Creatio Academy

Develop your first application

Step 4. Implement the population of the timetable

Version 7.17

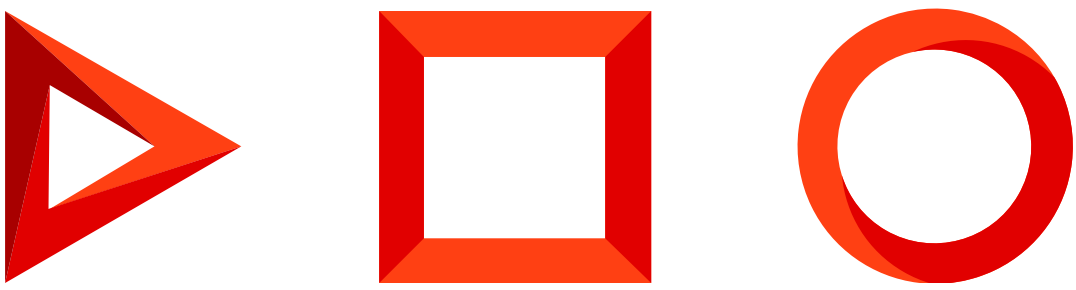


Table of Contents

Step 4. Implement the population of the timetable

Beginner

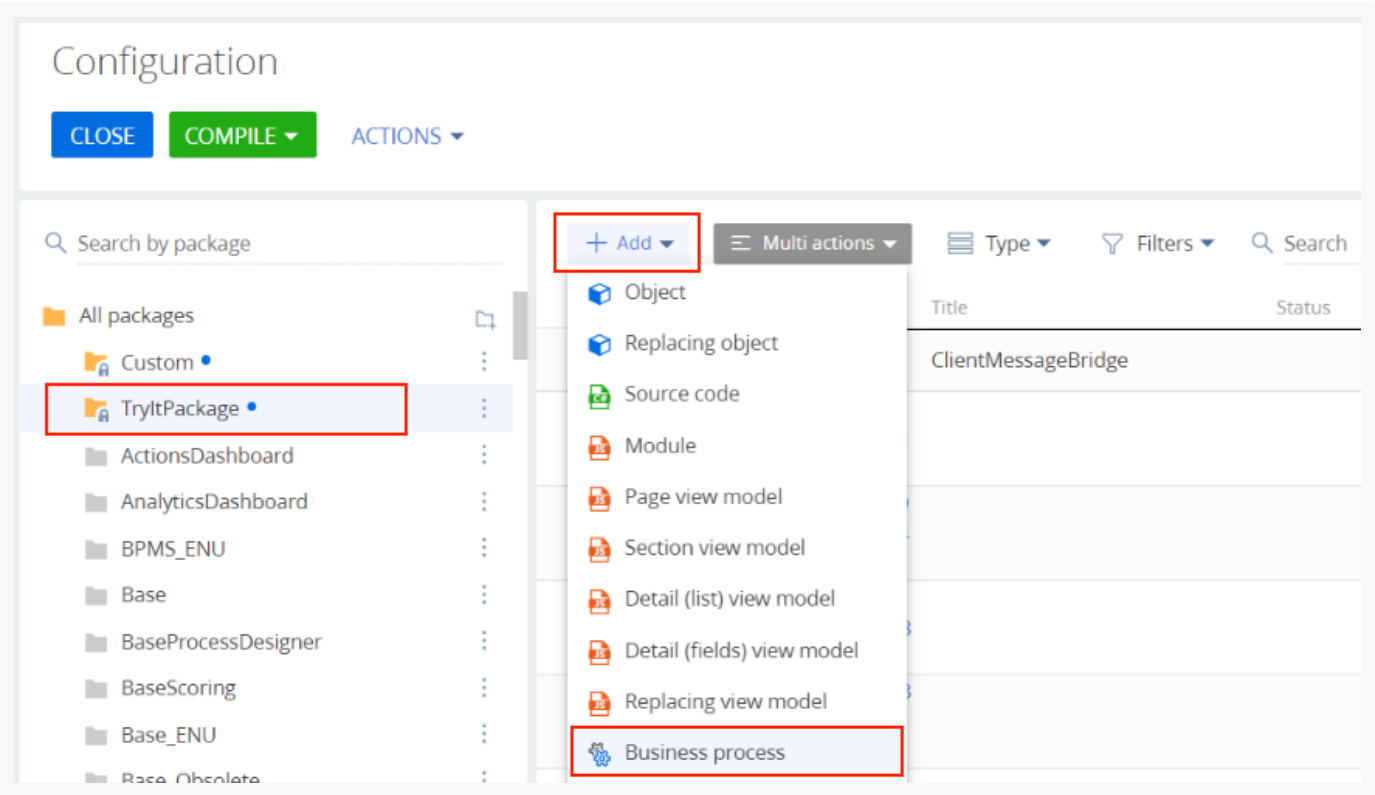
On the previous step, we added a check for the current daily class number matching the group gym number to the page.

Now, implement the population of the group class timetable. Add a page action that inserts 4 new records into the timetable.

To do this, create a business process that populates the database table with relevant data and returns a message to the page schema that the records were added successfully.

Create a business process

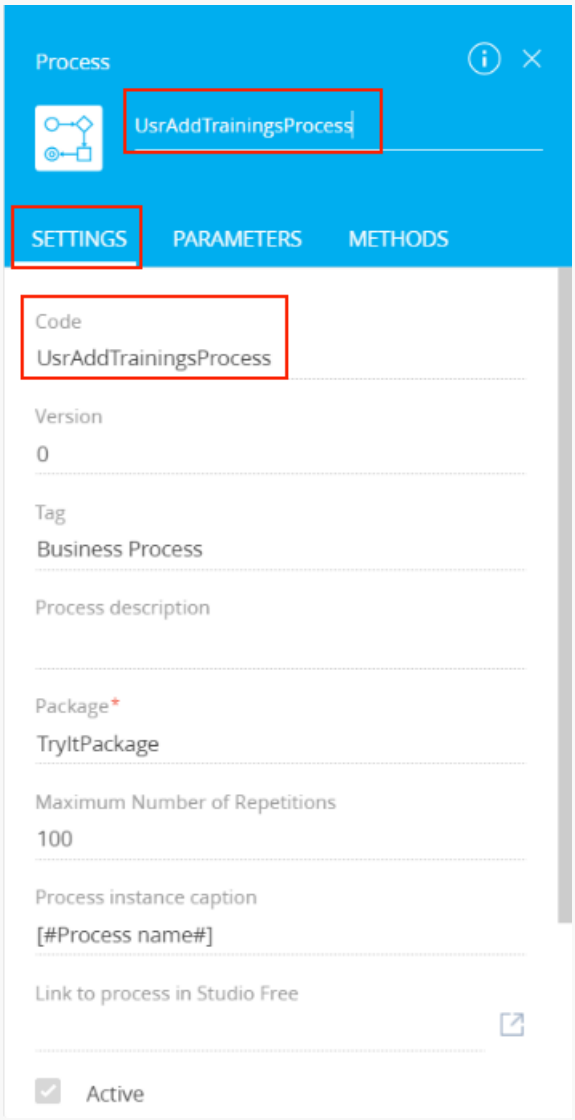
1. Go to the [*Configuration*] section.
2. Select the "TryItPackage" package from the package list.
3. Click [*Add*] on the workspace toolbar and select the [*Business process*] configuration element type.



4. **Fill out the property fields** as follows:

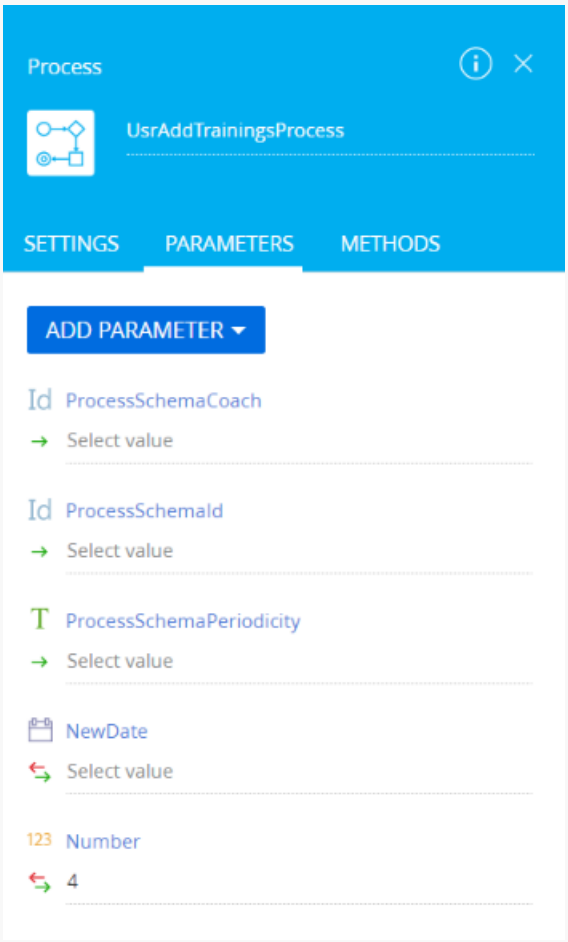
   - Set [*Name*] to "`UsrAddTrainingsProcess`."

- Set [*Code*] to "`UsrAddTrainingsProcess`."



5. Add the incoming parameters and parameters the process requires to the [*Parameters*] tab:

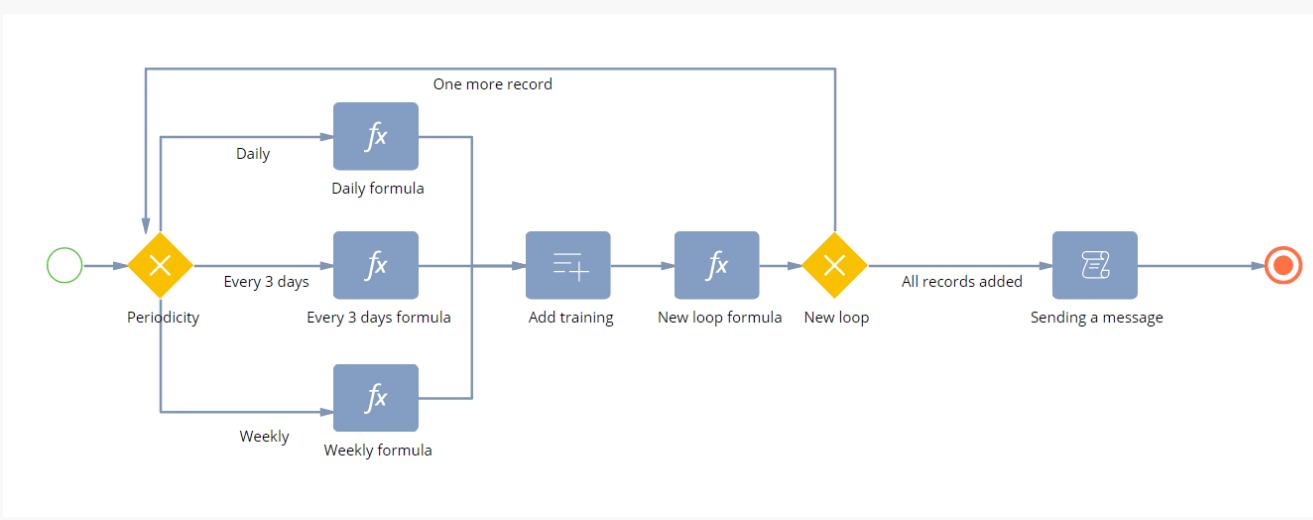
[***Title***]	[***Code***]	[***Data type***]	[***Direction***]	[***Value***]
`ProcessSchema Periodicity`	`ProcessSchema Periodicity`	Text (500 characters)	Input	
`NewDate`	`NewDate`	Date/Time	Bidirectional	
`Number`	`Number`	Integer	Bidirectional	4
`ProcessSchemaId`	`ProcessSchemaId`	Unique identifier	Input	
`ProcessSchemaCoach`	`ProcessSchemaCoach`	Unique identifier	Input	



6. Add the following elements to the process working area:

   - [*Exclusive gateway (OR)*]
   - [*Formula*]
   - [*Add data*]
   - [*Script task*]

   Connect the elements as specified in the figure below.



The process must run as follows:

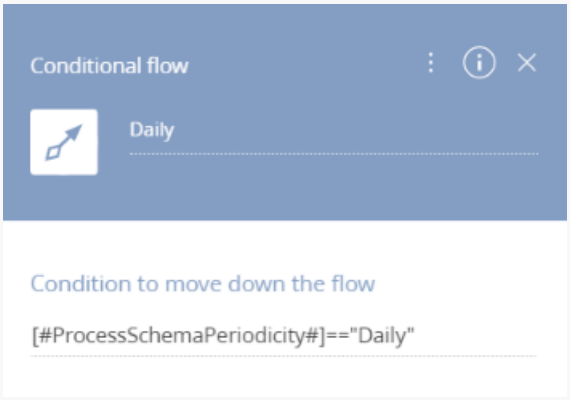
- The "Periodicity" [*Exclusive gateway (OR)*] gateway checks the periodicity of the class to add to the timetable and branches the process on the next step depending on the periodicity.
- The [*Formula*] elements calculate the class date depending on the periodicity. The process writes the date to the `NewDate` process parameter.
- The [*Add data*] element writes the new group class data to the database table.
- The "New loop formula" [*Formula*] element calculates the number of remaining empty records. For this example, the process must populate and write 4 new records. The results must be written to the `Number` process parameter.
- The "New loop" [*Exclusive gateway (OR)*] gateway checks the value of the `Number` process parameter. If it is more than 0, the process repeats the previous steps. If it equals 0, the process moves to the next step.
- The [*Script task*] element publishes the message that the new group class records were added.

7. Set up the business process elements.

   Periodicity [*Exclusive gateway (OR)*].

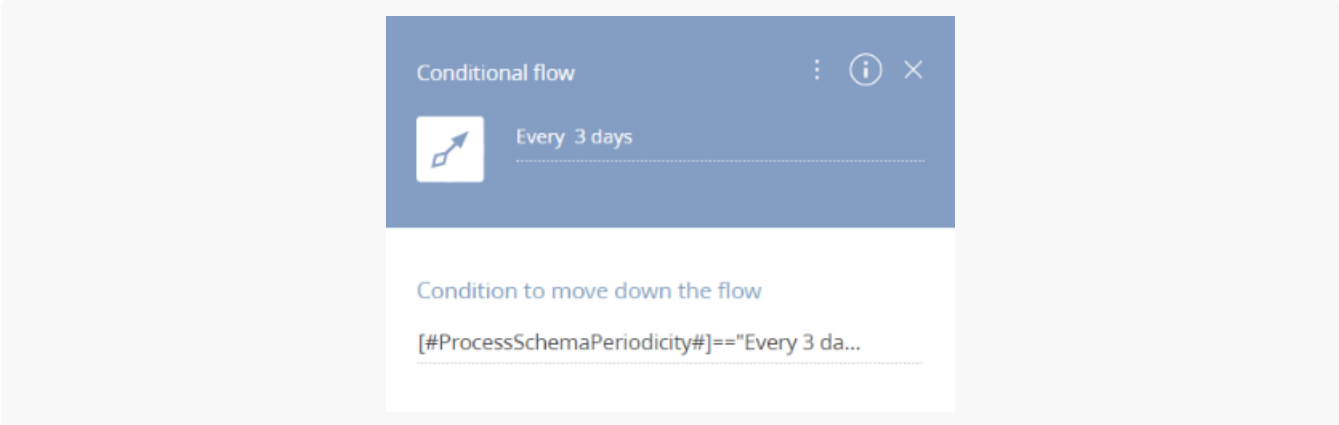
   a. Daily [*Conditional flow*]:

   - Set [*Condition to move down the flow*] to `[#ProcessSchemaPeriodicity#]=="Daily"`.



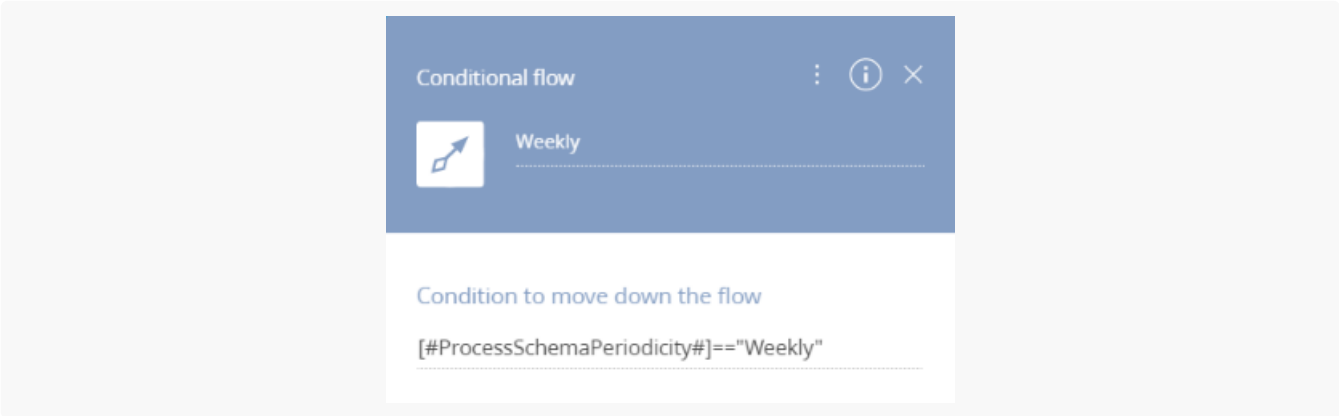
   c. Every 3 days [*Conditional flow*]:

   - Set [*Condition to move down the flow*] to `[#ProcessSchemaPeriodicity#]=="Every 3 days"`.



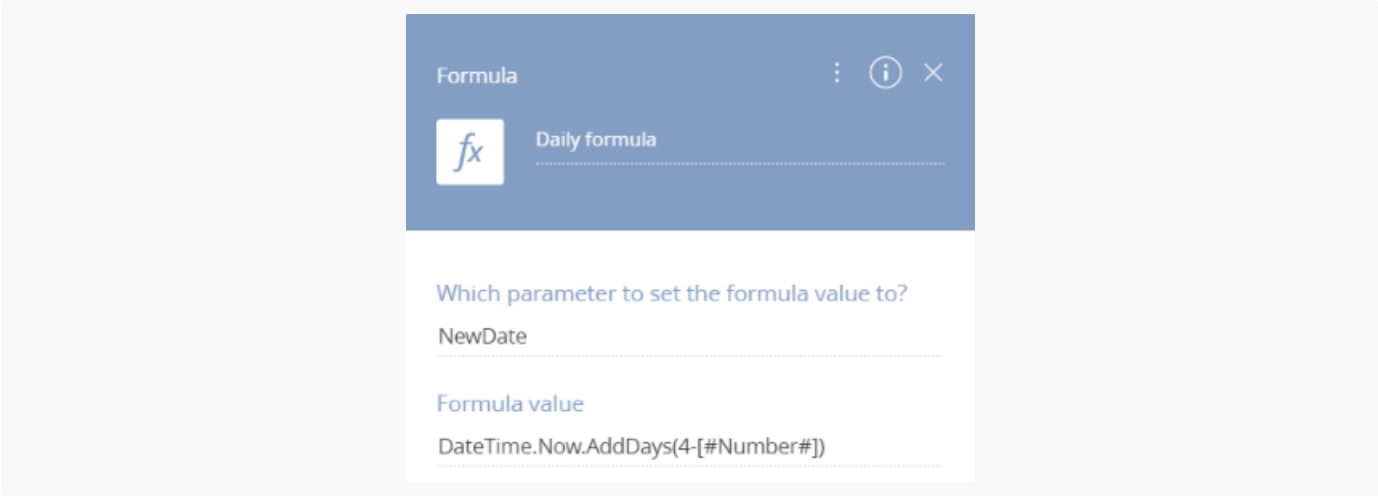
e. Weekly [*Conditional flow*]:

- Set [*Condition to move down the flow*] to `[#ProcessSchemaPeriodicity#]=="Weekly"`.



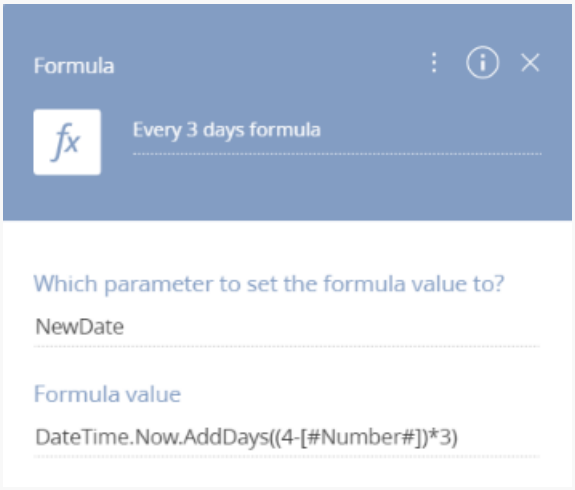
Daily formula [*Formula*]:

- Set [*Which parameter to set the formula value to?*] to "NewDate".
- Set [*Formula value*] to `DateTime.Now.AddDays(4-[#Number#])`.



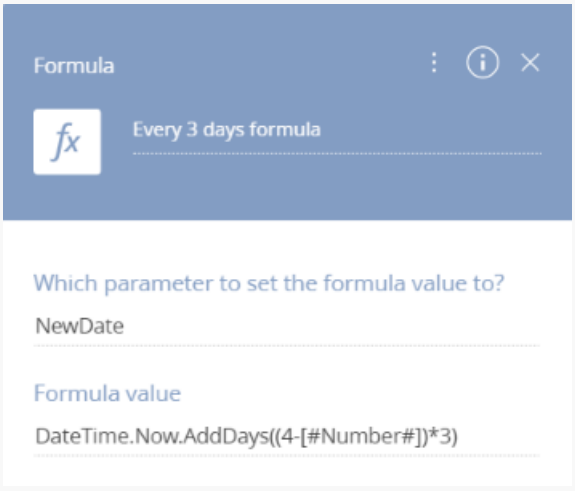
Every 3 days formula [*Formula*]:

- Set [*Which parameter to set the formula value to?*] to "`NewDate`".
- Set [*Formula value*] to `DateTime.Now.AddDays((4-[#Number#])*3)`.



Weekly formula [*Formula*]:

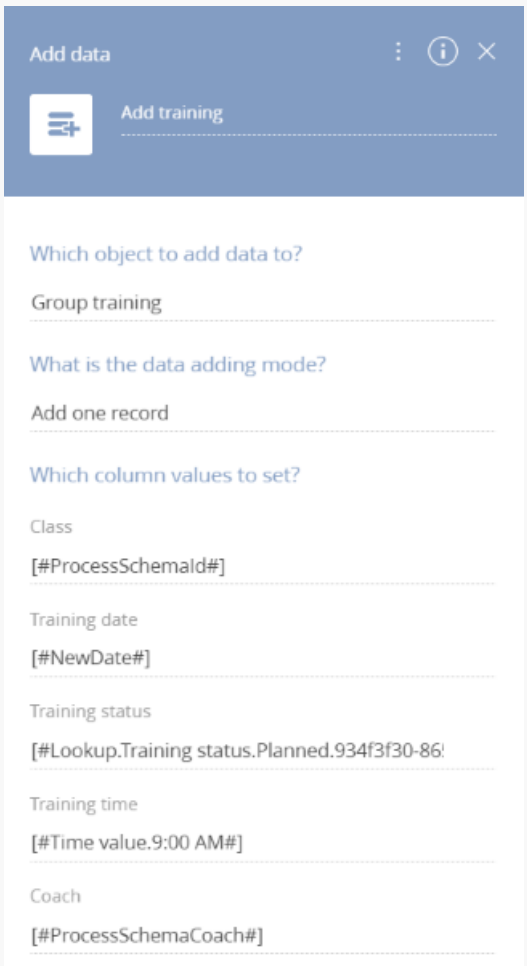
- Set [*Which parameter to set the formula value to?*] to "`NewDate`".
- Set [*Formula value*] to `DateTime.Now.AddDays((4-[#Number#])*7)`.



Add training [*Add data*]:

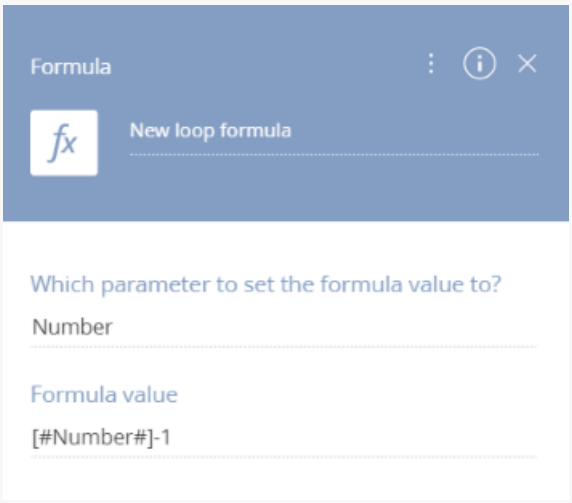
- Set [*Which object to add data to?*] to "`Group training`".
- Set [*What is the data adding mode?*] to "Add one record".
- Set [*Which column values to set?*] as follows:

Object column	Type of data to add	Value
`[Class]`	Process parameter	`ProcessSchemaId`
`[Training date]`	Process parameter	`NewDate`
`[Training status]`	Lookup value	`Planned`
`[Training time]`	Time selection	`09:00 AM`
`[Coach]`	Process parameter	`ProcessSchemaCoach`



New loop formula [*Formula*]:

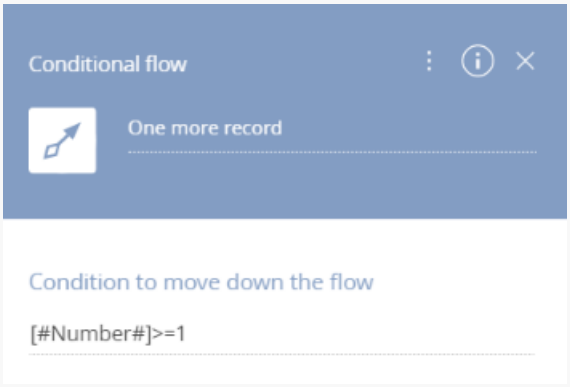
- Set [*Which parameter to set the formula value to?*] to "`Number`".
- Set [*Formula value*] to `[#Number#]-1`.



New loop [*Exclusive gateway (OR)*].

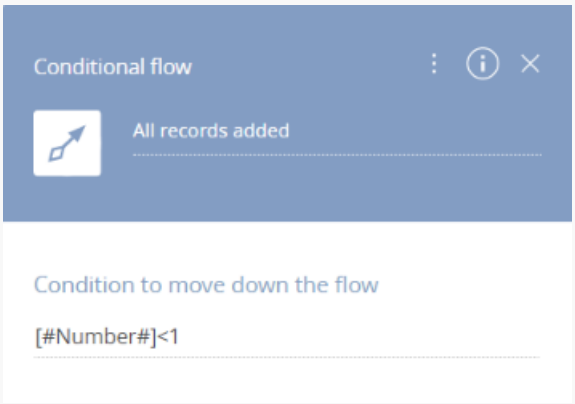
a. One more record [*Conditional flow*]:

- Set [*Condition to move down the flow*] to `[#Number#]>=1`.



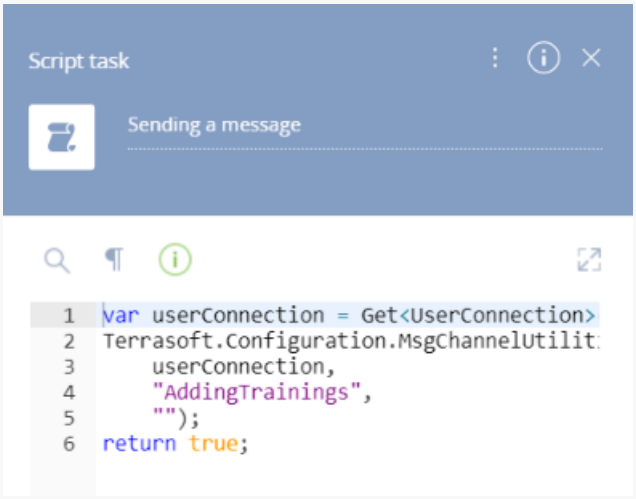
c. All records added [*Conditional flow*]:

- Set [*Condition to move down the flow*] to `[#Number#]<1`.



Sending a message [*Script task*]:

```
var userConnection = Get<UserConnection>("UserConnection");
Terrasoft.Configuration.MsgChannelUtilities.PostMessage(
    userConnection,
    "AddingTrainings",
    "");
return true;
```

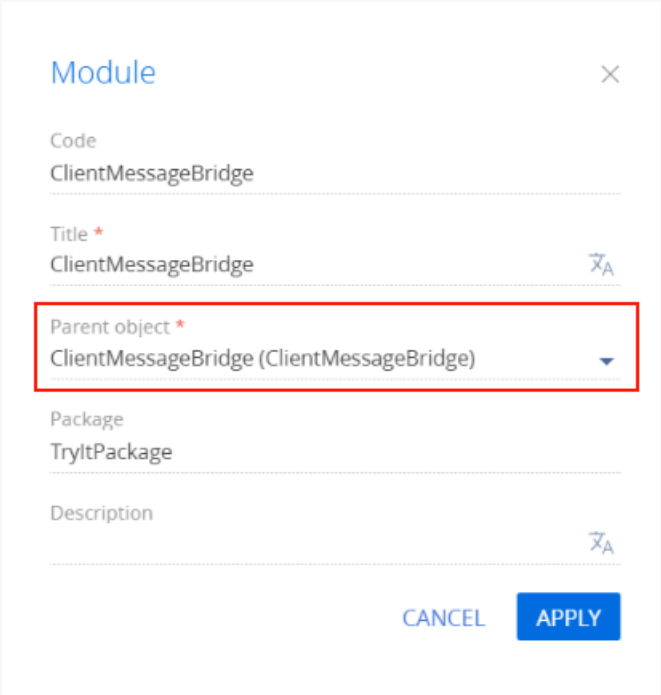


8. Save the process.

Create a replacing view model

Create a view model that replaces the `ClientMessageBridge` base schema to implement message passing between the process and the page module. The base schema implements the broadcasting of WebSocket messages to Creatio subscribers.

1. Go to the [*Configuration*] section.
2. Select the "TryItPackage" package from the package list.
3. Click [*Add*] on the workspace toolbar and select the [*Replacing view model*] configuration type.
4. Set the [*Parent object*] field to "`ClientMessageBridge`" ("`ClientMessageBridge`"). Creatio will populate the other fields.



5. Add the source code in the Schema Designer.

ClientMessageBridge.js

```
define("ClientMessageBridge", ["ConfigurationConstants"],
    function(ConfigurationConstants) {
        return {
            /* The messages. */
            messages: {
                /* The message name. */
                "AddingTrainings": {
                    /* Set the message type to broadcast, do not specify the subscriber. */
                    "mode": Terrasoft.MessageMode.BROADCAST,
                    /* Set the message direction to publishing. */
                    "direction": Terrasoft.MessageDirectionType.PUBLISH
                }
            },
            methods: {
                /* Initialize the schema. */
                init: function() {
                    /* Call the parent method. */
                    this.callParent(arguments);
                    /* Add a new configuration object to the configuration object collection.
                    this.addMessageConfig({
                        /* The name of the message received via WebSocket. */
                        sender: "AddingTrainings",
                        /* Send the message with this name. */
                        messageName: "AddingTrainings"
                    });
```

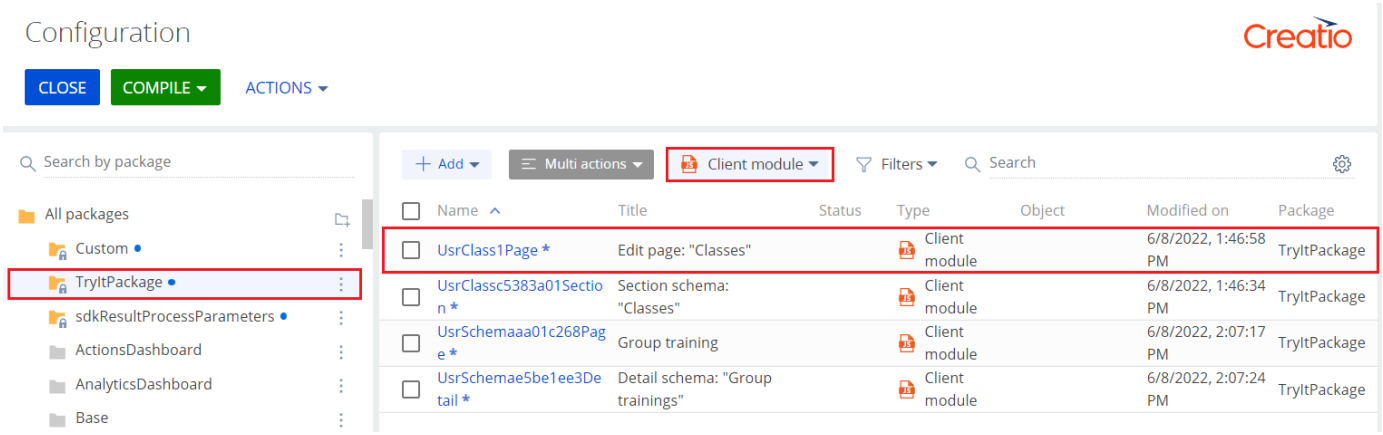
```
                }
            }
        };
    });
```

6. Click the [*Save*] button to save the schema.

Modify the page source code

Add the class page action that adds new group classes to the "Group trainings" detail.

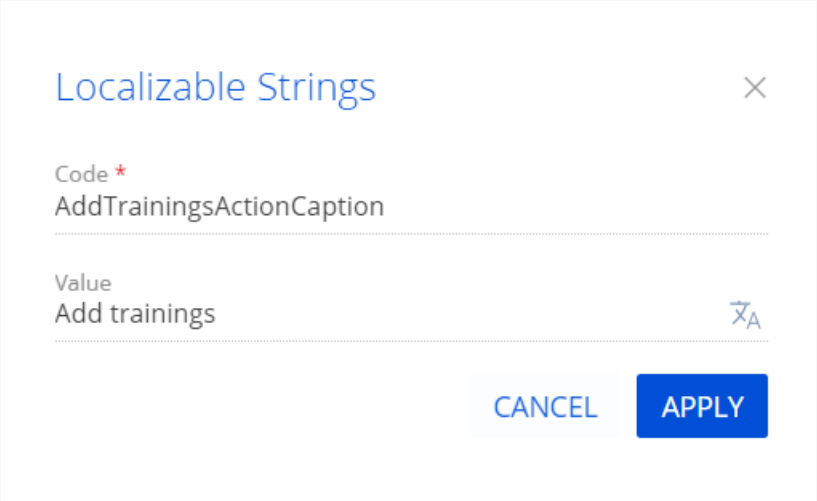
1. Go to the [*Configuration*] section.
2. Select the "TryItPackage" package from the package list.
3. The Wizards added schemas of various types to the package. Filter schemas by the [*Client module*] type.



4. Double-click the `UsrClass1Page` schema to open it.
5. Add a new localizable string for the action name to the schema.

   Click the + button in the [*Localizable strings*] block of the properties panel and fill out the **localizable string properties**:

   - Set [*Code*] to "`AddTrainingsActionCaption`".
   - Set [*Value*] to "`Add trainings`".



6. Modify schema source code:

UsrClass1Page.js

```
define("UsrClass1Page", ["ProcessModuleUtilities"], function(ProcessModuleUtilities) {
 return {
  entitySchemaName: "UsrClass",
  messages: {
   /* The message that calls the detail update. */
   "AddingTrainings": {
    "mode": Terrasoft.MessageMode.BROADCAST,
    "direction": Terrasoft.MessageDirectionType.SUBSCRIBE
   }
  },
  details: /**SCHEMA_DETAILS*/{
   // ...
  }/**SCHEMA_DETAILS*/,
  /* Add new methods to the existing methods. */
  methods: {

   // ...

   init: function() {
    this.callParent(arguments);
    /* Subscribe to the message that calls the detail update. */
    this.sandbox.subscribe("AddingTrainings", this.updateTrainings, this);
   },
   /* Add the action to the action menu. */
   getActions: function() {
    var actionMenuItems = this.callParent(arguments);
    actionMenuItems.addItem(this.getButtonMenuItem({
     "Caption": {bindTo: "Resources.Strings.AddTrainingsActionCaption"},
     /* Define the handler method for the action. */
     "Click": {bindTo: "getBusinessProcessAddTrainings"},
```

```
      "Enabled": true
     }));
     return actionMenuItems;
    },
    /* Call the update of the record page detail. */
    updateTrainings: function() {
     this.updateDetail({
      /* The detail code from details section.*/
      "detail": "UsrSchema12c4c6adDetail5a26acfb",
      "reloadAll": true
     });
    },
    /* The handler method for the new menu action. */
    getBusinessProcessAddTrainings: function() {
     /* Retrieve the incoming parameters the process requires. */
     var id = this.get("Id");
     var periodicity = this.get("UsrPeriodicity").displayValue;
     var coach = this.get("UsrCoach").value;
     if (!periodicity) {
      return;
     }
     /* Create a configuration object for running the process. */
     var args = {
      /* The name of the process you created on the previous steps. */
      sysProcessName: "UsrAddTrainingsProcess",
      /* The incoming process parameters. */
      parameters: {
       ProcessSchemaId: id,
       ProcessSchemaPeriodicity: periodicity,
       ProcessSchemaCoach: coach
      }
     };
     /* Run the process. */
     ProcessModuleUtilities.executeProcess(args);
    },
   },
   /* No changes. */
   diff: /**SCHEMA_DIFF*/[

   // ...

   ]/**SCHEMA_DIFF*/
  };
});
```

7. Click [*Save*] to save the schema.

As a result, we implemented the population of the group class timetable. On the next step, implement the web service that provides the information about the number of classes in the timetable.