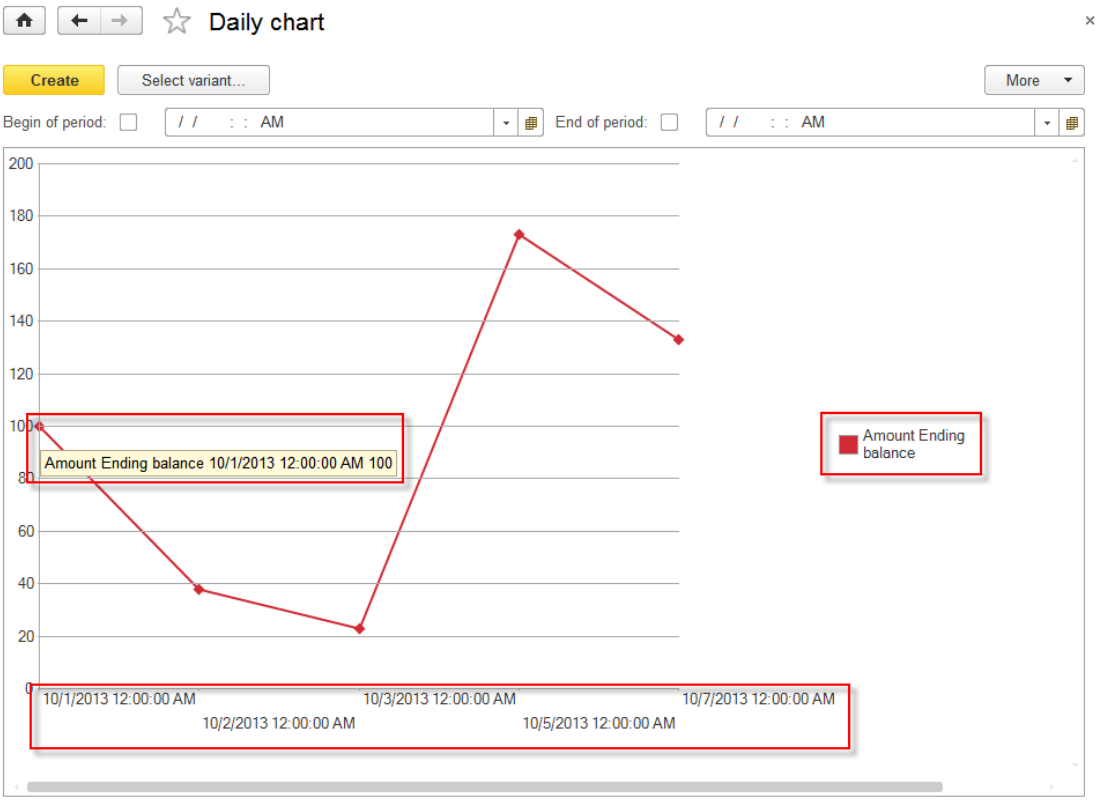

Figure 8-60. Daily chart report

It is easy to improve this chart. In the Designer mode**,** open the **MainDataCompositionSchema** template for this report.

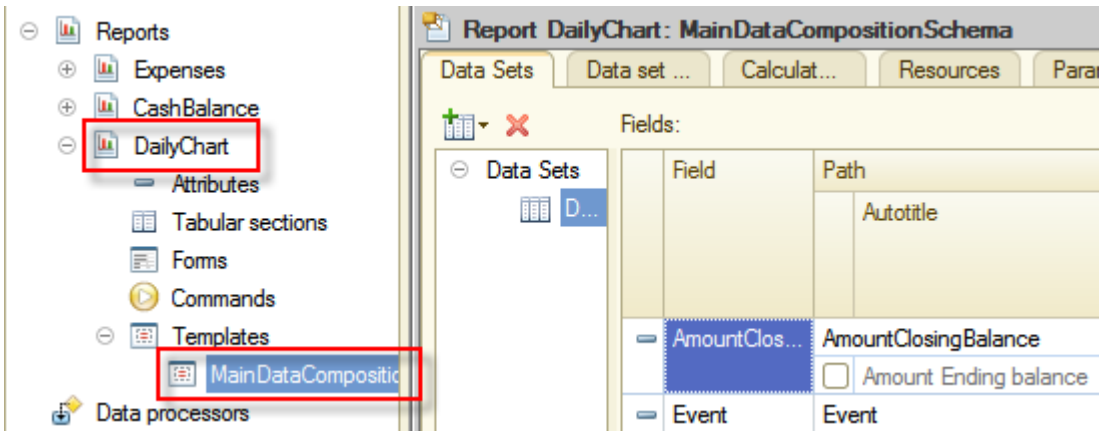

Figure 8-61. MainDataDompositionSchema of the Daily chart report

On the **Data sets** tab, adjust the legend by editing titles of **AmountClosingBalance** and **Period** fields. In addition, for **Period** field set a more user-friendly date format.

Next, set the title of the **AmountClosingBalance** field to **Balance**. To be able to do this, find this field in the list of fields and then select check box on the left to the title that you want to edit. Autotitle caption of this column will automatically change to **Title**. Then instead of **Amount Ending balance** enter **Balance**.

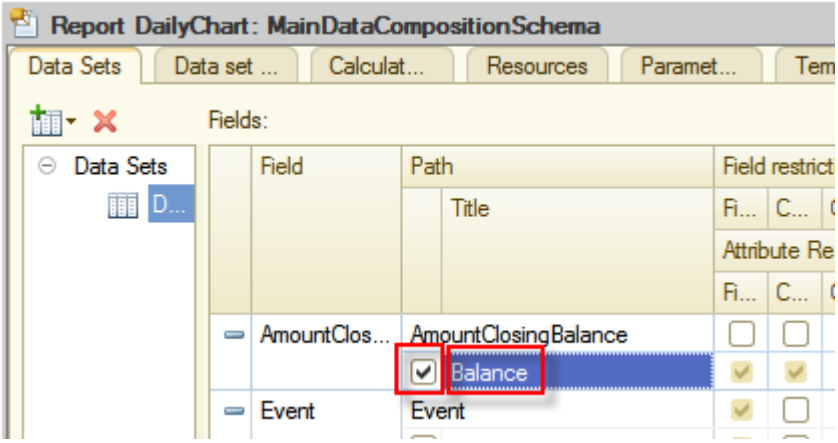

Figure 8-62. Setting the title of the AmountClosingBalance field

Do the same with the **Period** field, change its title to **Date**.

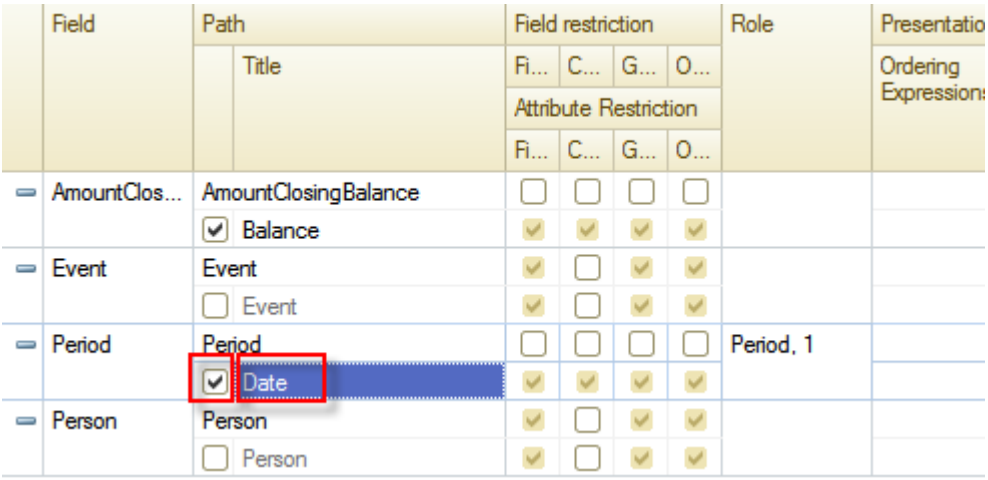

Figure 8-63. Setting the title of the Period field

Now for the **Period** field change the date format. Enter the appearance column of **Period** by clicking **Select** ...

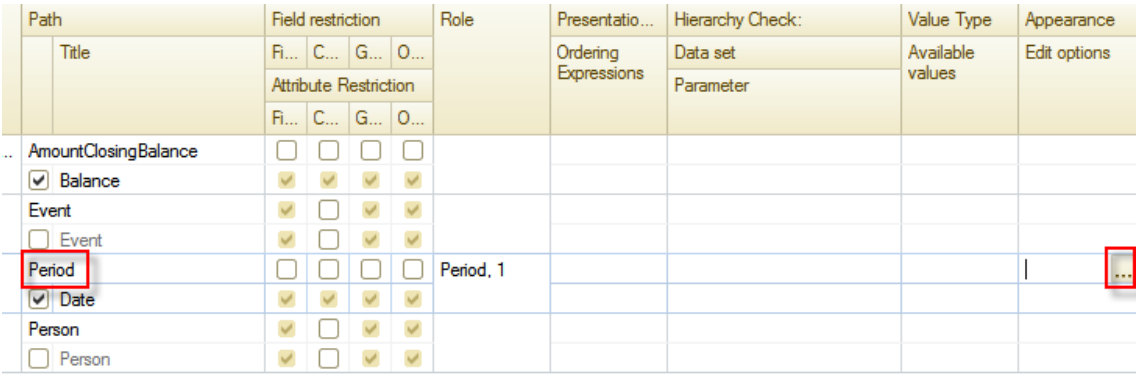

Figure 8-64. Editing the field appearance

Open the **Field format** editor, and find the **Format** property.

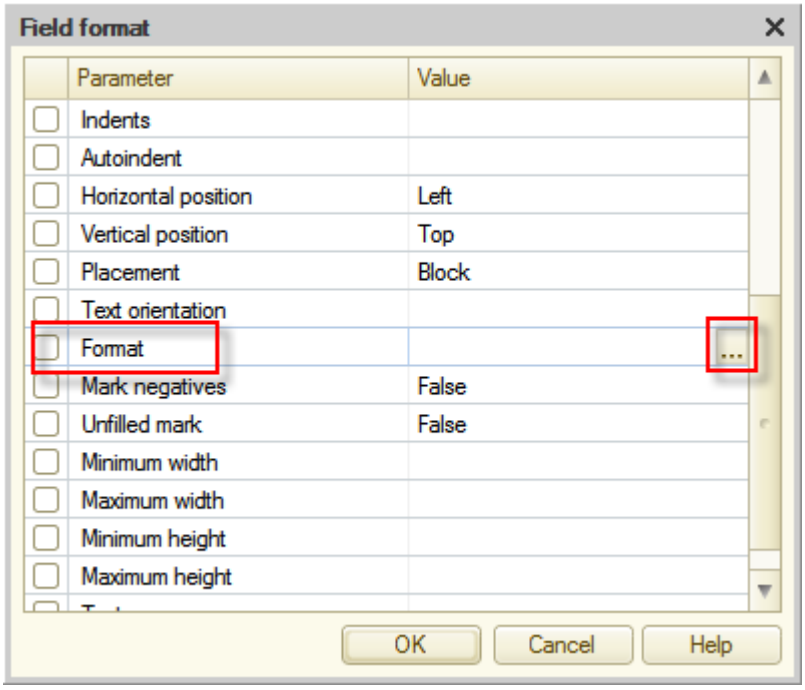

Figure 8-65. The Field format editor

Open **Format String Wizard** by clicking **Select** ..., and then click the **Date** tab and in the **Date format** field enter **MM/dd/yyyy**. After that, click **OK**.

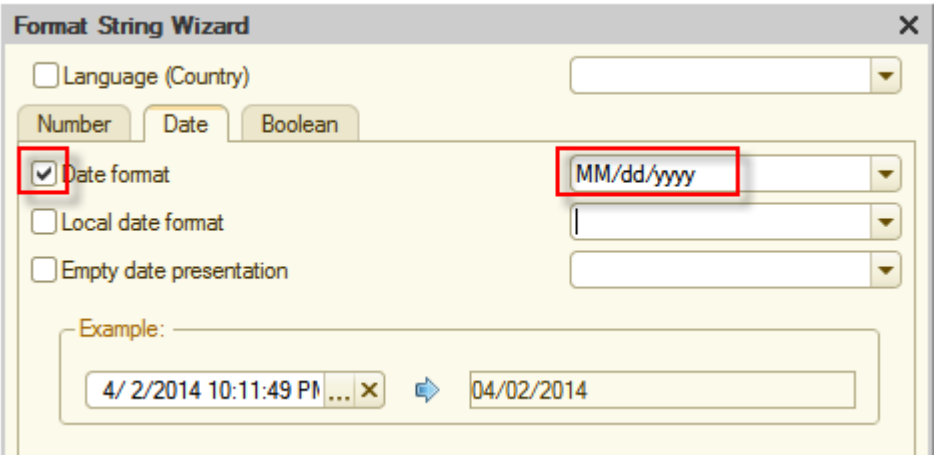

Figure 8-66. Format String Wizard

Next, click **OK** to close the **Field format** editor. After that value in the **Appearance** column on the **Data Set** tab for the **Period** field will look as follows:

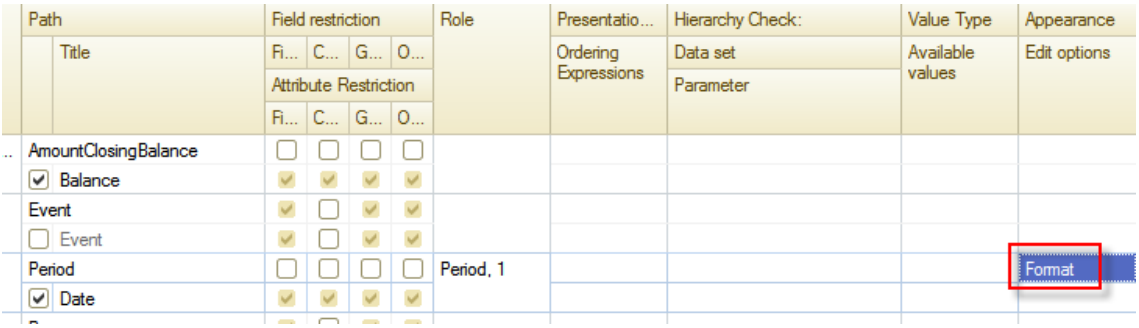

Figure 8-67. The adjusted Period field

The last thing is left to do is to adjust view of plot points. To do this, click the **Settings** tab. Then, on the bottom tabs click the **Other settings** tab.

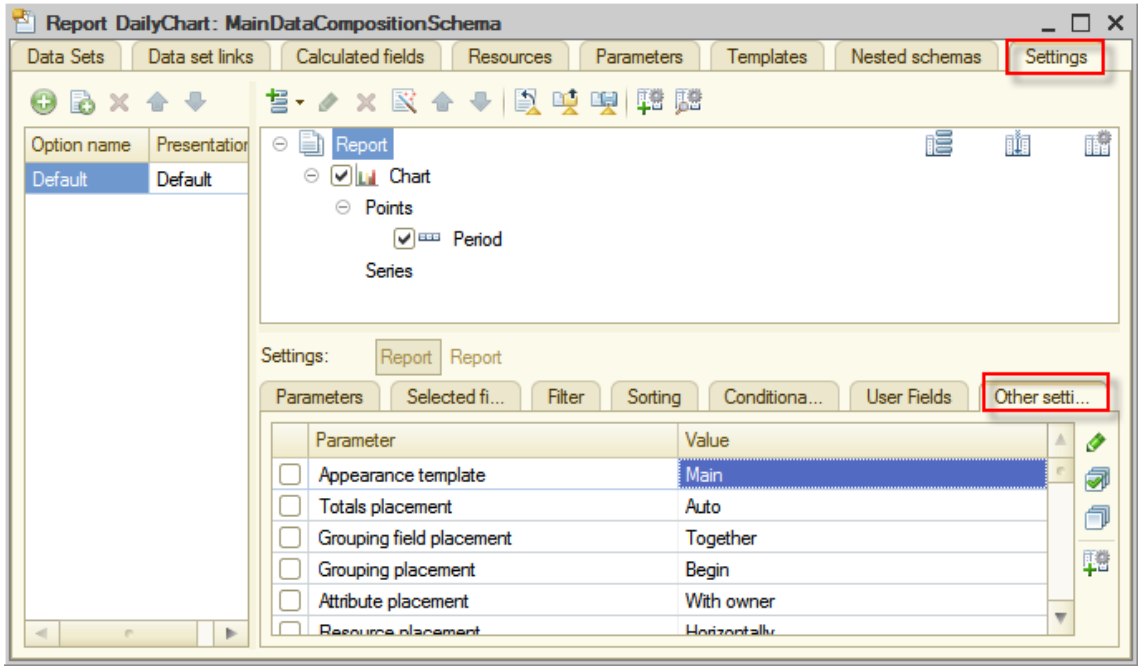

Figure 8-68. The Other settings tab

Find the **Label list** parameter under **Chart type**, and then from the drop-down list select **Value**.

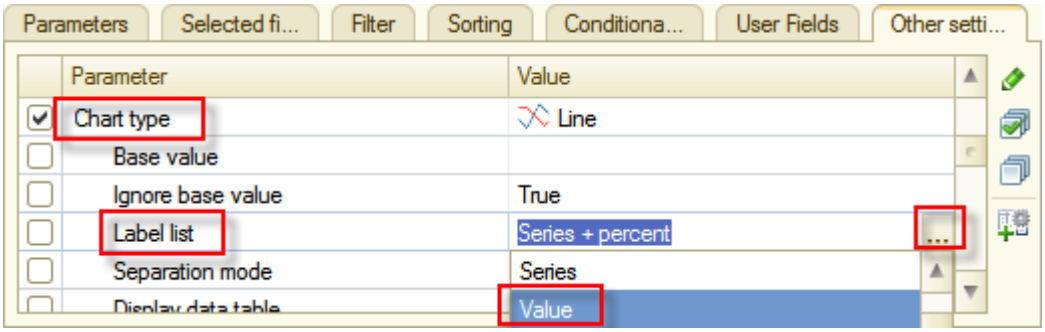

Figure 8-69. Selecting Value as a value of the Label list parameter of Chart type

Now, check it in the 1C:Enterprise mode to see the result of changes.

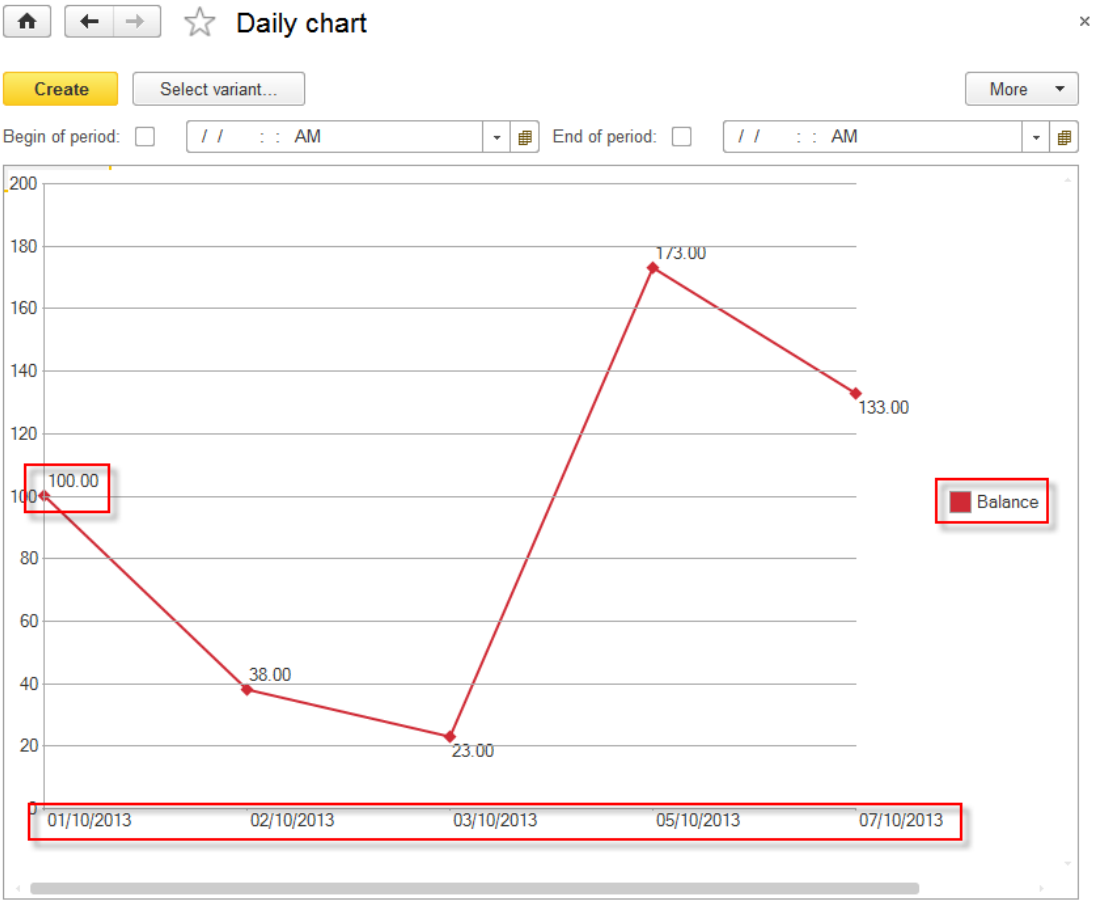

Figure 8-70. The adjusted chart

Excellent! Everything that was required is implemented. The chart shows meaningful legend and values are displayed next to plot points. This allows analyzing finances easier.

Report variants

You might ask the right question: "Do I have to open the report the Designer mode each time when I need to change its view settings?" After all, the user might not be able or allowed to use the Designer mode, for example, when connecting through HTTP protocol. The answer is no, you do not have to use the Designer mode each time you need to change something in the report settings. Most of report settings, including view settings, can be changed in the 1C:Enterprise mode without engaging a developer.

As an example, you will change the **Expenses** report. The default view of the report, meaning the settings that the developer made for it, are as follows:

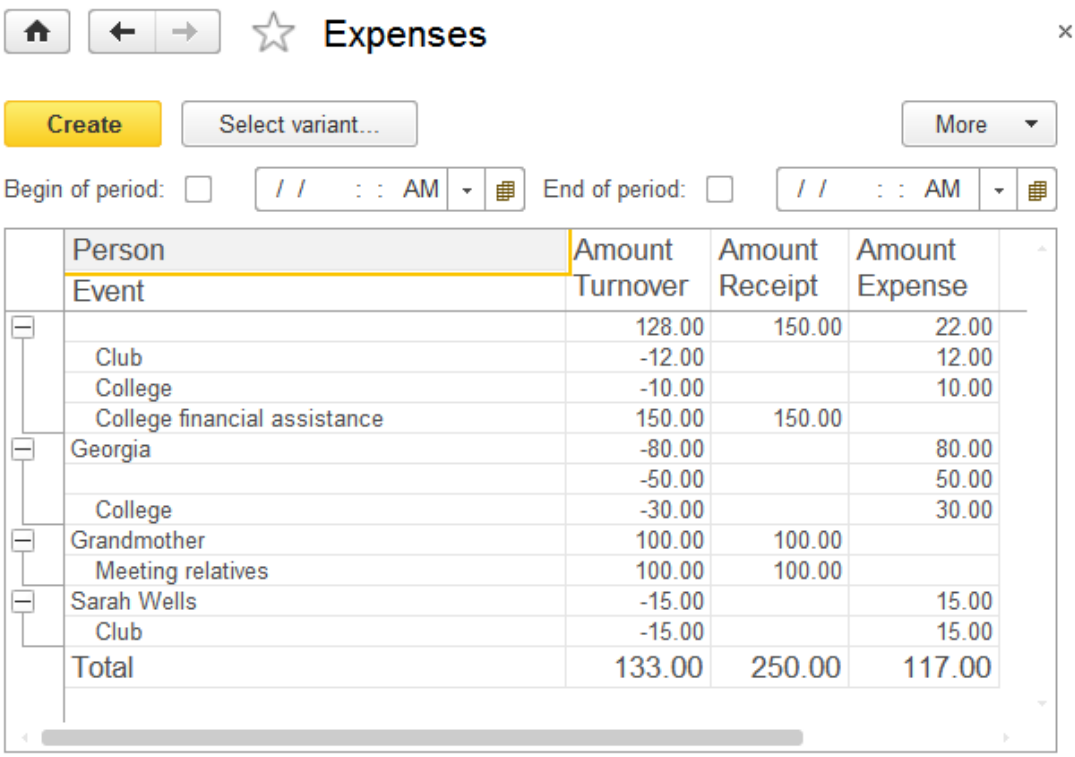

Figure 8-71. The Expenses report with default settings

Imagine that a chart would be easier for the analysis, so you need to change the report into a chart. Click **More** and then click **Change variant...**

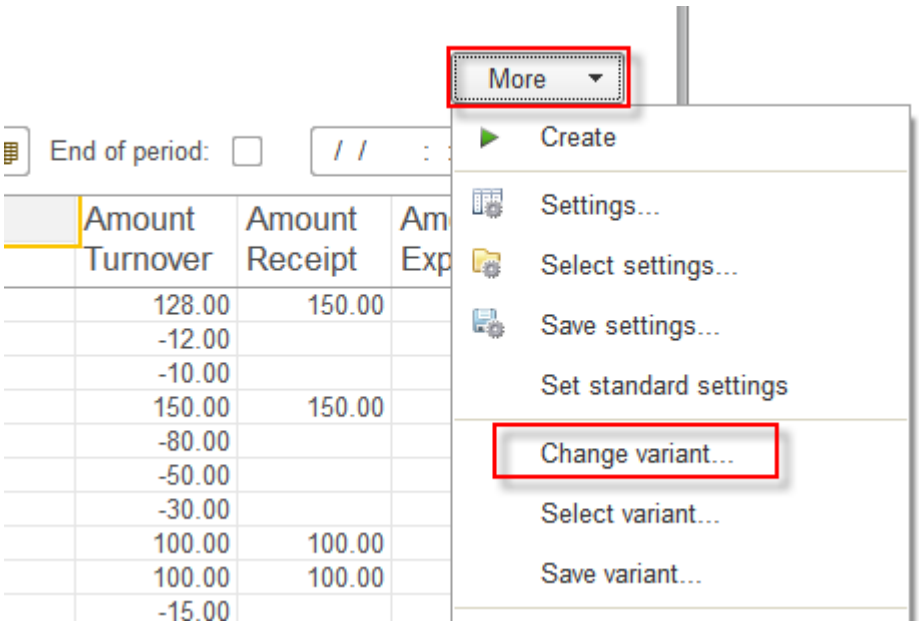

Figure 8-72. Changing a report variant

You will see the report settings window, the same as you used in the Designer mode when configured the data composition schema for this report. The default settings are displayed. The one that you have configured in the Designer mode for this report.

Click the **Person** node, and then click **Delete** (Del) to delete these settings. Confirm the deletion when the dialog appears.

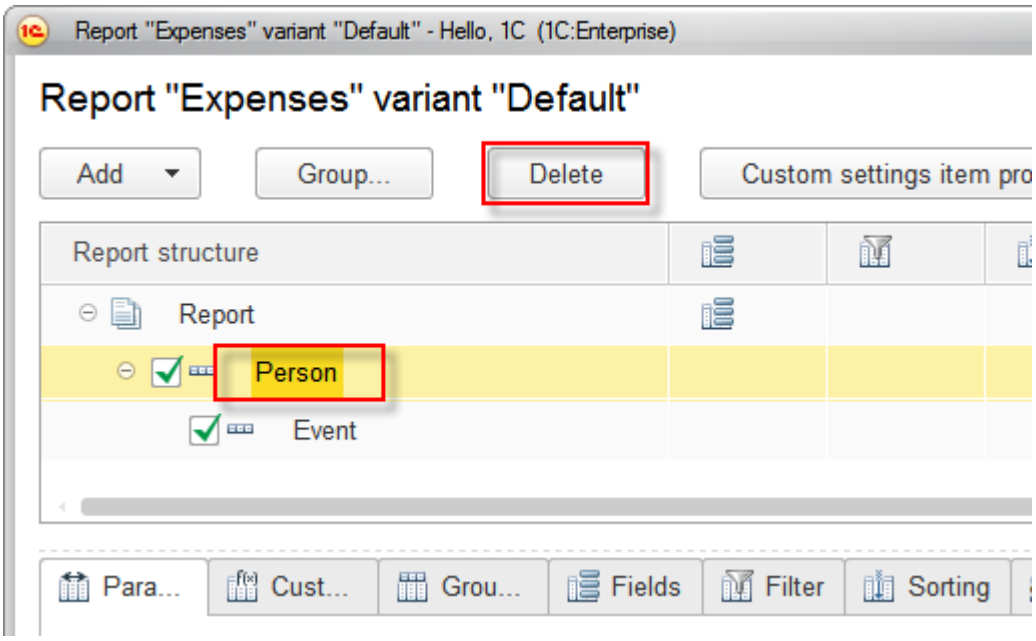

Figure 8-73. Deleting settings of the current variant

After that, right-click the **Report** node, and in the context menu click **New chart**.

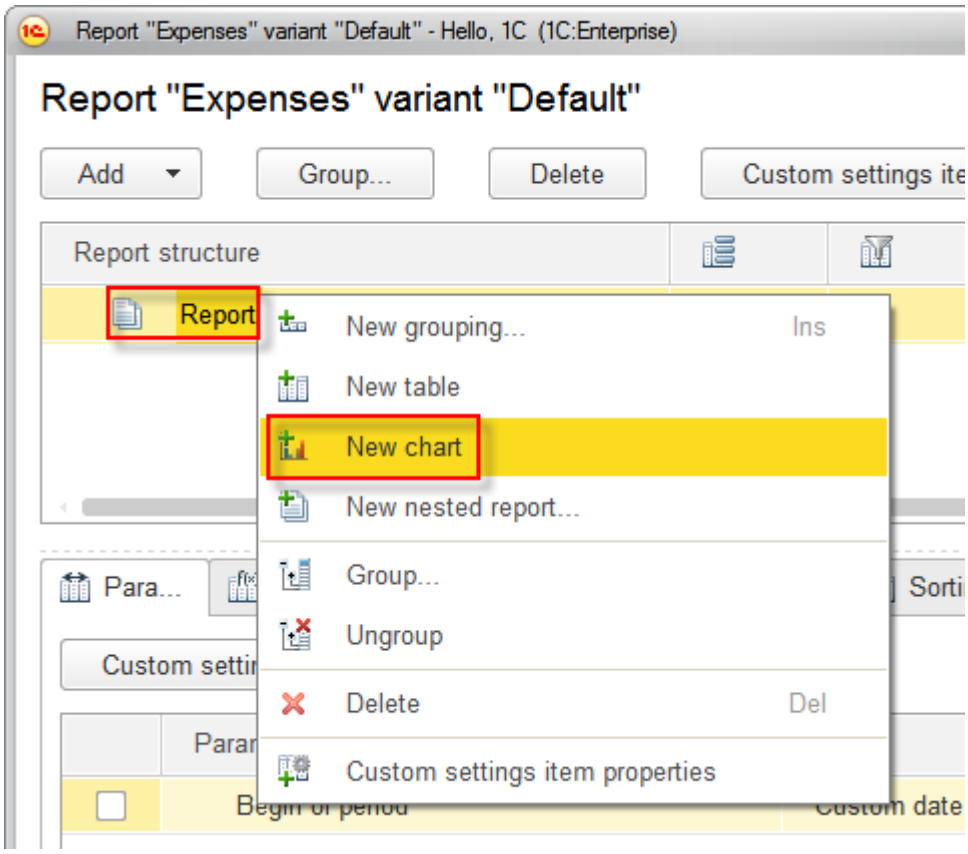

Figure 8-74. Creating a chart

Next, right-click the **Points** node, and in the context menu click **New grouping...** In the opened window, select the **Event** field.

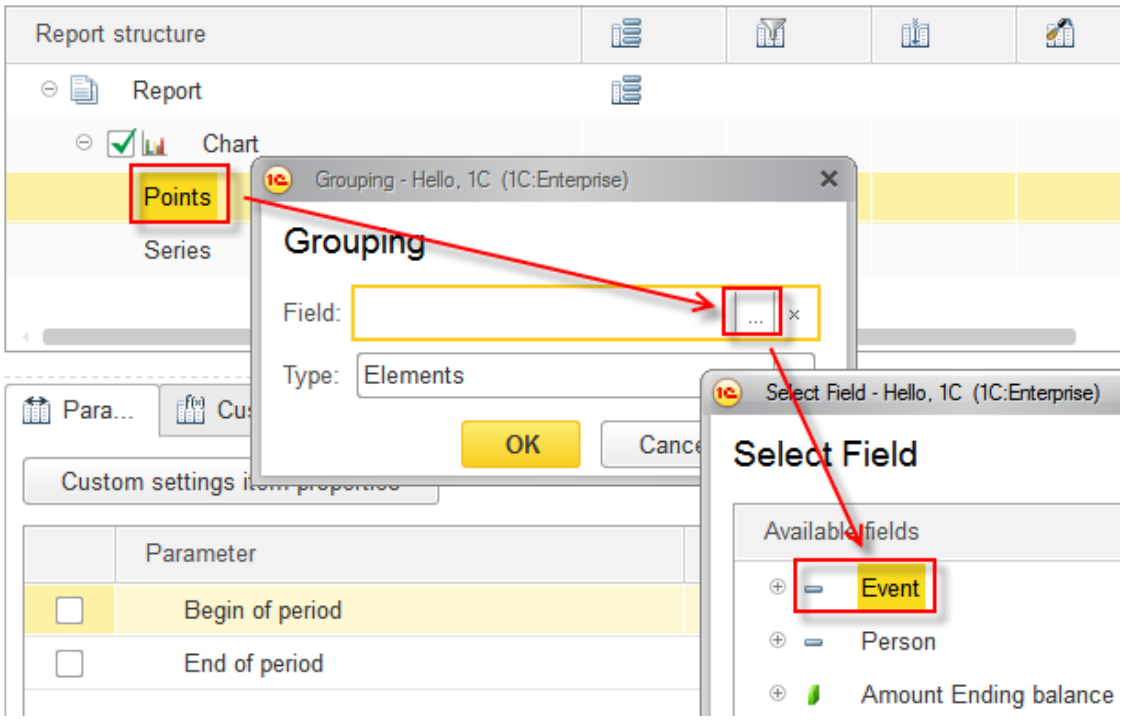

Figure 8-75. The Event grouping field

Then, right-click the **Series** node, and in the context menu click **New grouping...** In the opened window, select the **Person** field.

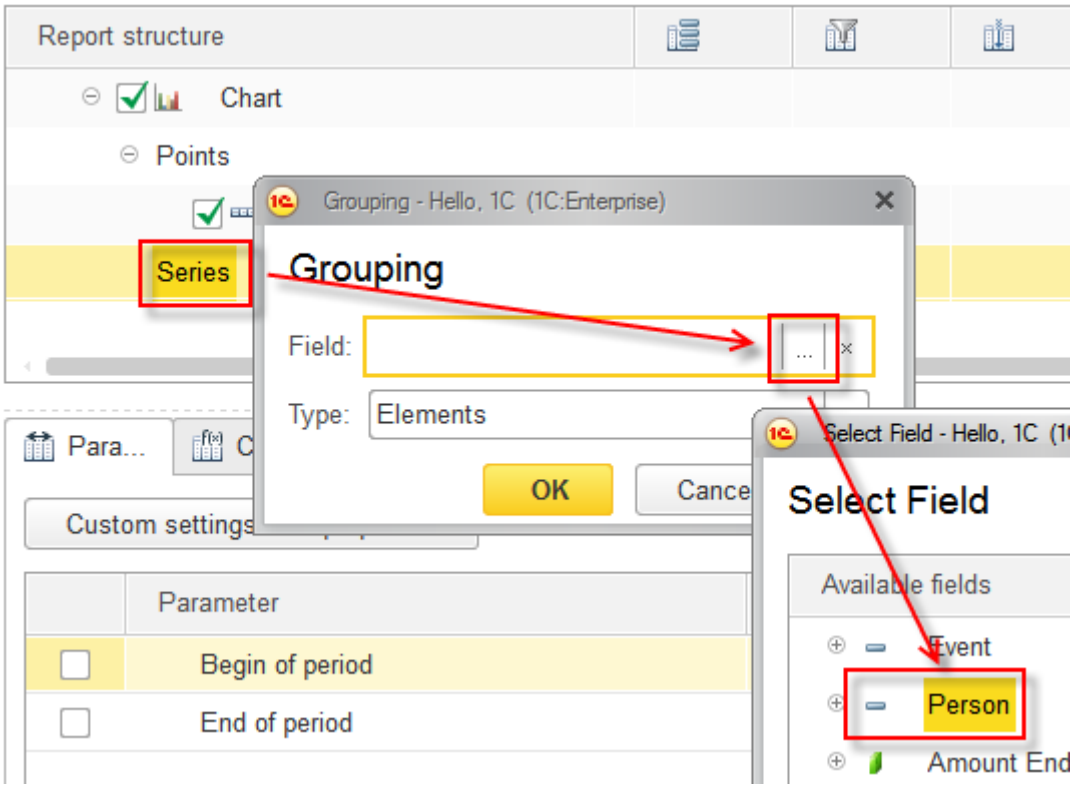

Figure 8-76. The Person grouping field

Now, click the **Report** node, then click the **Fields** tab.

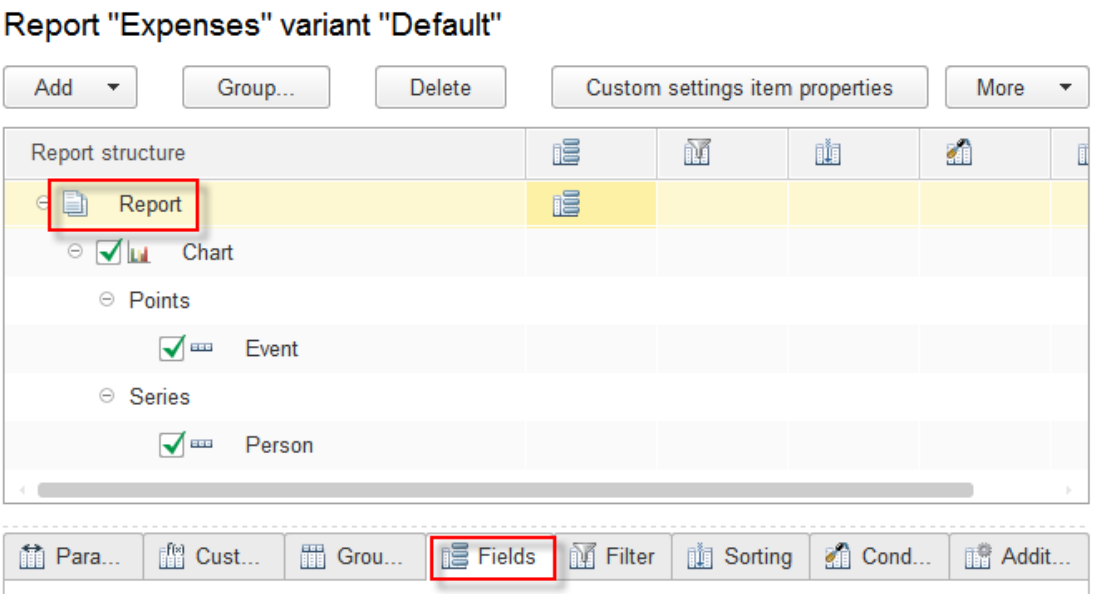

Figure 8-77. Report fields

On the **Fields** tab check that the **Amount Turnover** field is placed right after **Person** and **Event** fields. In addition, check boxes on the left from **Amount Receipt** and **Amount Expense** fields are cleared.

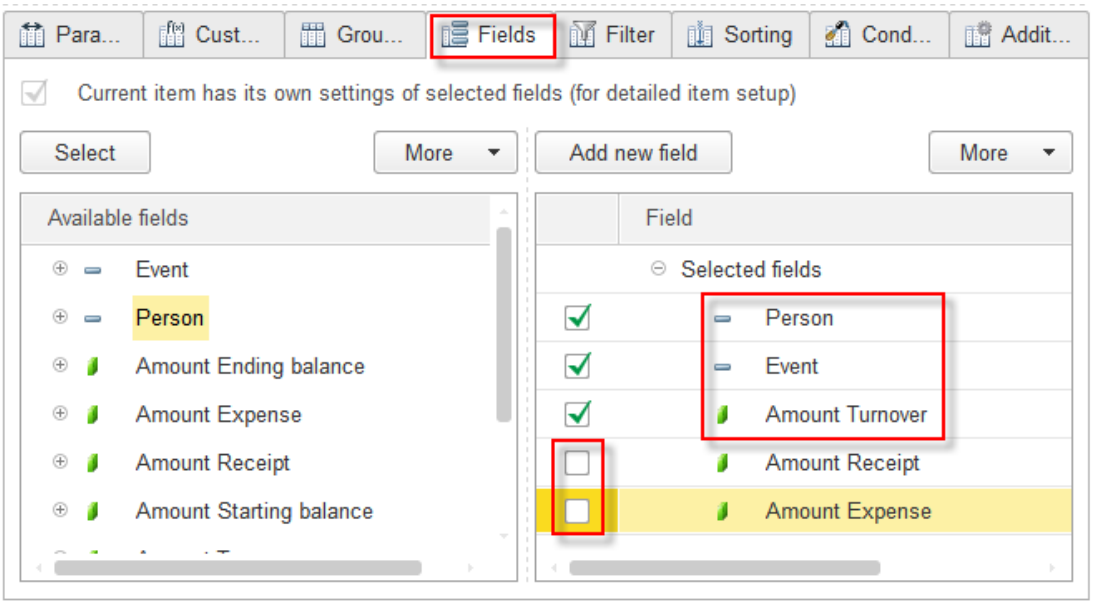

Figure 8-78. Selected fields of the report

Now click the **Additional settings** tab, and for the **Chart type** property select **Bar Graph**.

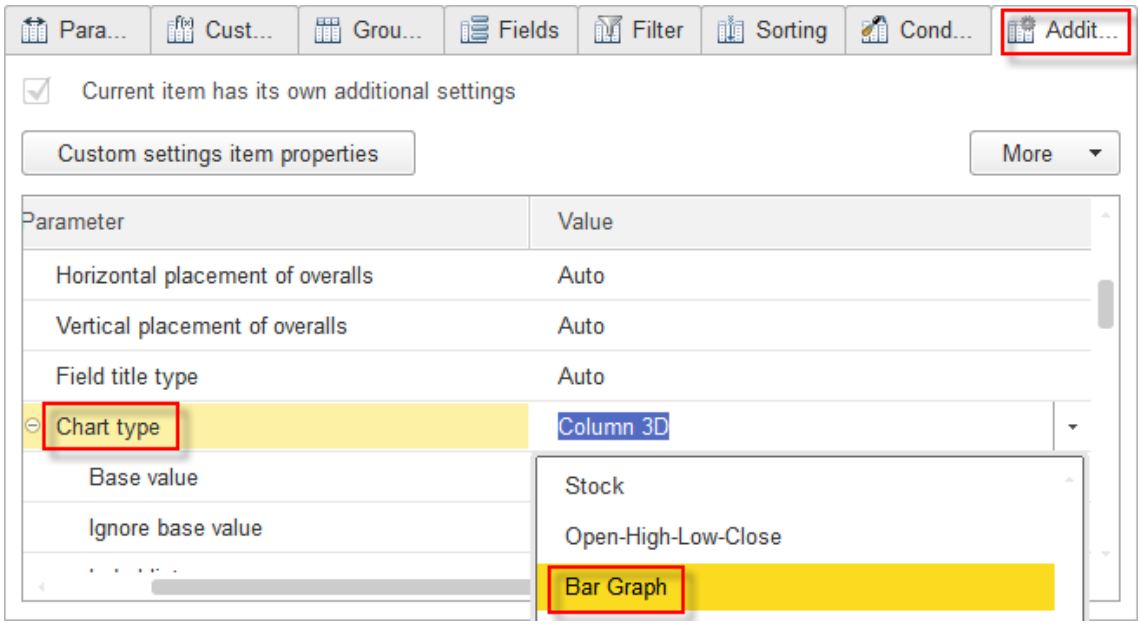

Figure 8-79. Selecting the chart type

Click **Finish editing**, to save and close the report editor

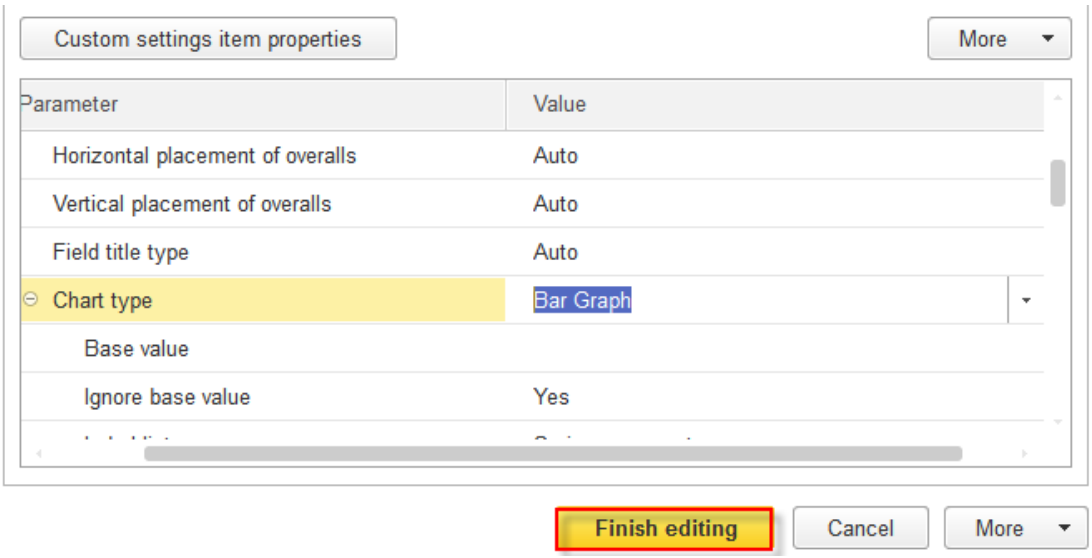

Figure 8-80. Saving the report settings

In the report window, click **Create** and see how dramatically the appearance of the report has changed. Now the user can clearly see what the most money is being spent for.

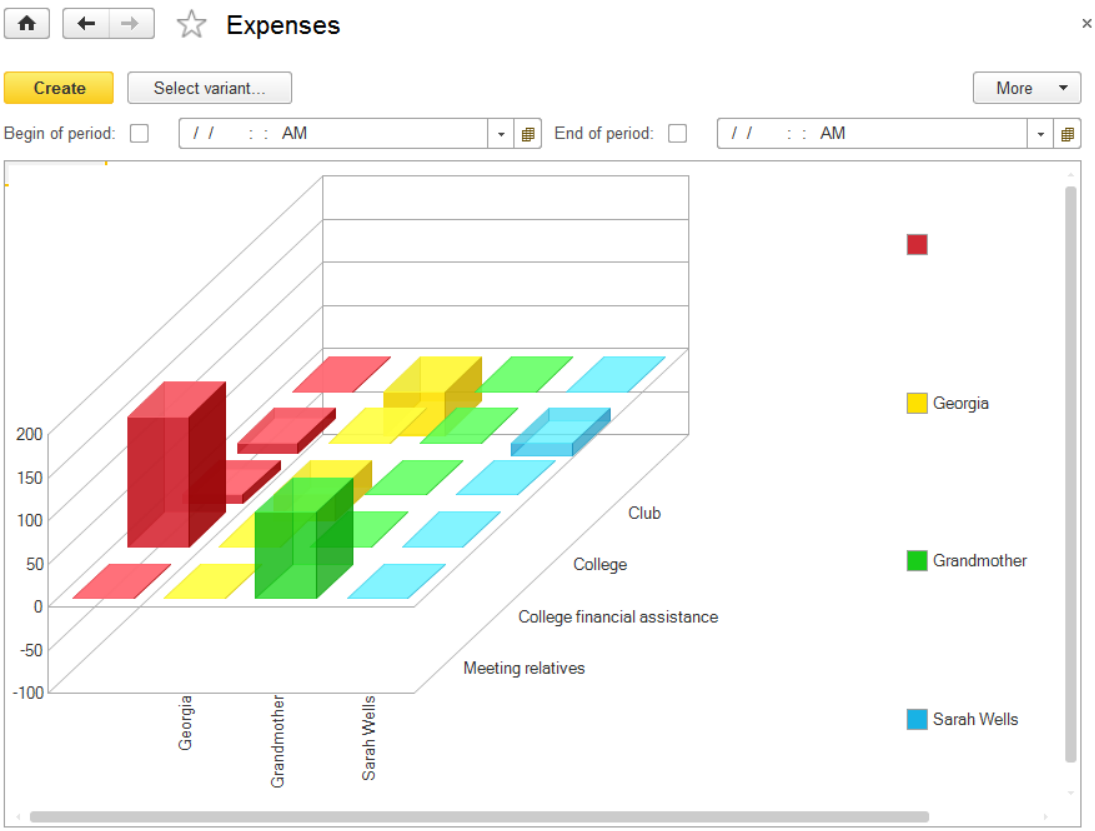

Figure 8-81. The new appearance of the Expenses report

You do not need to change the default settings of the report because the platform provides convenient save and load commands for report variants.

Click **More** and then click **Save variant...**

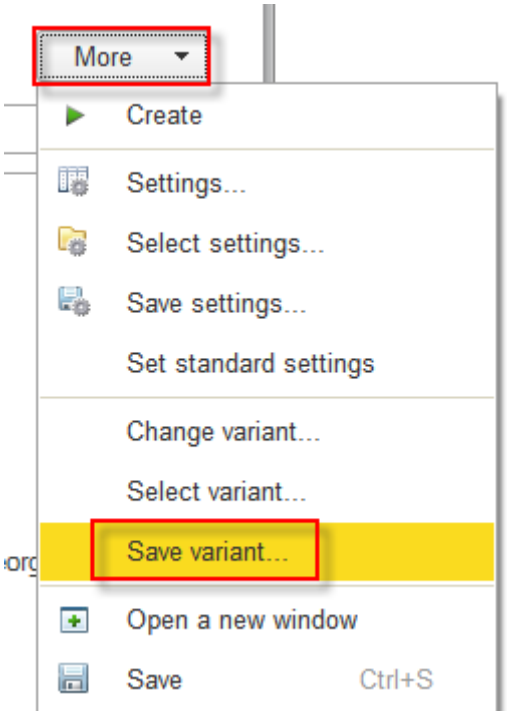

Figure 8-82. Saving the report variant

Input **3D Chart** for the name and click **Save**.

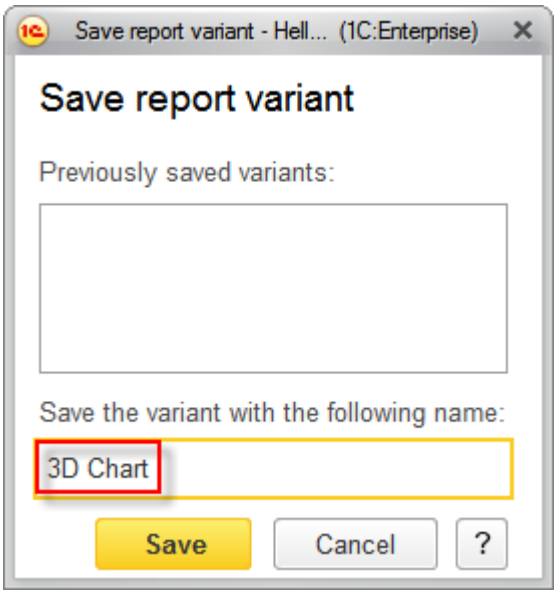

Figure 8-83. Specifying the report variant name

In the future, you will be able to use this variant as many times as it will be necessary. To load the report variant click **Select variant...**

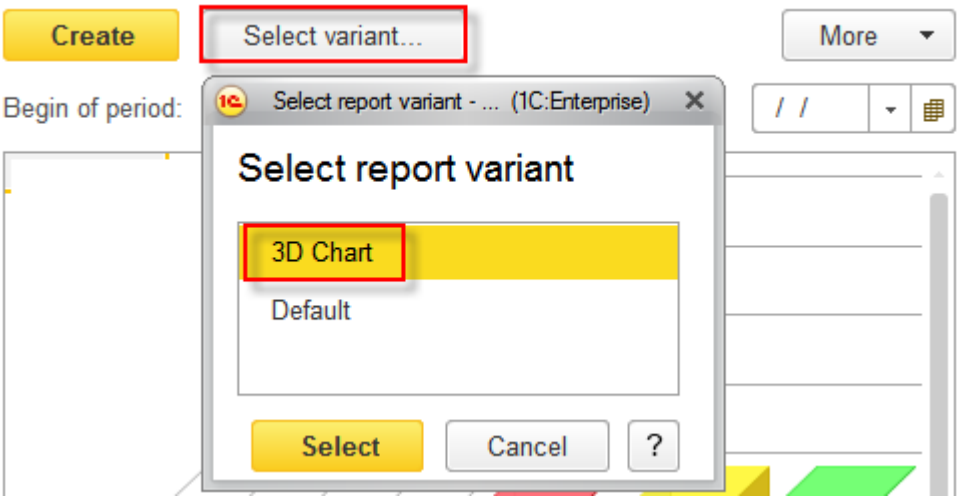

Figure 8-84. Loading the custom report variant

Note that you can always restore the version that was created by the developer in the Designer mode because it is listed as **Default** in the list of report variants.

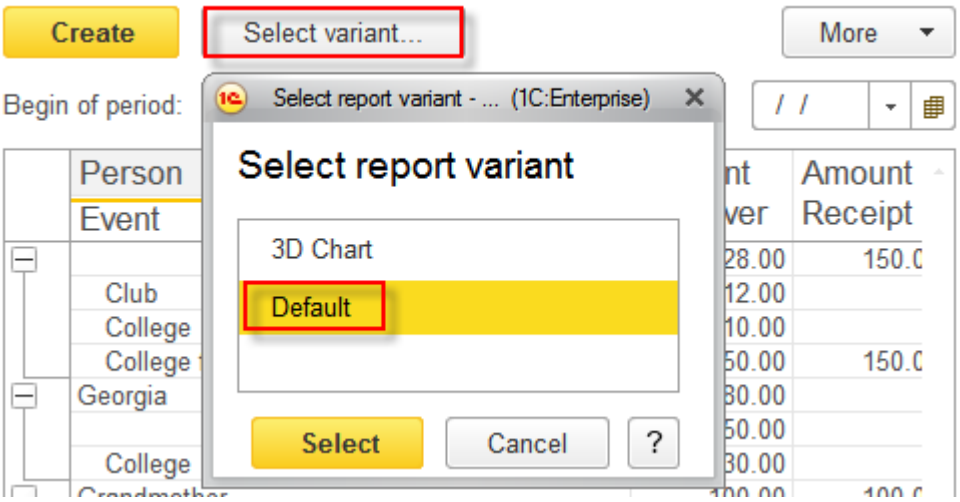

Figure 8-85. Loading the default report variant

Functional options

Another useful part of 1C:Enterprise 8 is the functional options feature. Functional options allow you to group some of applied solution features and then with a single click manage whether the set of features will be used in the specific application or not.

Functional options are advantageous during the applied solution implementation. For example, the CRM that you create in this tutorial include three accounting sections: managing people, events, and finances. If you will decide to share your applied solution with your friends or to enhance its functionally and sell, you will see that not everything that the applied solution is capable of is useful for each user. For example, not everyone will need to keep all events that occur in his or her life. However, maintaining a list of acquaintances and keeping records of money income and outgoings is useful for most users.

Thus, keeping in mind that someone may find events feature not useful, you can segregate it to the functional option and let a user to decide whether to use it or not. If you do so, 1C:Enterprise 8 platform will maintain all references to objects of the functional option in all parts of the applied solution. Moreover, if the user does not need features of the functional option, the platform will automatically disable and hide those objects.

The platform will perform all these changes on its own, so that the developer does not need to write the script for that. The only thing that is required is to set the desired value of the functional option in the 1C:Enterprise mode!

Consider a simple example where the value of the functional option is stored in the constant of the Boolean type. If the value is True, then the functional option is enabled. If the value is **False**, then the functional option is disabled.

Because you are creating the first custom option of the application, it is a good time to create the **Settings** subsystem, and specify the **SettingsSubsystem.png** icon for it.

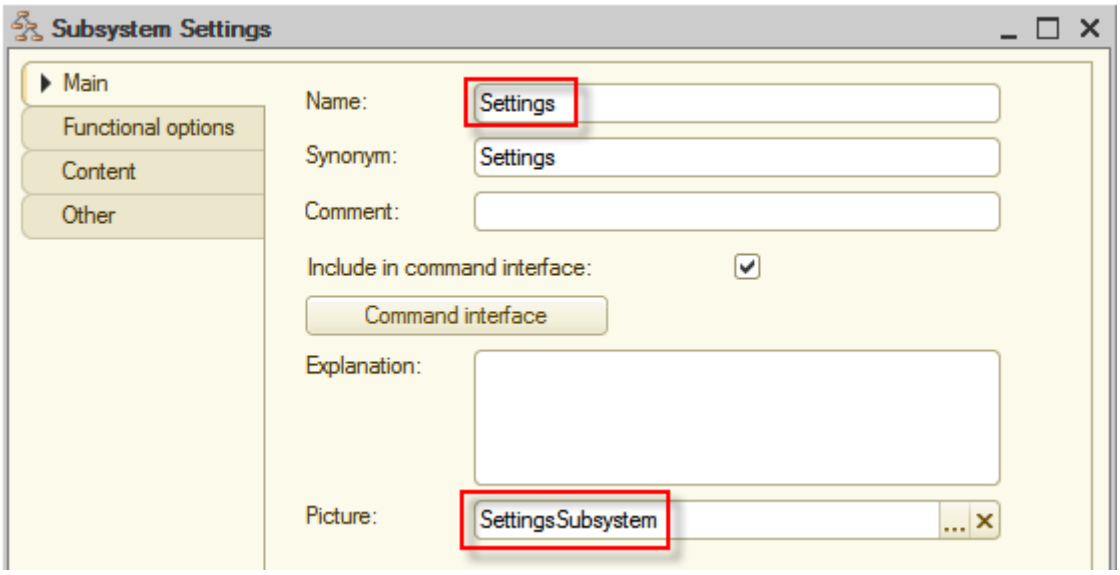

Figure 9-1. Adding the Settings subsystem

At once, place the new subsystem after the operational subsystems. To do this, click the root configuration node, and then in **Properties** click the **Open** link of the **Command interface** property.

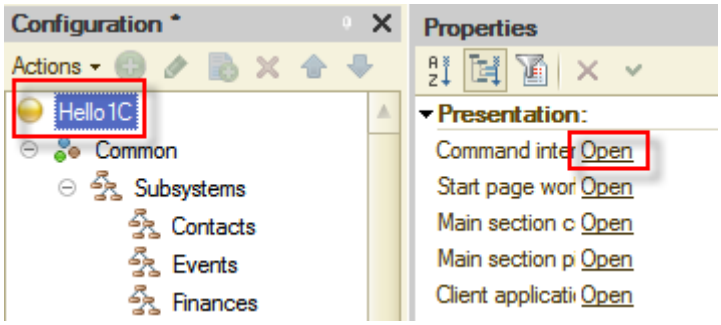

Figure 9-2. Opening the Command interface editor

Move the **Settings** subsystem to the bottom.

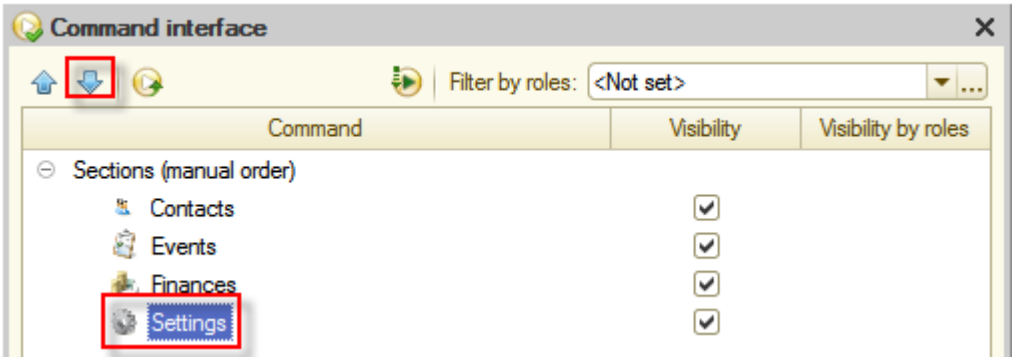

Figure 9-3. Moving the Settings subsystem down

Create a constant.

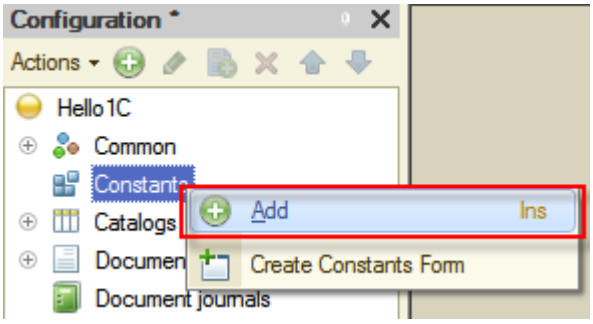

Figure 9-4. Creating a constant

Name it **UseEvents**, and set its **Type** to **Boolean**.

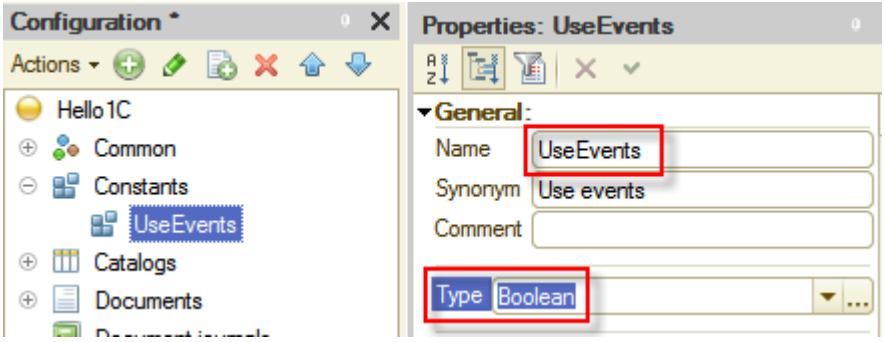

Figure 9-5. Adjusting the Use events constant

After that, right-click the **UseEvents** node in the **Configuration** tree and then click **More** in the context menu.

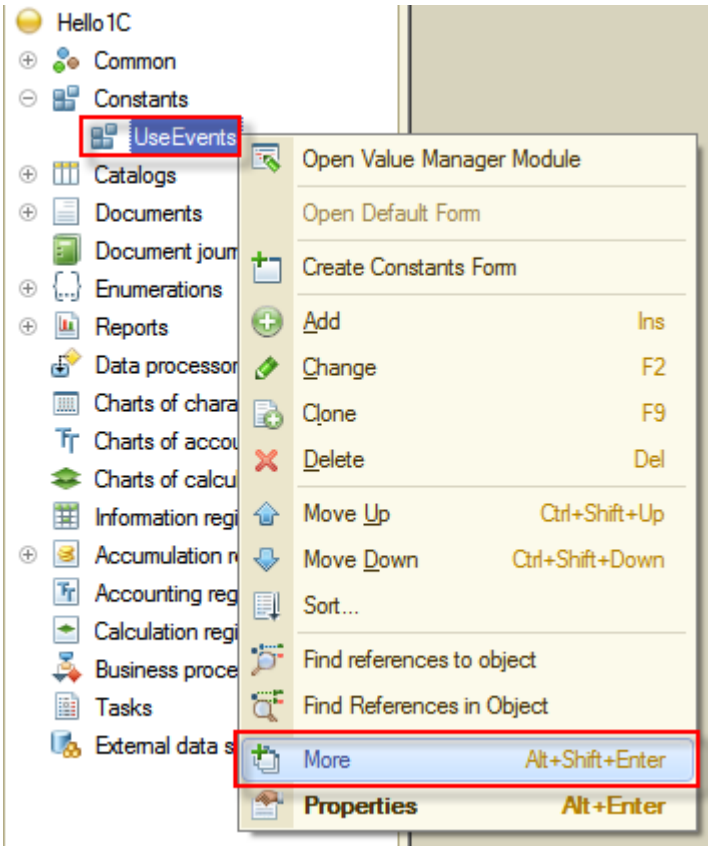

Figure 9-6. Opening the More properties of the UseEvents constant

Then, include the new constant to the **Settings** subsystem.

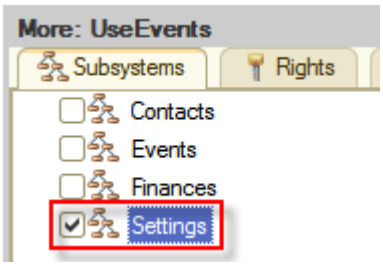

Figure 9-7. Including Use events to the Settings subsystem

Then create the **Use events** functional option and adjust it to keep its value in the **Use events** constant. For that click **Functional options** node in the **Configuration** tree, then click **Add** (Ins) in the command bar of the **Configuration** tree. Then input **UseEvents** as the name of the functional option, and then select **UseEvents** constant in the **Data path** property.

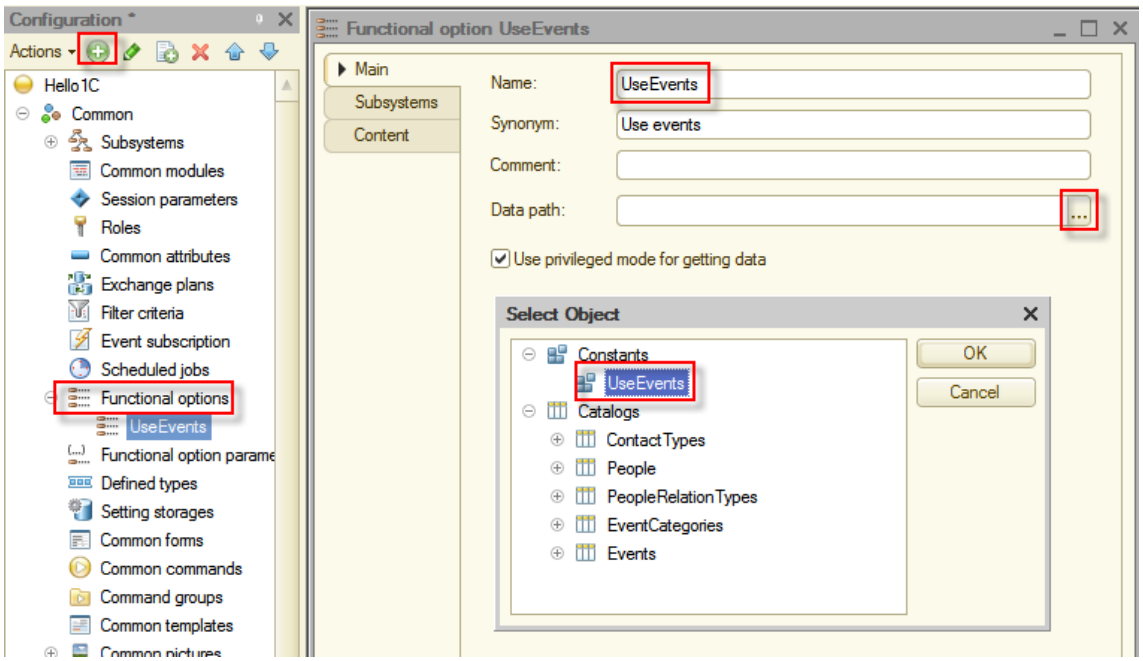

Figure 9-8. Creating the UseEvents functional option

Because of these actions, a command that allows to open the form and change the value of **Use events** functional option in the 1C:Enterprise mode will appear in the **Settings** section.

Now you need to include configuration objects and object properties to the functional option. Ask yourself: "What in your applied solution is related to events?" The answer is:

- the **Events** subsystem;
- the **Events** catalog;
- the used only for events **Event categories** catalog;
- the **Event** attributes of tabular sections of the **Cash receipt** and the **Cash payment** documents;
- the **Event** dimension of the **Financial transactions** accumulation register.

It is very easy to include the mentioned above objects and properties to the **Use events** functional option. To do this, double-click the **Use events** functional option to open it in the editor and then click **Content** tab. After that, in the tree on top set check boxes on the left of configuration objects and object properties. For verification see the panel on bottom that the platform fills with selected objects.

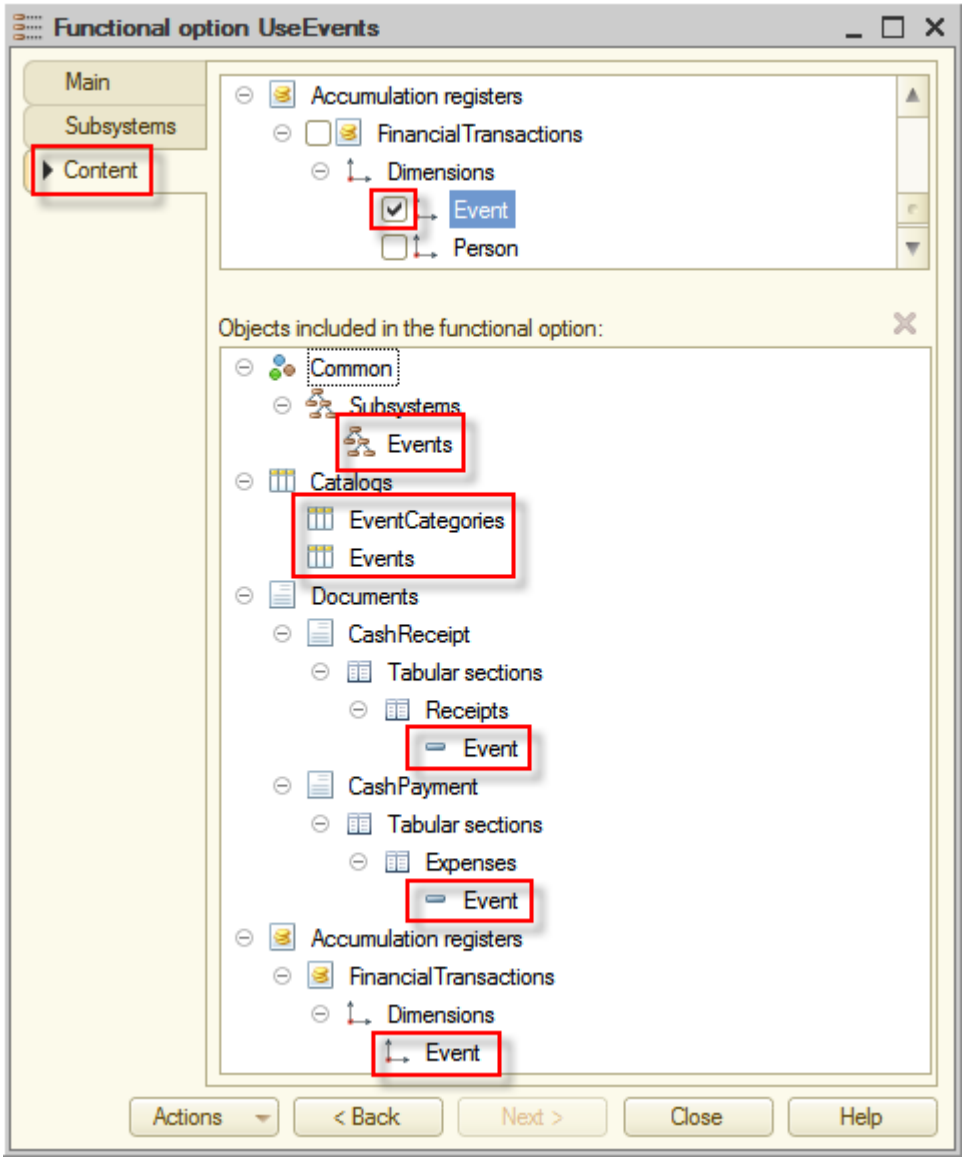

Figure 9-9. Including objects to the functional option

Now, start the application in the 1C:Enterprise mode, click the **Settings** section, click the **Tools** menu, and then click **Use events**.

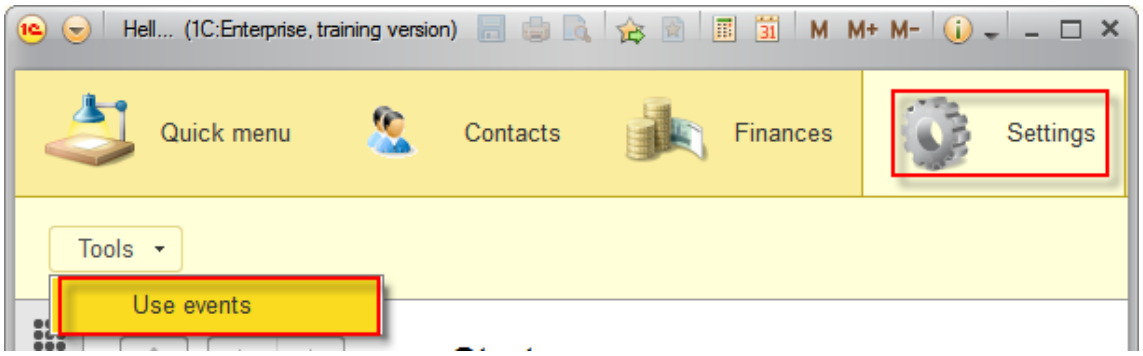

Figure 9-10. Opening the Use events constant form

Enable **Use events** to make sure that the created functional option did not affect anything in the application. For this, select the **Use events** check box, and then click **Save and close**.

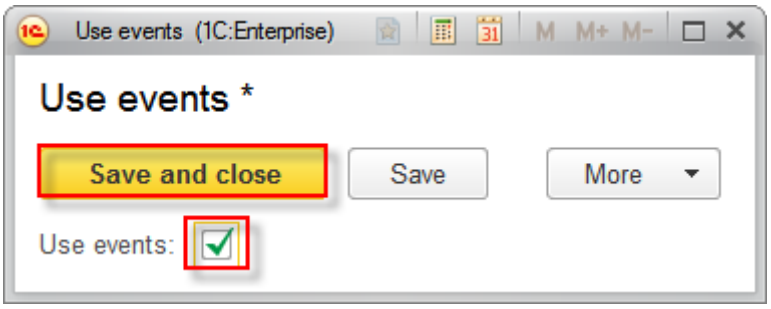

Figure 9-11. Enabling the Use events functional option

Close the main application window and start the application in the 1C:Enterprise mode once again to see that the application works same as it was before you created the functional option.

Now, in the **Settings** section disable **Use events** functional option.

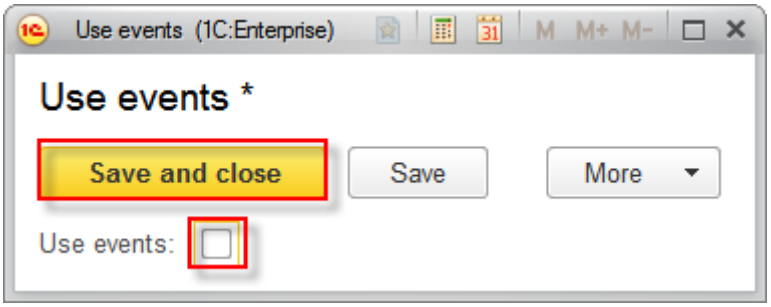

Figure 9-12. Disabling the Use events functional option

After saving changes, restart the application in the 1C:Enterprise mode.

Look at the application carefully.

First, that you might mention is that the **Events** section and various object references related to events are no longer displayed on **Start page**.

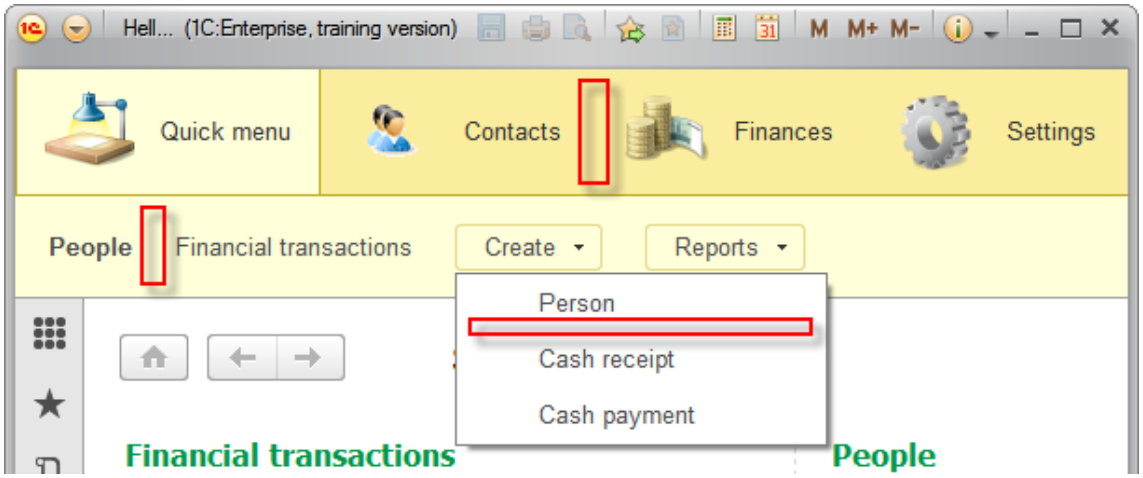

Figure 9-13. The Events section and Events catalog are disappeared

Second is that references to events have disappeared from all forms, including the information displayed in reports.

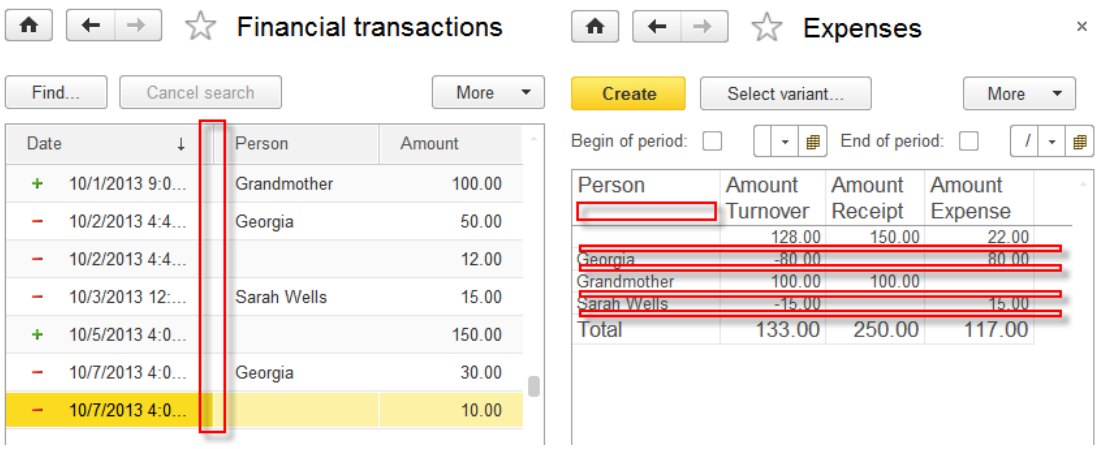

Figure 9-14. The Events column in the list form and Event details of the report are disappeared

You made what was required, the use of events in the applied solution is now optional and depends on settings of a specific application.

Cross-platform design

Now it is time to explain on what operating systems and devices you can run your application.

Linux

You can use the application that you have developed in this tutorial without any modification both on computers with Microsoft Windows and Linux operating systems.

1C:Enterprise 8.3 platform have introduced following analogs of already available for Microsoft Windows client applications:

- **Thin client** allows users to run applications in the 1C:Enterprise mode;
- **Thick client** allows users to run applications in the 1C:Enterprise and the Designer modes.

Client applications support both file and client/server modes. They are available both for x86 and x86-64 architectures.

This means that now not only the 1C:Enterprise mode users can work on Linux, but also developers and administrators.

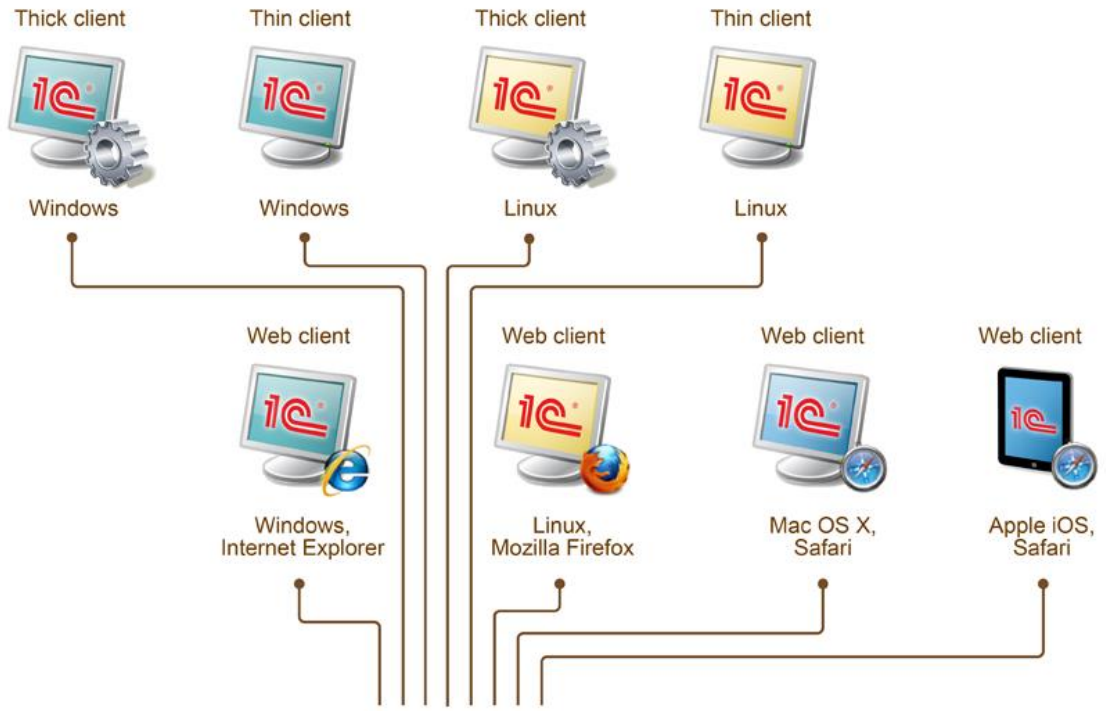

Figure 10-1. Cross-platform design

The full list of supported operating systems you can read in System requirements.

In Linux, the application will look and feel the same as it does on Microsoft Windows.

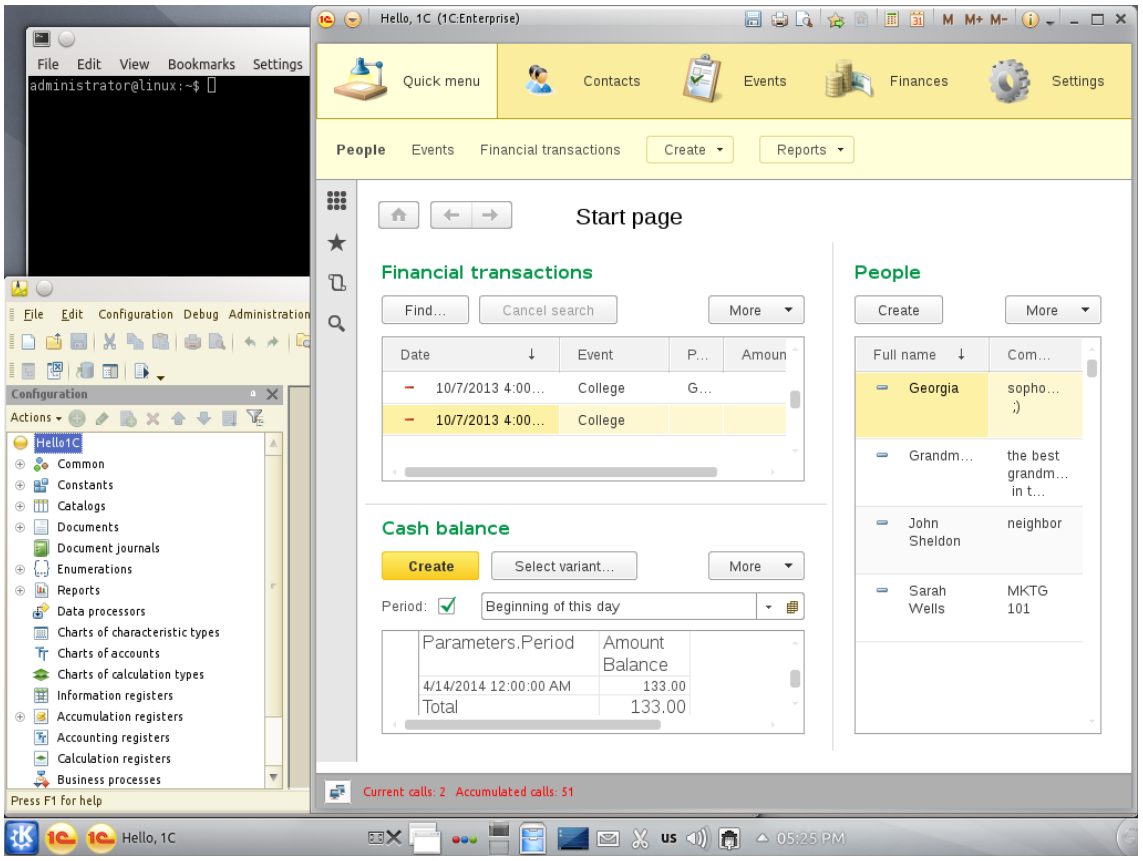

Figure 10.2 1C:Enterprise on Linux

Web client

It is a common situation when you are not at your workplace, but need some data urgently. In this case, an access through the Internet using a web browser might save you.

It is no doubt that you can develop a web interface for any system. To do this you need to develop, debug and deploy it as well as maintain compatibility with different web browsers. 1C:Enterprise platform allows developers to save efforts on development of the web interface.

The platform allows you almost in a single click publish on the web server entire the application that you have developed in this tutorial. After that, users will be able to access the application using regular web browsers. At the time of writing this book there are four supported web browsers, they are: Microsoft Internet Explorer, Mozilla Firefox, Google Chrome, and Apple Safari.

This means that users do not need to install any of 1C:Enterprise applications or licenses, the only requirements are the connection to the Internet and one of supported web browsers. At the same time, you will need to have a server where 1C:Enterprise platform and a web server are installed. At the time of writing this book there are two supported web servers, they are: Microsoft IIS and Apache.

To install the web server, click **Start**, then click **Control panel**, and then click **Uninstall a program**. On the navigation panel click **Turn Windows features on or off**. In **Windows Features**, select **Internet Information**

Services. The default set of features is enough. Click **OK** to setup and close this window.

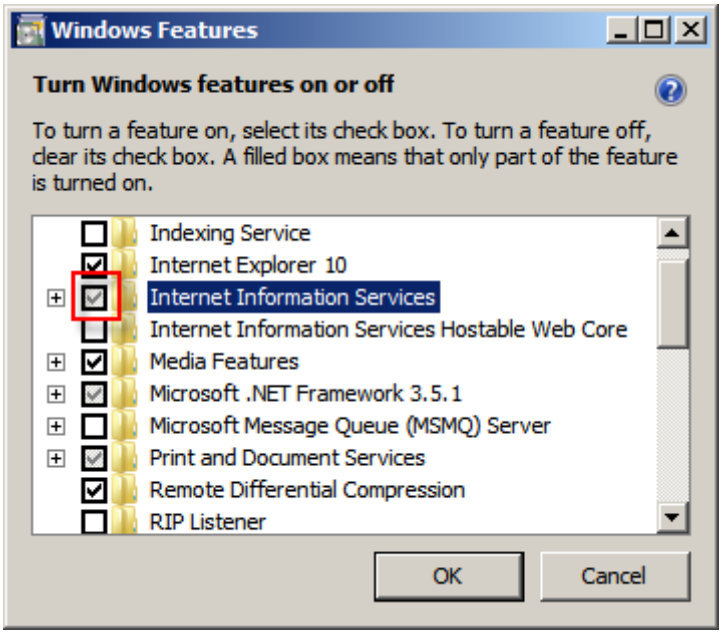

Figure 11-1. Installing the Internet Information Services web server

To publish web client you need to have the **Web server extension modules** component of 1C:Enterprise platform installed. To install the component, click **Start**, then click **Control panel**, then click **Uninstall a program**, in the list of applications find **1C:Enterprise 8 (training version) (<*version number*>)** and select it. The version must match the version that you use for publishing the application. Then click **Change** on top of the list. In the installation wizard select the **Modify** option, and click **Next**.

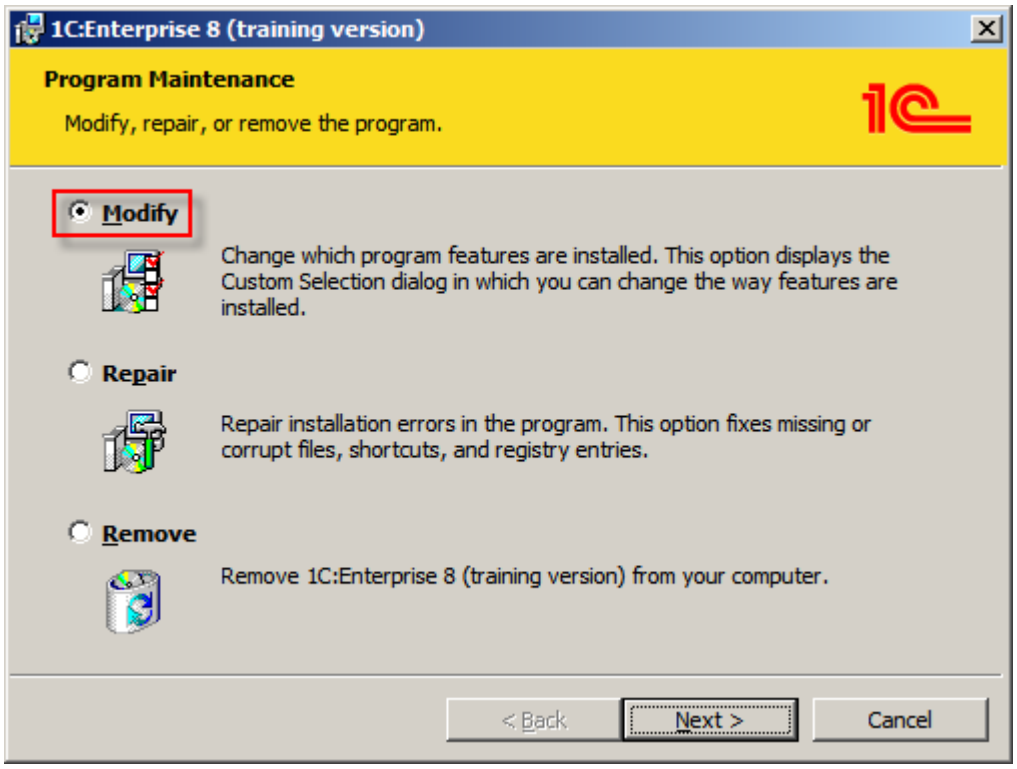

Figure 11-2. Modifying the 1C:Enterprise installation

On the next page select **This feature, and all subfeatures, will be installed on local hard drive** for **Web server extension modules**. After that, follow the wizard confirming default settings.

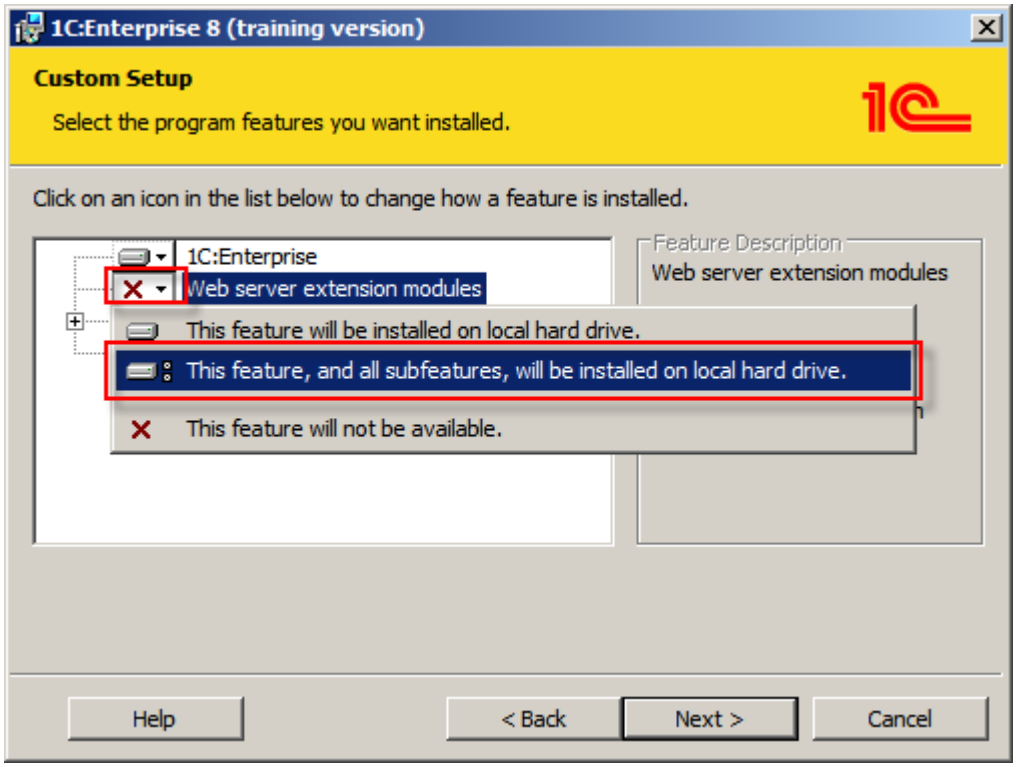

Figure 11-3. Installing the Web server extension modules component

Enabling an access to the application through the Internet is simple. Open the application in the Designer mode. In **Main menu** click **Administration** and then click **Publishing on web-server...**

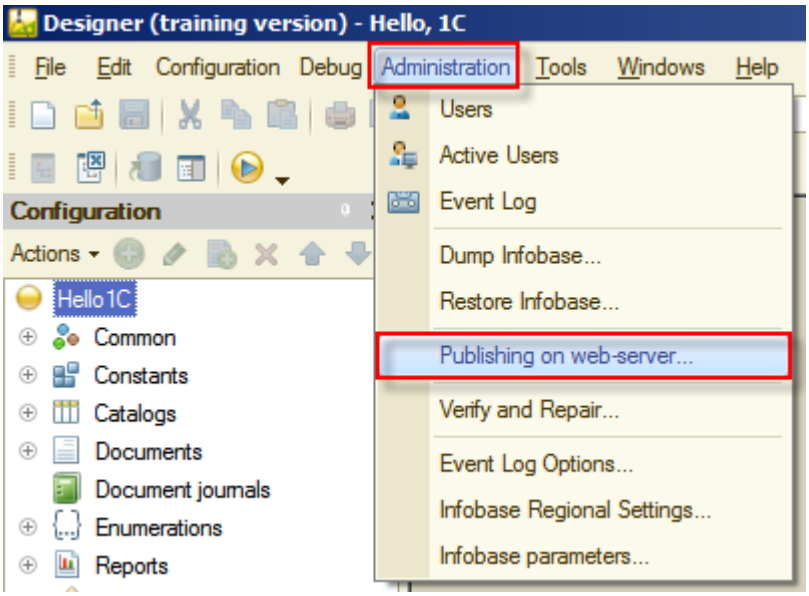

Figure 11-4. Opening the Publish to the web server wizard

In the **Publish to web server** wizard in **Name** field input **hello_1c** and in **Directory** field see that the last folder is changed to **hello_1c**. Then, click **Publish**.

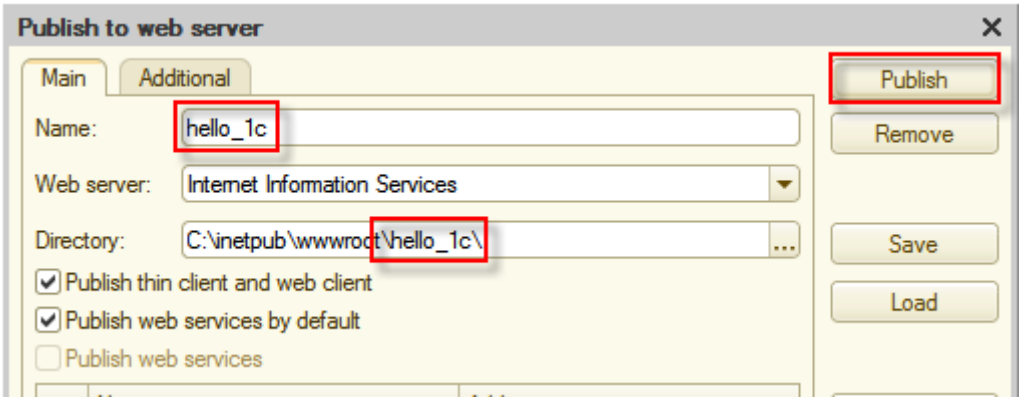

Figure 11-5. Publishing the application on the web server

The platform will ask for confirmations during publishing. Agree with them.

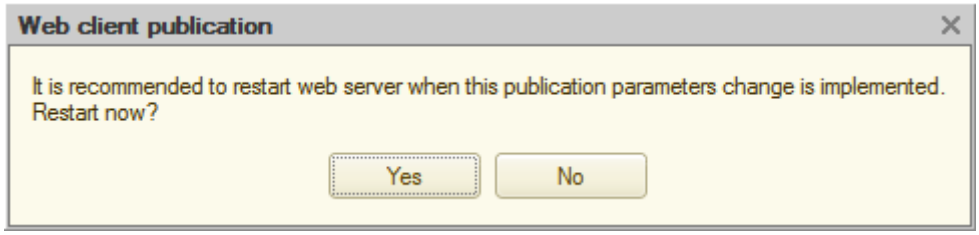

Figure 11-6. An example of the publication confirmation dialog.

After the application is published, you can access it from any place around the world. To connect to the application you will need to know only its URL.

Notice: Details of configuring web servers are skipped in this book. In short, you will need to let **IIS_IUSRS** user on the server an access to 1C:Enterprise installation **bin** folder, to the folder that you have specified in **Directory** field of the **Publish to web server** wizard, and to the folder where your infobase is placed. Then, you need to enable **Anonymous Authentication** in Authentication settings of the **hello_1c** IIS application and for 64-bit Windows you need to set to **True** value of **Enable 32-Bit Applications** in advanced settings of the **DefaultAppPool** application pool.

In the tutorial you run it on your local computer, thus the URL will be local only. To log on type http://localhost/hello_1c in the address bar of your web browser and you will see the 1C:Enterprise interface, with which you are already familiar.

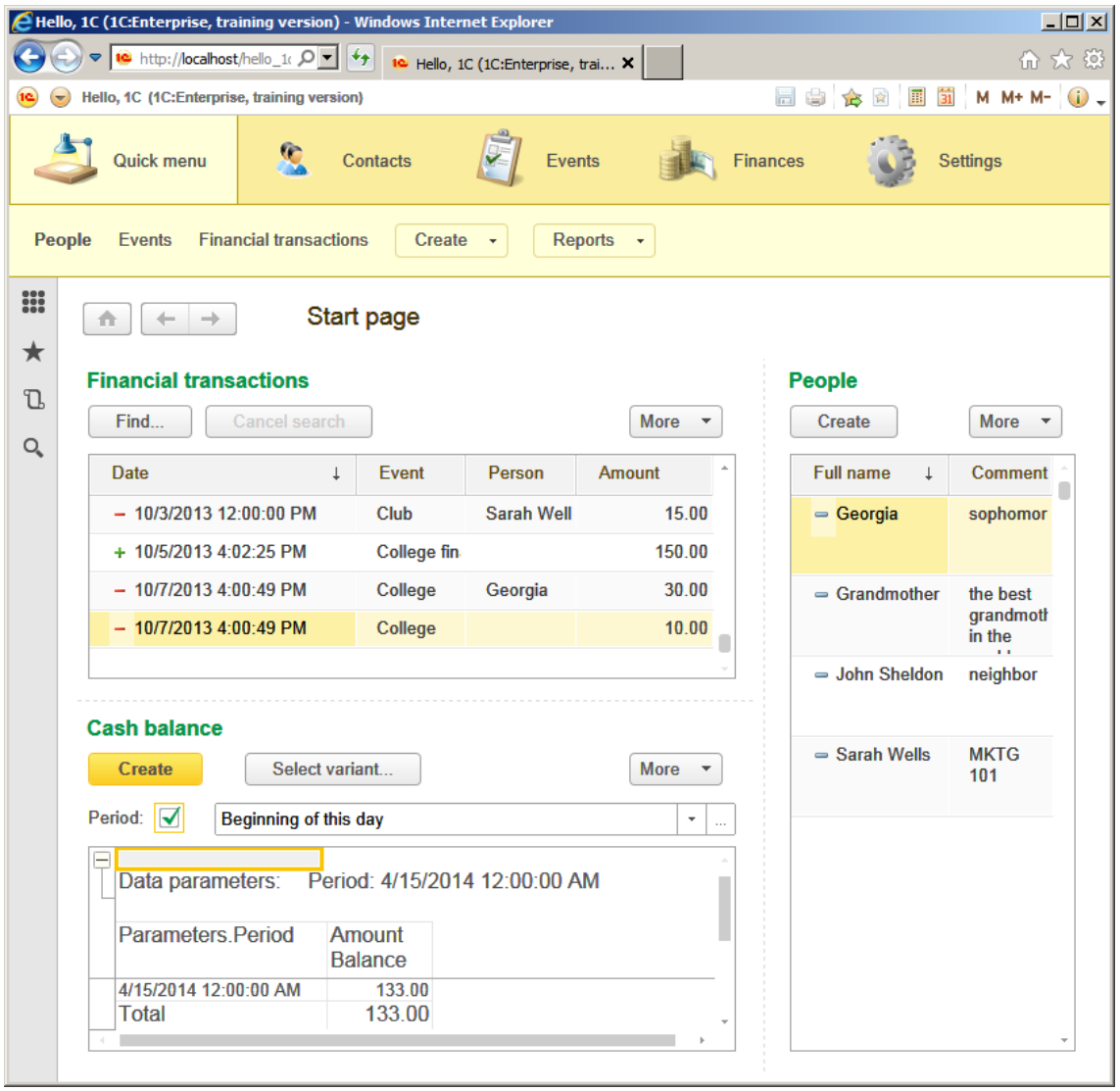

Figure 11-7. Web client in Microsoft Internet Explorer

As it is said above, you can use your favorite web browser to work with your application.

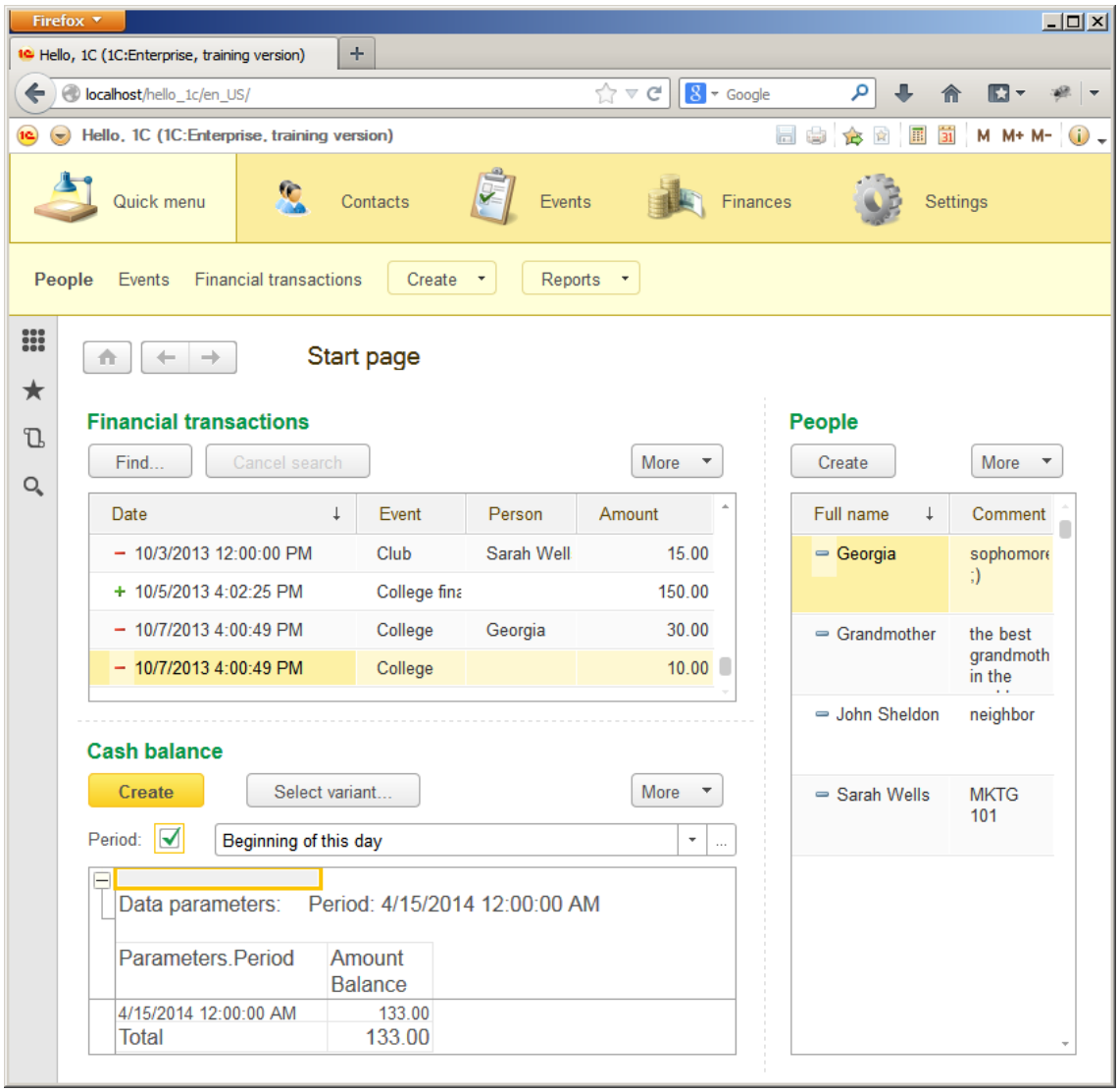

Figure 11-8. Web client in Mozilla Firefox

Notice: The 1C:Enterprise (training version) has a set of limitations, thus you can not protect it with the password as users feature is disabled for the training version. In addition, to log on using the web client you need to quit the Designer mode first as there can be only one concurrent user in the training version. The actual at the time of writing this book limitations are listed in 1C:Enterprise 8 (training version) chapter on page 149.

You can see live demo on 1C:Developer Network in Applications section.

Mobile platform

One more interesting thing is that the application that you have developed with almost no efforts can be started on iOS or Android mobile devices. This feature is provided by the mobile platform that is a part of 1C:Enterprise 8.3 platform.

The look and feel of applications for Android and iOS devices can be a slight different.

Notice: Systematic instructions on how to create a mobile application for end users you can find in 1C mobile application rapid application

development tutorial. In this tutorial, you are going to start your application in the mobile platform for developers.

Details on features of applications based on 1C:Enterprise platform for mobile devices you can see in Chapter 25. Developing solutions for the mobile platform of 1C:Enterprise 8.3. Developer guide.

1C:Enterprise 8 mobile platform is a set of tools that let you to create applications that work on Android or iOS mobile devices. Those devices are mostly smartphones and tablet PCs.

The 1C:Enterprise mobile application that you install on the mobile device is a combination of the mobile platform and the infobase. The mobile infobase is similar to the file infobase. It consists of a database that stores data in a file and a mobile applied solution that is a program, which can be started on a mobile device and manage data in the database.

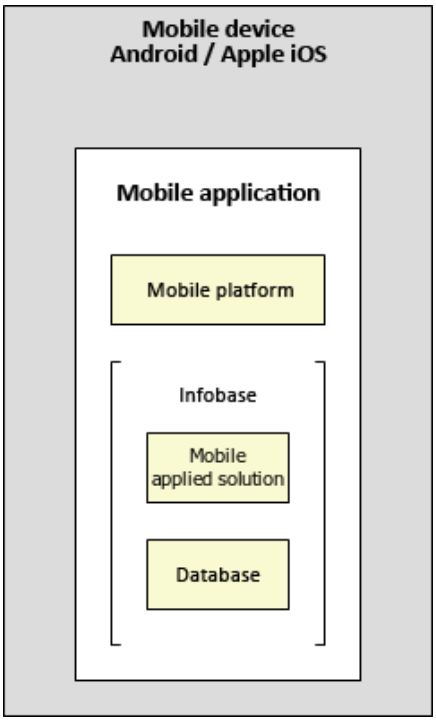

Figure 12-1. 1C:Enterprise 8 mobile platform

Thanks to the mobile platform you can right now, without using any third-party IDE, install, start, and see how works your application on the mobile device. As an example, you will do this for the Android device.

For this, you will need following:

- Make your applied solution compatible with the mobile platform. You will have to do a few changes for that.
- The mobile platform for developers installed on an Android mobile device.
- The web server running on the computer where you develop the application, and accessible by IP from your mobile device.

Start the application in the Designer mode and verify it for capability of being started on mobile devices. For this, in **Main menu** click **Configuration**, and then click **Check configuration...**

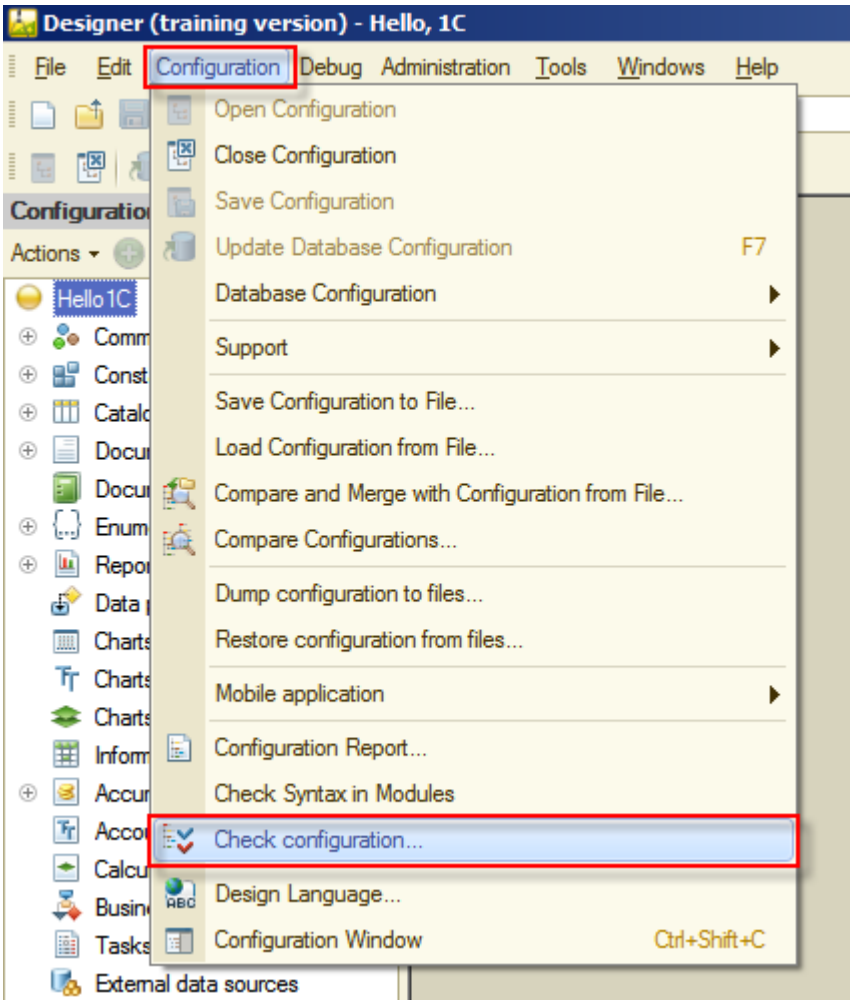

Figure 12-2. Open Configuration check

You can set the name for the current set of configuration verification options to make it easier to repeat this verification later. Let it be **Mobile**. Select following check boxes:

- **Verifying configuration integrity and consistency**;
- **Find unresolved references**;
- **Mobile application - client**;
- **Mobile application - server**;
- **Search for unsupported functionality**.

Click **OK**.

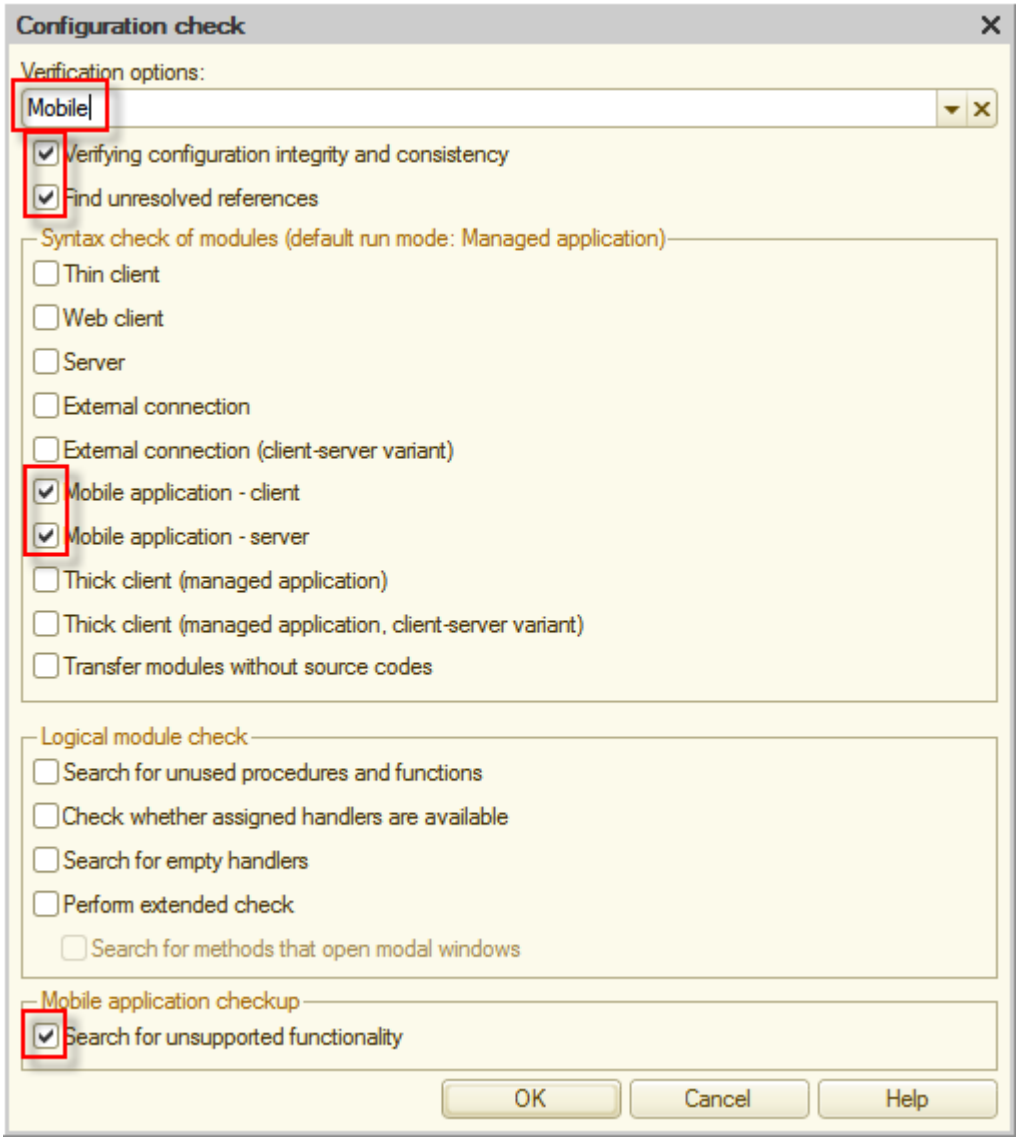

Figure 12-3. The Configuration check window

The configuration will be verified for compatibility with the mobile platform. The result of this verification will be 11 errors.

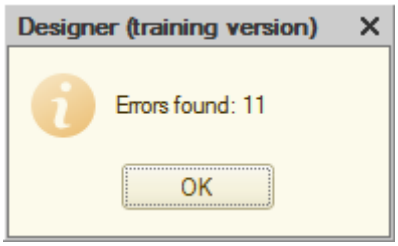

Figure 12-4. The result of verification

Those errors should not make you worry. Here they are:

```
Messages
Configuration.Hello1C : Mobile application does not support multi-form desktop.
Subsystem.Contacts : Metadata class not supported by mobile application platform.
Subsystem.Events : Metadata class not supported by mobile application platform.
Subsystem.Finances : Metadata class not supported by mobile application platform.
Subsystem.Settings : Metadata class not supported by mobile application platform.
FunctionalOption.UseEvents : Metadata class not supported by mobile application platform.
Report.Expenses : Metadata class not supported by mobile application platform.
Report.CashBalance : Metadata class not supported by mobile application platform.
Report.CashBalance.Form.ReportForm : Metadata class not supported by mobile application platform.
Report : Type not supported by mobile application platform.
Report.DailyChart : Metadata class not supported by mobile application platform.
```

Figure 12-5. The list of errors

If you look carefully, you might mention that there are several unsupported by the mobile platform features used in your application. This happened because until this moment you did not take care of compatibility with mobile devices and used all features that you needed.

Now, as a simplest way, you will delete the use unsupported by the mobile platform features and configure **Mobile desktop** that will be compatible with the mobile platform.

Notice: To backup the current application, you can in the Designer mode in **Main menu** click **Administration**, and then click **Dump infobase...** In the opened dialog select a file name and click **Save**. To restore the application, in **Main menu** click **Administration**, and then click **Restore infobase...** In the opened dialog select the previously saved DT file, and click **Open**.

So, delete all subsystems, functional options, and reports.

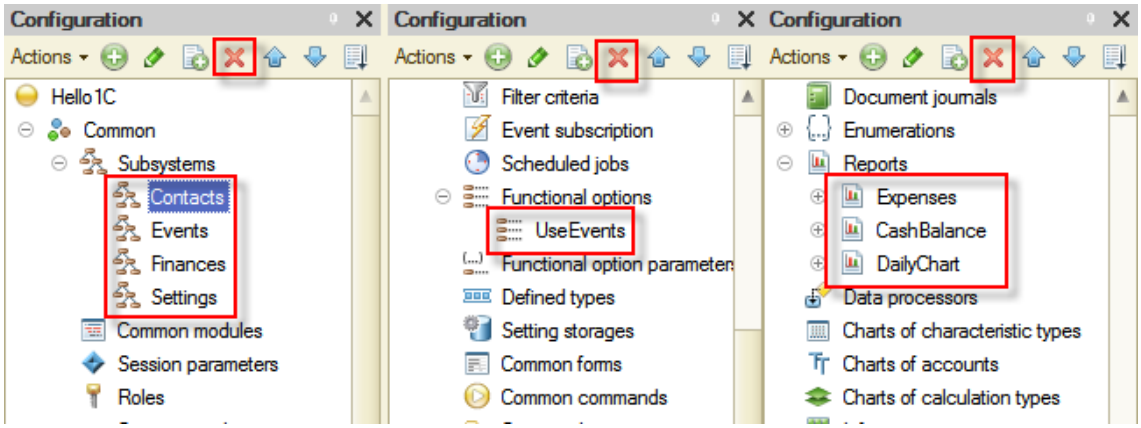

Figure 12-6. Deleting unsupported metadata objects

In **Properties** of the root of the **Configuration** tree, click **Select** ..., and in **Use purposes** list select **Mobile device** check box, and clear **Personal computer** check box. Then click **OK**.

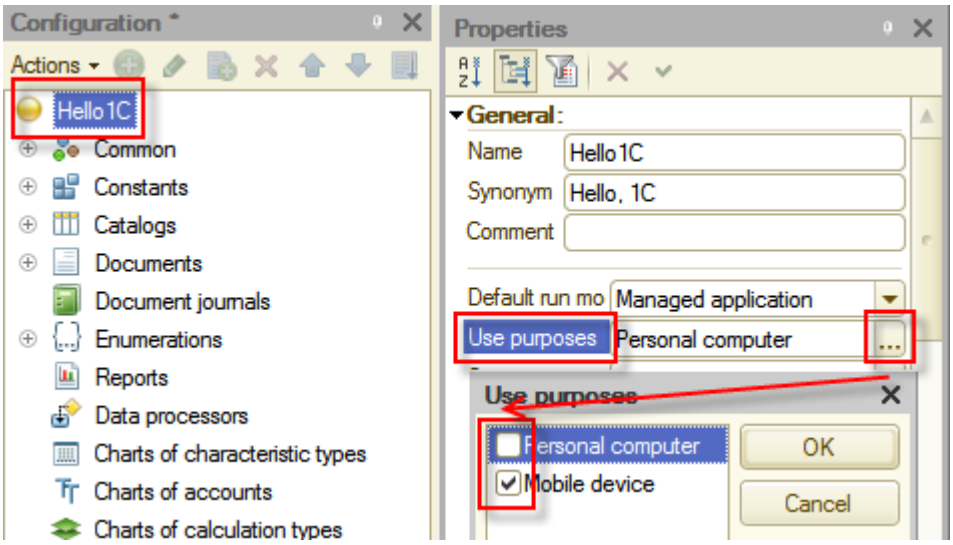

Figure 12-7. Adjusting Use purposes

Notice: Unsupported by the mobile platform application objects are disabled now.

The **Mobile desktop** of the mobile application is not configured yet. After it will be done, the application will be compatible for the mobile platform.

Create a common form.

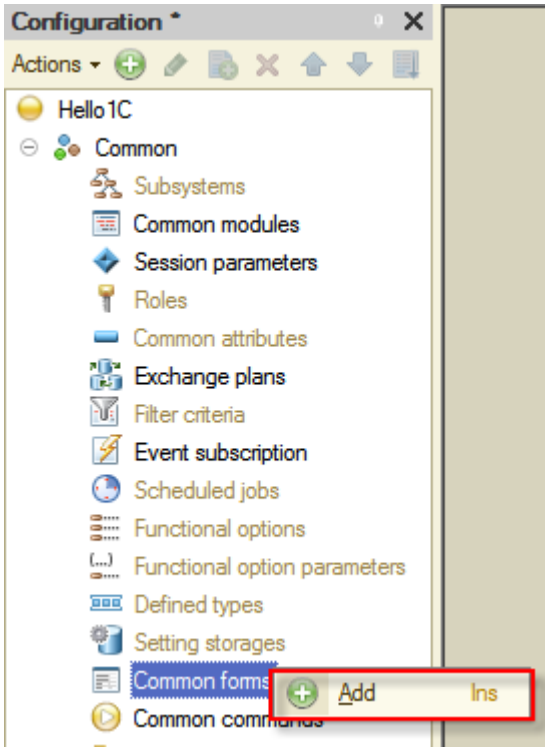

Figure 12-8. Creating a common form

Name it **MobileDesktop** and set **Synonym** to **Hello, 1C**. The form type should stay **Generic form** as for default. After that, click **Finish**.

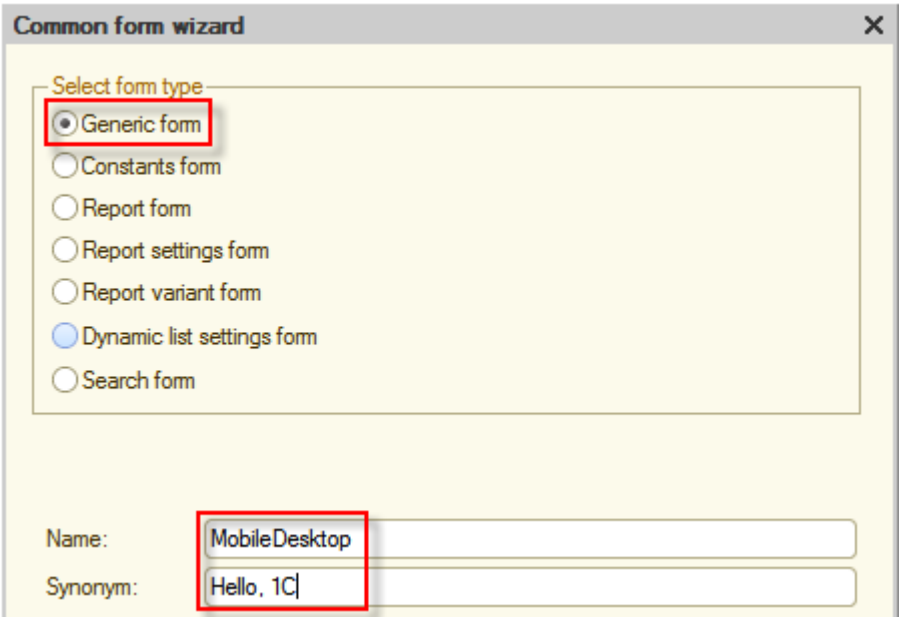

Figure 12-9. Adjusting the common form

It seems reasonable that users would like to see the most often used **People** and **Events** lists on the **Mobile desktop**. To provide this information to users, create two attributes of the **DynamicList** type. The first one will be **People**. To add the form attribute, in the top right panel of the form editor click **Add attribute** (Ins). In **Properties** of the new attribute set **Name** value to **People**, and then click **Select** ... of the **Type** property, then select **DynamicList**.

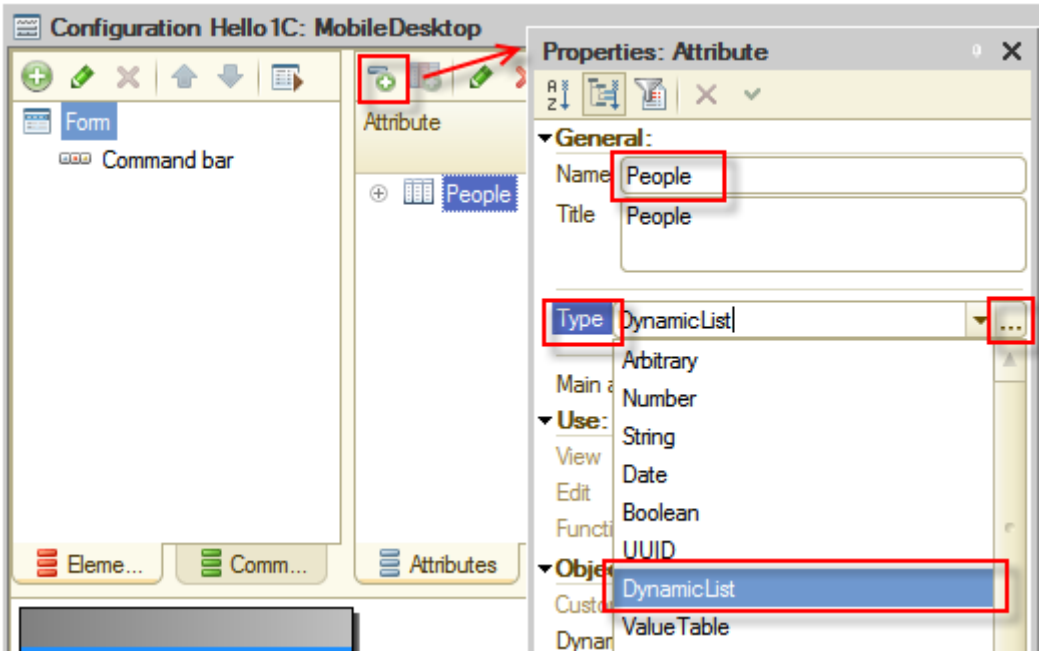

Figure 12-10. Adding the People attribute of the common form

For the **MainTable** property, click **Select** ..., then in **Select table** select the **People** catalog, and then click **OK**.

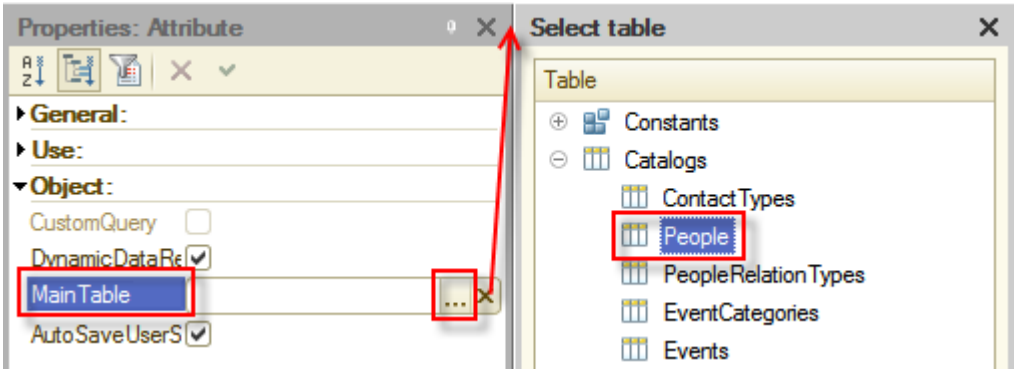

Figure 12-11. Selecting the main table of the dynamic list

In the same way, add the **Events** attribute. With the only difference, the **MainTable** property will be the **Events** catalog.

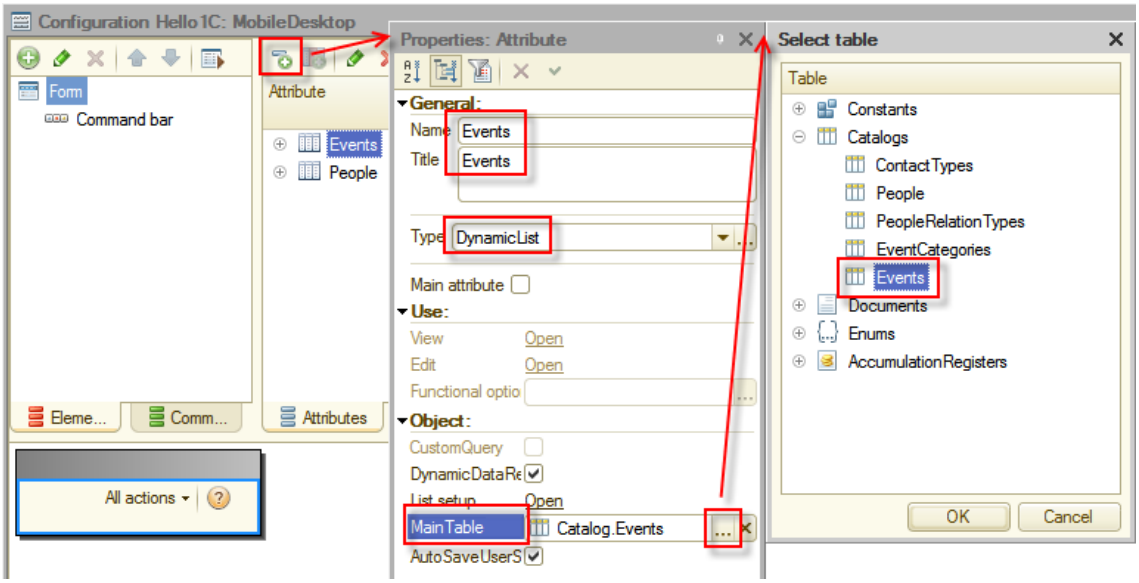

Figure 12-12. Adding the Events form attribute

Now, drag both attributes to the **Form** node on the top left panel of the form editor. You can select more than one form attribute by holding the **Ctrl** key and clicking form attributes, and then drag them all at a time to the destination.

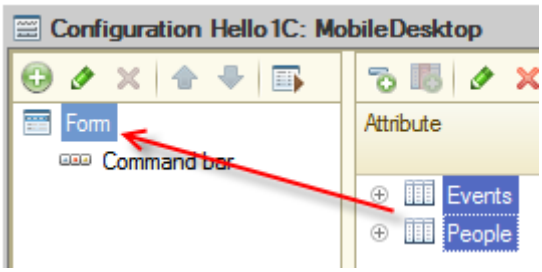

Figure 12-13. Placing form attributes on the form

On the opened confirmation dialog, click **Yes** to agree that columns of both tables will be added to the form automatically.

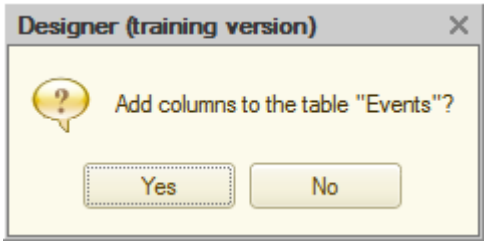

Figure 12-14. The confirmation of that columns will be added automatically

After that, you will have two tables on the form, displaying all attributes of the **Events** and the **People** catalogs.

Delete not required columns from the form. For that, in the **People** table holding the **Ctrl** key select **PeopleCode**, **PeopleGender**, and **PeopleRelationType** and then click **Delete current item** ✕ (Del).

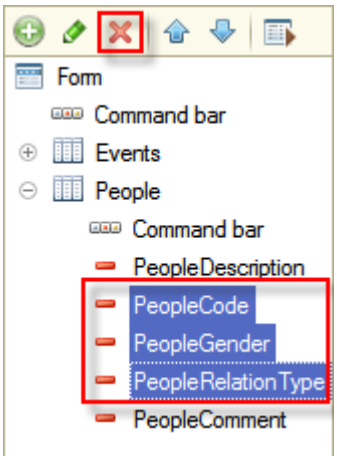

Figure 12-15. Deleting not required columns from the People table

From the **Events** table delete the **EventCode** and the **EventCategory** columns.

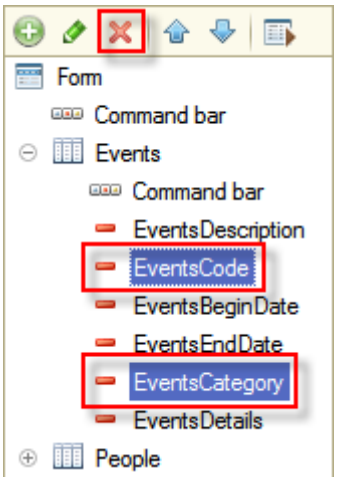

Figure 12-16. Deleting not required columns from the Events table

Open the **Starting page work area** editor (see Figure 8-19). For that, right-click the root node of the **Configuration** tree and then click **Open start page work area**. Delete the **Financial transactions** accumulation register list form from the **Left column** and the **People** catalog list form from the **Right column**. In **Starting page template**, select **One column**. After that, add the **MobileDesktop** common form to the list of forms.

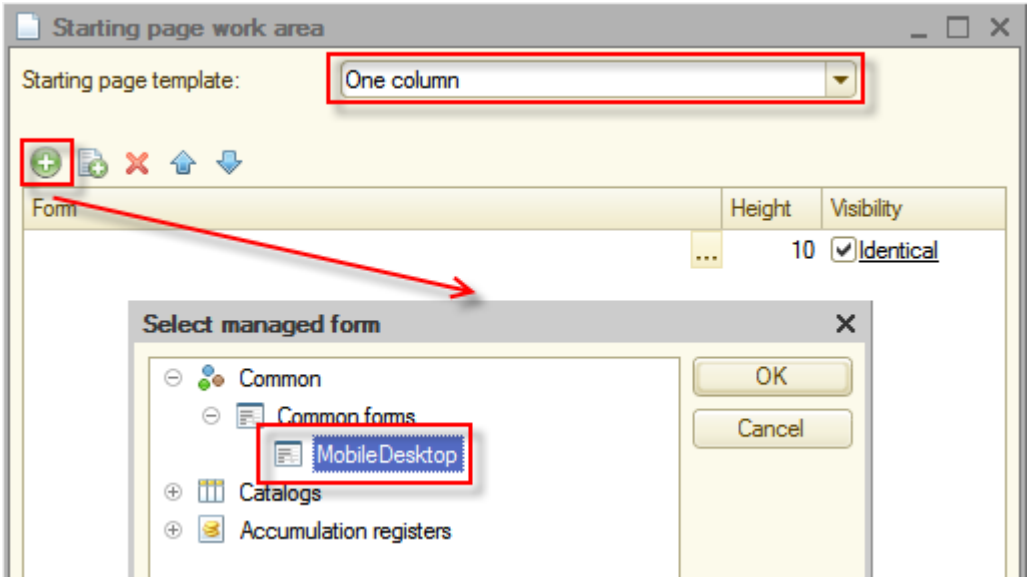

Figure 12-17. The single MobileDesktop form on the Start page

Now, you are done with the configuration changes. To save them press the **F7** key. You can check the configuration again (see Figure 12-2), this time there will be no errors found.

The last step is to upload and start this application on the mobile device. You will use the tool that is similar to publication on the web server that was described in Web client chapter on page 126.

The web server will keep the application as a single XML file. The mobile platform for developers connects to this web server, downloads that XML file and installs on the mobile device. After that, the application can be started on the mobile device.

In the Designer mode in **Main menu** click **Configuration**, then click **Mobile application**, and then click **Publish...**

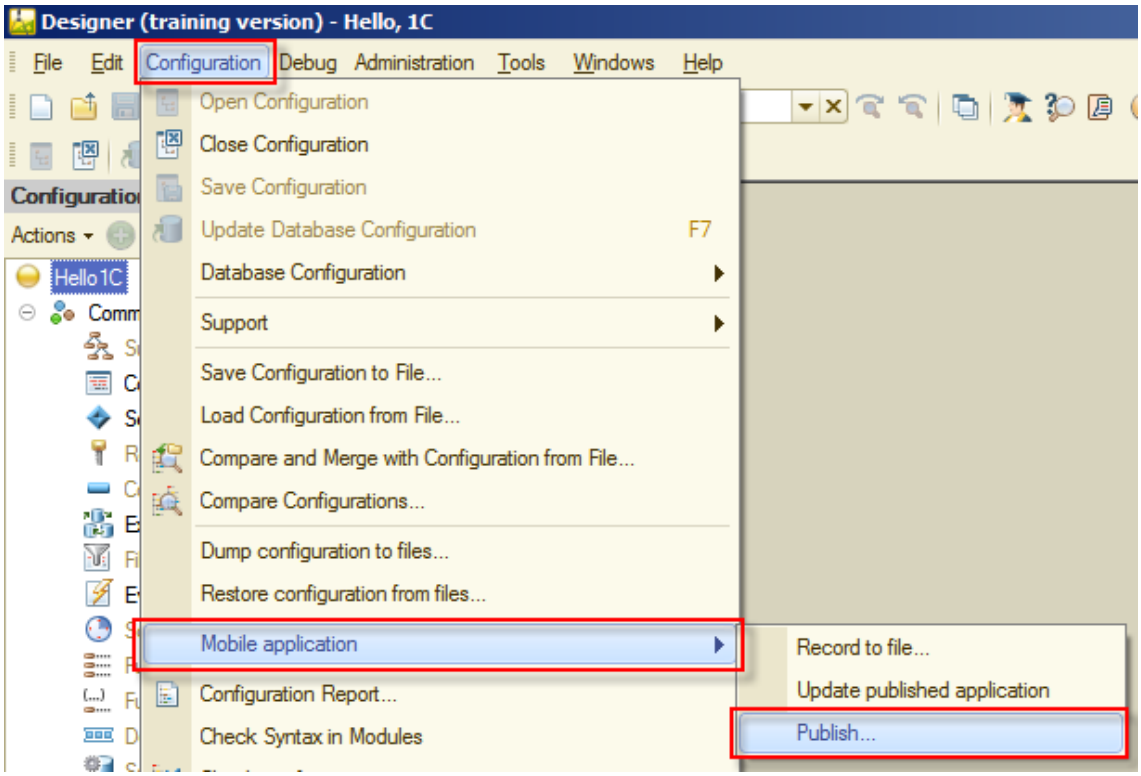

Figure 12-18. Opening Mobile application publishing

In the opened window select the **Create virtual directory on web server** and **Update mobile application when a database configuration is updated** check boxes, and input **hello_1c_mob** in **Name**. See that the last folder in **Directory** is also changed to **hello_1c_mob**. Then, click **Publish**.

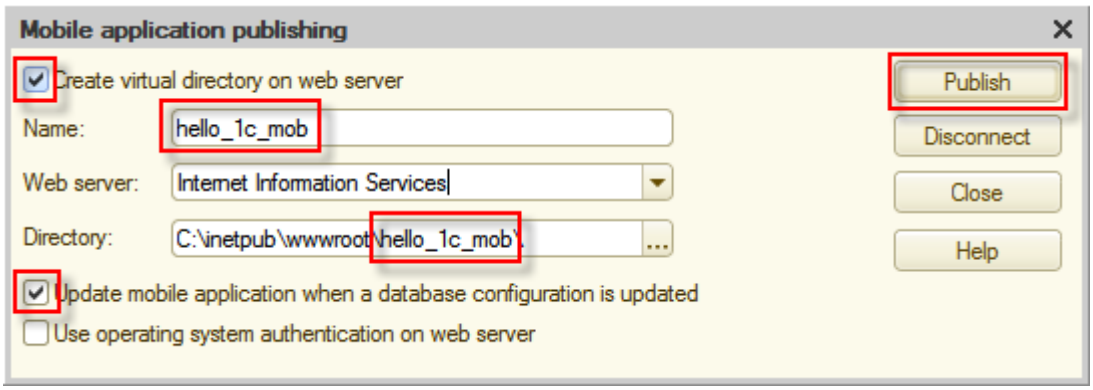

Figure 12-19. Publishing the mobile application

Confirm all actions that the platform will ask confirmation for, and your application will be published.

To install and start the application on iOS devices in the developer mode you will need the developer account. Moreover, you will need a Macintosh computer with XCode developer environment.

For Android devices, you only need to install 1C:Enterprise mobile platform for developers using packages that are available in the distribution kit of 1C:Enterprise mobile platform.

For this, upload one of APK installation packages to the mobile device, for example using USB cable. These files are available in **Android** folder of the distribution kit. Install **1cem-arm.apk** if you have a device with ARM

processor (suits for most devices) or **1cem-x86.apk** if you have a device with x86 processor.

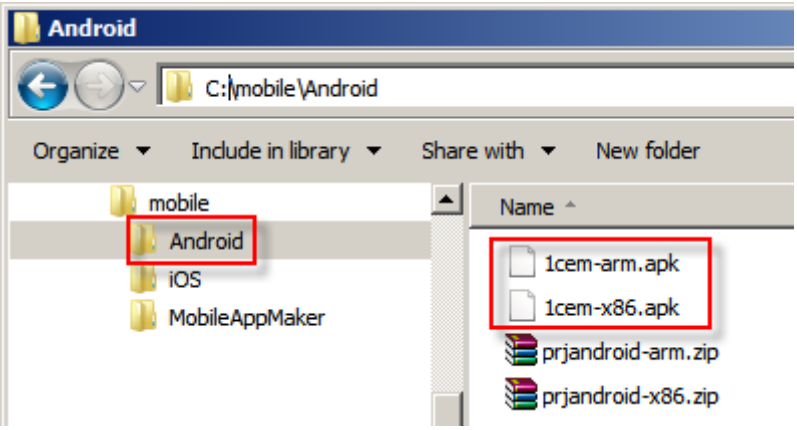

Figure 12-20. Mobile application for developer installation packages for Android

Then, run the installation package.

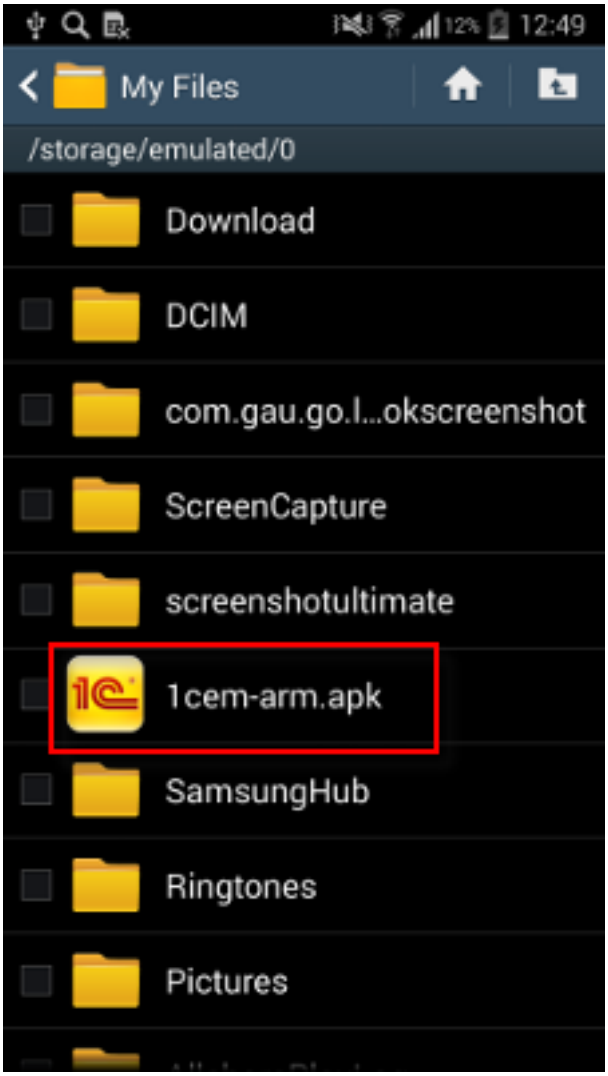

Figure 12-21. Running the installation package

It will require an access to several system features.

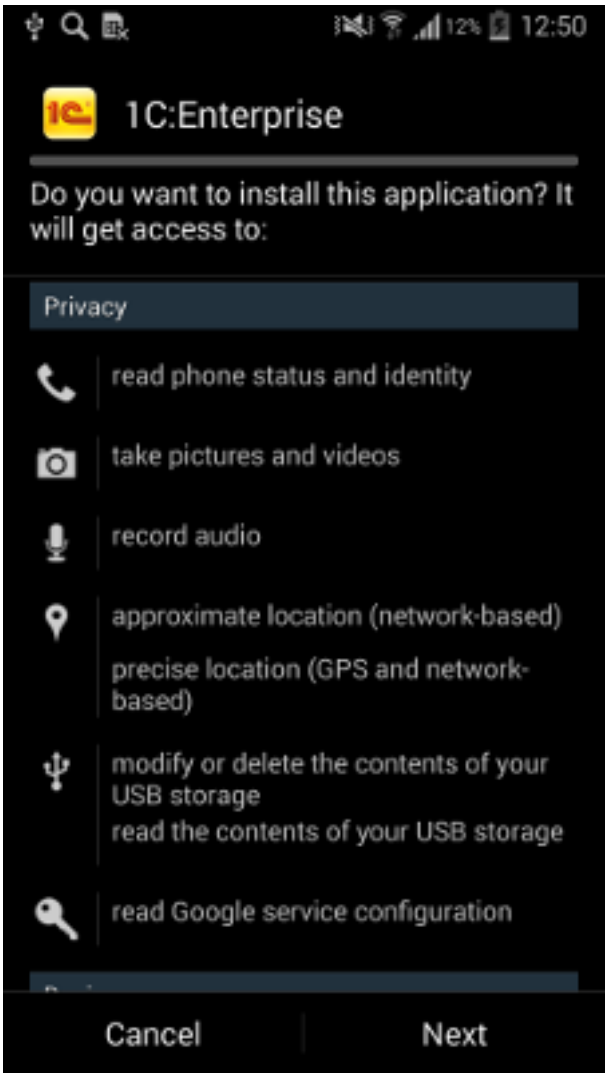

Figure 12-22. Starting the installation

After that, it will install in a few moments.

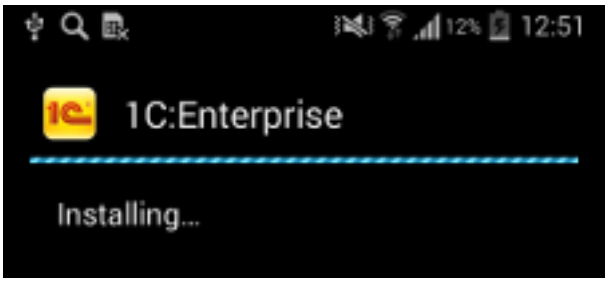

Figure 12-23. Installing the mobile application

When you will have the mobile platform for developers installed, tap **1C:Enterprise** icon to start it.

Figure 12-24. Starting the mobile platform for developers

If you start the mobile platform for the first time, it will show you an empty list of applications. To add a new application, tap **Add application**.

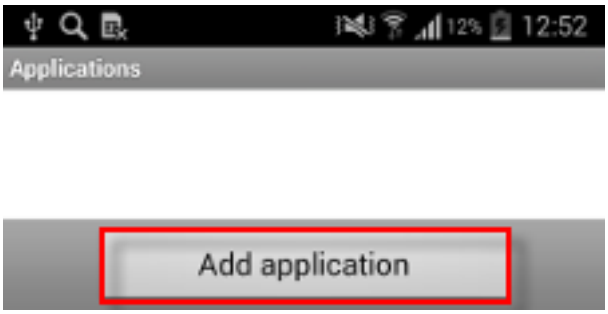

Figure 12-25. Adding a new mobile application

In **Address** input **http://*<your computer IP>*/hello_1c_mob**. Instead of *<your computer IP>* insert IP of the computer where you have the web server with published mobile application.

Notice: To find out the computer IP, see its network connection properties. The mobile device and the server should be in the same network. The easiest way of doing that is using a Wi-Fi router. Both, the computer with web server, and the mobile device should be connected to it.

Then, tap **Download**.

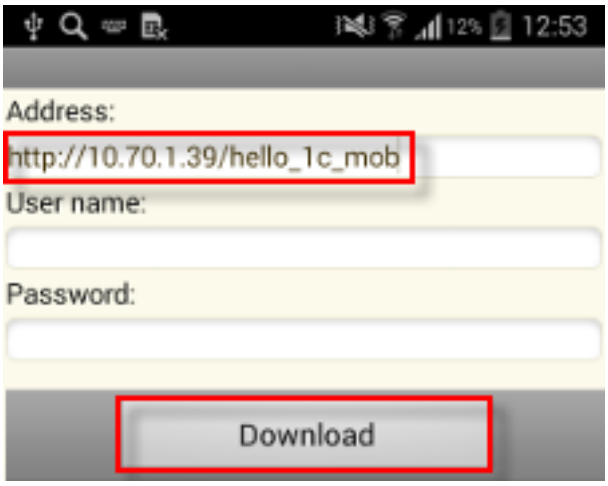

Figure 12-26. The mobile application URL

The mobile platform will download the application.

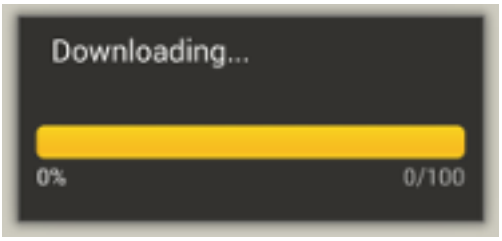

Figure 12-27. Downloading the mobile application

Then, the mobile platform will suggest you to adjust some properties. Leave **Application name** as it is and select the **Restart from designer** check box. Then, tap **Done**.

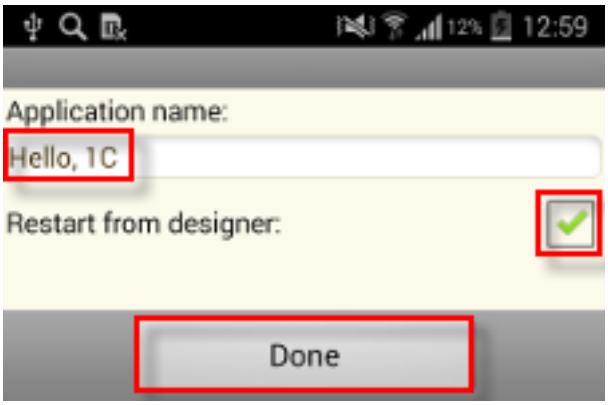

Figure 12-28. The mobile application startup parameters

The application will be installed.

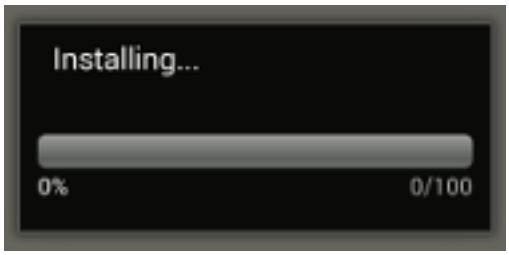

Figure 12-29. Installing the mobile application

You will see the application in the list of applications. To start it, tap the application name.

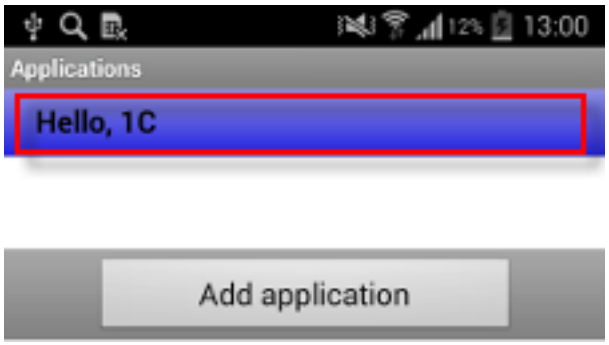

Figure 12-30. Starting the mobile application

The first screen that you see after that is the **Mobile desktop** form.

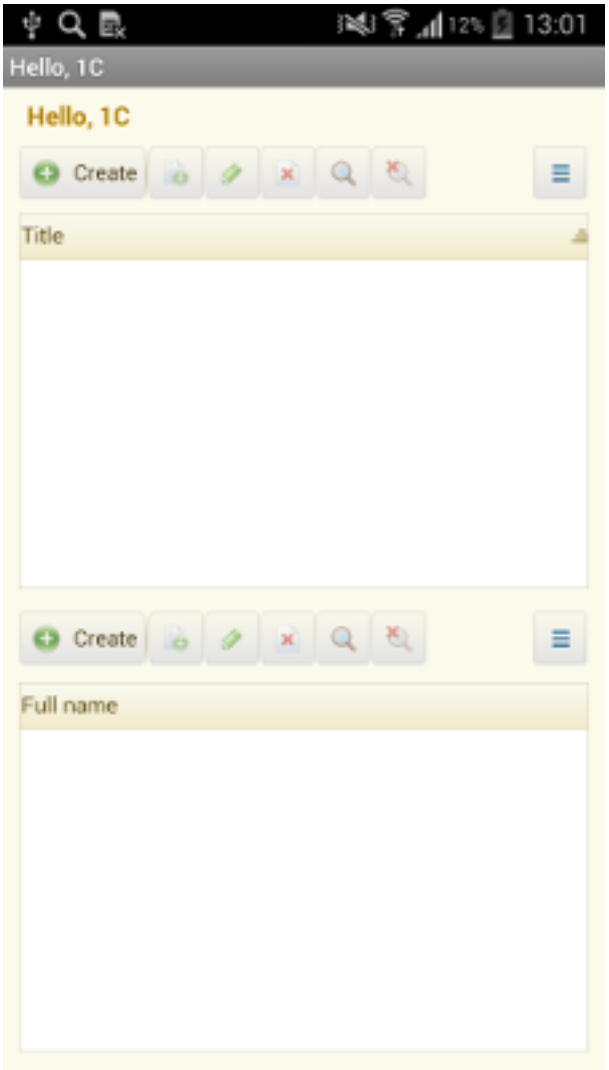

Figure 12-31. The Mobile desktop form

If you tap menu button of your mobile device you will see the application menu. Notice that the command interface of the **Main section** is displayed in the application menu in mobile platform.

Figure 12-32. The mobile application menu

The important notice: the application that you developed and filled with data, when opened on the mobile device contains no data. This is not an error. The data synchronization between several applications is an interesting, but separate topic. The 1C:Enterprise platform can solve these problems with ease. For an example, see Homework 3 of 1C mobile application rapid application development tutorial.

In this chapter spending only a few minutes you developed a mobile version of your CRM application, which is compatible with Android and iOS mobile devices.

To distribute your application for end users you need to create a solid package containing the mobile platform, the applied solution, and the database. For this, 1C:Enterprise platform has packing tools. You can read more about it in Mobile application building example chapter of 1C mobile application rapid application development tutorial. The packed application can be published in Google Play and Apple App Store.

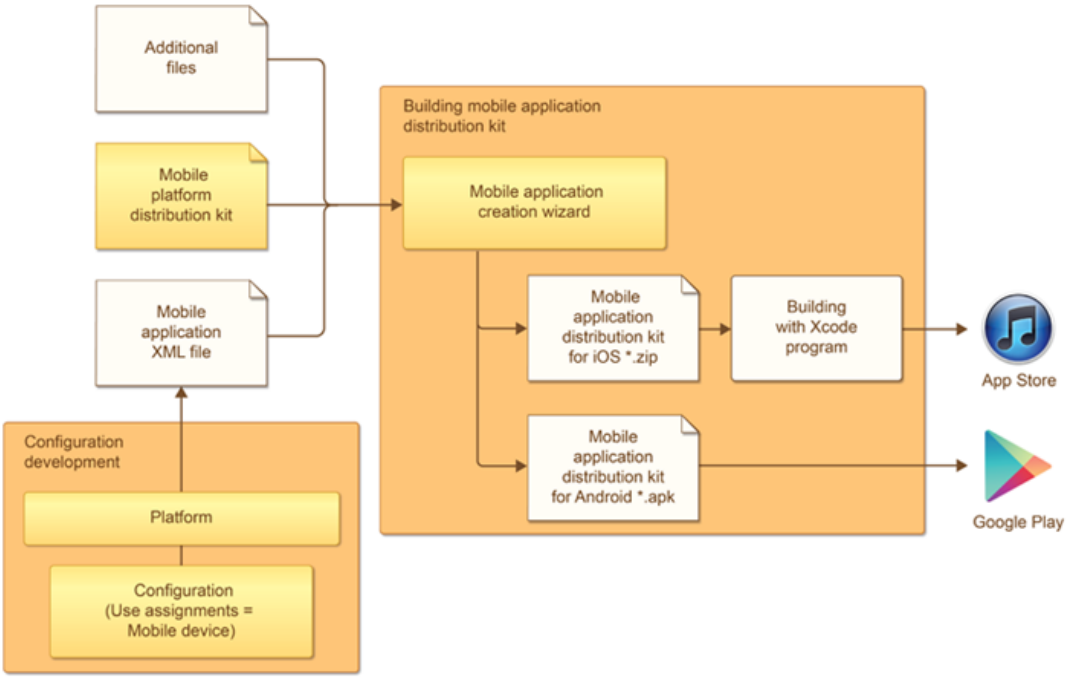

Figure 12-33. Publishing the mobile application for end users

Where and how to study 1C:Enterprise

The experience of many professionals prove that it is easy to successfully master 1C:Enterprise on your own. To help you with that, 1C Company publishes training versions of the platform, demo versions of applications, documentation, books, and tutorials. They are available free of charge on 1C:Developer Network.

1C:Enterprise 8 (training version)

1C:Enterprise 8 (training version) is an affordable solution for a wide range of users who want to get started with 1C:Enterprise 8. It allows you to learn development methods: create and edit the metadata structure, write script modules, configure forms and interfaces, and administrate the application.

You can also use the training version to modify existing applied solutions, keeping in mind its limitations. Applied solution configurations are the same for both demo and commercial versions. There are no limitations on the configuration complexity in the training version. However, the training version cannot be used for keeping of actual records, for this you need to purchase a commercial version of 1C:Enterprise platform.

Training version limitations are listed below.

- The volume of data is limited for documents, catalogs, registers, charts of accounts and other object tables:
  - Maximum number of records in account tables is 2000.
  - Maximum number of records in main object tables is 2000.
  - Maximum number of records in tabular sections of objects is 1000.
  - Maximum number of records in record sets is 2000.
  - Maximum number of records from external data sources is 200.
- Client/server mode is not supported.
- Distributed infobases are not supported.
- COM connection is not supported.
- It is not possible to use passwords and operating system authentication.
- Printing and saving spreadsheet documents are only supported in the Designer mode.
- Performance is lower compared to the commercial 1C:Enterprise 8 version.
- It is not possible to copy content of multiple cells of a spreadsheet document in the 1C:Enterprise mode.
- Operations with the configuration repository are not supported.
- Configuration delivery features are not available.
- Only one concurrent infobase session in possible.

To get 1C:Enterprise (training version) support, use the public Studying 1C:Enterprise platform forum.

You can download updates for 1C:Enterprise (training version) free of charge on 1C:Developer Network.

1C:AccountingSuite demo

1C:AccountingSuite is a small business accounting and inventory software. The applied solution supports US GAAP and IFRS accounting, and reporting standards. The software provides full visibility into purchasing and sales transactions, automates accounts receivable and accounts payable. 1C:AccountingSuite makes managing payments, receipts, and bank transactions a breeze, providing business owners with a real-time financial snapshot of operations.

The demo version lets you:

- emulate accounting of the live business;
- learn accounting and taxes calculation;
- learn how to adjust documents and reports;
- estimate economy on the manual labor cost;
- generate reports, including Balance sheet and Profit and loss statement;
- adjust and adopt the applied solution.

The demo version is not intended to be used in live business accounting according to limitations of 1C:Enterprise (training version) (see page 149).

The demo version includes:

- 1C:AccountingSuite applied solution;
- 1C:AccountingSuite user manual PDF book;
- 1C:Enterprise (training version);
- Installation instructions.

You can find this application description, download files and support in 1C:AccountingSuite section of 1C:Developer Network.

Business automation

1C Company has built a software development, distribution, and implementation business, which has enabled thousands of companies to start and grow profitable business. By automation of management and accounting fields 1C Company provided hundreds of thousands of companies and individuals with an access to the mission-critical information.

1C Company created a prestigious profession and even industry. Hundreds of thousands of competent professionals are helping companes and individuals to improve their efficiency by using the state-of-art 1C:Enterprise automation platform (see page 151).

Most of employees of 1C Company and its partners graduated from universities and colleges. Therefore, one of 1C Company priorities, as the leading software developer, is to cooperate with students and educational institutions to give young people the opportunity to gain practical skills of working with 1C:Enterprise platform. As well as to assist in employment to the position that allows to utilize acquired skills.

1C:Enterprise 8

1C:Enterprise 8 application collection is designed for the automation of management and accounting based on the state-of-art 1C:Enterprise platform. The platform provides extensive functionality, flexibility, and scalability starting from single user applications in file mode and until cloud applications with server clusters deployment. Features and architecture of 1C:Enterprise 8 platform is designed in anticipation of global trends in the business automation. Many of 1C:Enterprise solutions are unique and are beyond the competition.

1C Company (see page 154) and its partners (see page 152) using 1C:Enterprise 8 platform develop mass-market, industrial, and custom business solutions, including following:

- 1C:AccountingSuite (see page 150)
- 1C:Small Business
- 1C:Translator

1C:Enterprise 8 offers:

- To Director of Development: tools for analysis, planning, and flexible resource management, that gains its competitive advantages.
- To Department Director, managers, and employees who are directly involved with production, marketing, logistics, and other activities: tools to gain the efficiency of their daily work.
- To accounting department: tools for automated record keeping that is in full compliance with both the legal requirements and the corporate standards of the company.
- To IT professionals: tools and environment for the development, modification, deployment, administration, and maintenance of corporate information systems that meet latest standards.

Mass-market, industrial, and custom business solutions powered by 1C:Enterprise let you:

- Find the optimal automation strategy.
- Implement the application spending minimum time and resources.
- Quickly get real benefits from the implementation.
- Simplify the user training, maintenance, and administration of the application.
- Develop the system in accordance with the needs of the company without any downtime.

All of this ensures the high efficiency of 1C:Enterprise specialists and is a foundation of their success on the automation market.

1C partners

International experience shows that bulk sales of technically complex products can only be effectively managed through a well-organized and diversified dealer network.

1C partners are underlying distribution and technical support centers. Clients perceive 1C partners to be representatives of 1C Company that work side by side with them. If one of users finds it difficult to use a new application to prepare an urgently needed Balance sheet or P&L, for example, he would want to be able to obtain sound advice as soon as possible. The fastest way to get such advice is from a local partner.

1C Company always strives to understand and take account of the interests of its partners, not just focus on the benefits for itself. 1C Company has a direct interest in growing total sales through the partner network and consequently tries to provide its partners with the best terms and conditions for cooperation and provide all possible manner of support. This has enabled 1C Company to attract more than 10,000 regular partners in more than 800 cities in Eastern Europe.

1C partner program consists of several levels. **1C:Official Partner** introductory status was established for partners who have not developed their own applied solutions powered by the 1C:Enterprise platform and need to acquire 1C:Enterprise licenses to implement projects for their clients. **1C:Solution partner** status is intended for partners who have an applied solution, which has been certified by 1C Company. The **1C:Service provider** status is established for those who are planning to organize and provide services based on products created on 1C:Enterprise platform. **1C:Regional partner** status is intended for partners who have their own certified applied solutions and sell them through their own regional partner network.

More details on how you can become a 1C partner you can read in Partner program information section of 1C:Developer Network.

Useful online resources

1C:Developer Network (http://1c-dn.com)

1C:Developer Network is the information resource for developers, creating business solutions on 1C:Enterprise platform. 1C:Developer network is a good choice for both novice and experienced developers. It is possible to learn from grounds and create a business solution by using 1C:Developer network documentation and support.

Everyone can create an account and use the materials provided on 1C:Developer Network without any fee. To obtain free of charge all needed tools for learning and beginning of development. Each user can ask a question or share his ideas and experience in 1C:Enterprise at the online forum on 1C:Developer Network.

On the website you can find up-to-date information on all aspects of 1C Company activities:

- Support
- List of 1C partners
- News for partners and users
- Information on training and certification
- Press information
- Price list
- Jobs

To help you continue learining 1C:Enterprise use following references:

- 1C:Enterprise platform documentation: user, administrator and developer guides.
- Learn section allows you to study 1C:Enterprise platform using samples and tutorials.
- 1C:Developer Network forums for developers, as well as users and partners.
- 1C:Enterprise platform 1C:Enterprise overview and key features.
- Best practices section contains standards and best practices for 1C:Enterprise developers.
- Demo applications for studying specific features of 1C:Enterprise platform.
- Video section is a collection of training videos for quick and easy 1C:Enterprise learning.
- Applications section is a collection of advanced applications based on 1C:Enterprise platform.
- System requirements contain a list of supported OS and DBMS.

About 1C Company

1C Company was established in 1991, for the purposes of software development, distribution, publishing, and support of computer applications and databases for business and home use. 1C Company works with clients through more than 10,000 partners in 600 of 23 countries to provide integration services for the automation of businesses.

Apart from 1C:Enterprise, the best-known 1C Company business lines are software products for home use and educational applications.

The 1C:Enterprise collection of applications is used daily by several million users in business and government to automate operations, accounting, finance, HR, and management activities. 1C Company provides an array of vertical solutions for manufacturing, distribution, and service businesses. With its innovative 1C:Enterprise platform, and a range of other applications, 1C Company has achieved wide popularity for its openness, fast modifications and software updates. 1C:Enterprise is a very flexible and scalable platform, which meets the needs of companies ranging in size from a single user to hundreds of users. 1C Company is the market leader in enterprise automation in Russia, Ukraine, Kazakhstan, and Belarus and is used widely in the global market.

For references visit: 1C:Developer Network.