

# Develop your first application

## Step 3. Add page validation

Version 7.17



This documentation is provided under restrictions on use and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this documentation, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

# Table of Contents

Step 3. Add page validation

4

# Step 3. Add page validation



On the [previous step](#), we added the example's required data to the interface and attached it to the development package.

Now, implement the business logic of the example.

## Provision.

- A daily class is scheduled only if there is an unoccupied gym.
- The number of the fitness center's gyms is set via a system setting and equals 4.
- When adding or editing a daily group class, Creatio must check if the changes result in the total number of active daily classes exceeding the value of the system setting. If yes, saving is disallowed and the message "No available gyms, no more than N classes allowed" is displayed, where N is the system setting's value.

To do this, modify the source code of the client module responsible for the section page operation.

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

The screenshot shows the 'Configuration' window in Creatio. On the left, a sidebar lists packages, with 'TryItPackage' selected. The main area displays a table of schemas filtered by 'Client module' type. The table has columns: Name, Title, Status, Type, Object, Modified on, and Package. The first row, 'UsrClass1Page', is highlighted with a red box. Below the table, a 'Client module' dropdown menu is also highlighted with a red box.

| Name                      | Title                            | Status | Type          | Object | Modified on          | Package      |
|---------------------------|----------------------------------|--------|---------------|--------|----------------------|--------------|
| UsrClass1Page *           | Edit page: "Classes"             |        | Client module |        | 6/8/2022, 1:46:58 PM | TryItPackage |
| UsrClassc5383a01Section * | Section schema: "Classes"        |        | Client module |        | 6/8/2022, 1:46:34 PM | TryItPackage |
| UsrSchemaaa01c268Page *   | Group training                   |        | Client module |        | 6/8/2022, 2:07:17 PM | TryItPackage |
| UsrSchemae5be1ee3Detail * | Detail schema: "Group trainings" |        | Client module |        | 6/8/2022, 2:07:24 PM | TryItPackage |

4. Double-click the "UsrClass1Page" schema to open it and modify its source code.

### UsrClass1Page.js

```
define("UsrClass1Page", [], function() {
    return {
        entitySchemaName: "UsrClass",
```

```

messages: {},
attributes: {
    /* The attribute that stores the number of currently active daily classes. */
    "responseCollectionTrainings": {
        "dataValueType": Terrasoft.DataValueType.INTEGER,
        "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN
    },
    /* The attribute that stores the value of the system setting. */
    "maximumDailyActiveSections": {
        "dataValueType": Terrasoft.DataValueType.INTEGER,
        "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN
    }
},
methods: {
    /* Run when the page's schema is loaded and call the method that counts the current number of active daily classes. */
    onEntityInitialized: function(){
        this.callParent(arguments);
        this.getPeriodicityActiveNumber();
        this.getMaximumDailyActiveSections();
    },
    /* Calculate the current number of active daily classes and write the calculated value to the attribute. */
    getPeriodicityActiveNumber: function() {
        var periodicity = "Daily";
        var esqPeriodicity = this.Ext.create("Terrasoft.EntitySchemaQuery", {
            rootSchemaName: "UsrClass"
        });
        esqPeriodicity.addColumn("UsrName");
        var groupFilters = this.Ext.create("Terrasoft.FilterGroup");
        var filterPeriodicity = this.Terrasoft.createColumnFilterWithParameter(this.Terrasoft.FilterOperatorType.AND, "UsrName", periodicity);
        var thisId = this.get("Id");
        var filterId = this.Terrasoft.createColumnFilterWithParameter(this.Terrasoft.FilterOperatorType.AND, "Id", thisId);
        var filterIsActive = this.Terrasoft.createColumnFilterWithParameter(this.Terrasoft.FilterOperatorType.AND, "IsActive", true);
        groupFilters.addItem(filterPeriodicity);
        groupFilters.addItem(filterIsActive);
        groupFilters.logicalOperation = this.Terrasoft.LogicalOperatorType.AND;
        groupFilters.addItem(filterId);
        esqPeriodicity.filters.add(groupFilters);
        esqPeriodicity.getEntityCollection(function(result) {
            if (!result.success) {
                this.showInformationDialog("Request error");
                return;
            }
            else {
                var lengthCollection = result.collection.collection.length;
                this.set("responseCollectionTrainings", lengthCollection);
            }
        }, this);
    },
}

```

```

/* Provide validation for the "Periodicity" field. The validator method will be c
setValidationConfig: function() {
    this.callParent(arguments);
    this.addColumnValidator("UsrPeriodicity", this.periodicityValidator);
},
/* The validator method: if the class is daily, compare the number of active dail
periodicityValidator: function() {
    var invalidMessage= "";
    var periodicity = this.get("UsrPeriodicity").displayValue;
    if (periodicity==="Daily") {
        var isActive = this.get("UsrIsActive");
        var myVariable = this.get("maximumDailyActiveSections");
        var lengthCollection = this.get("responseCollectionTrainings");
        if (lengthCollection >= myVariable && isActive) {
            invalidMessage = "The number of gyms is limited. No more than " + myV
        }
    }
    else {
        invalidMessage = "";
    }
    return {
        invalidMessage: invalidMessage
    };
},
/* Retrieve the value of the "GymNumber" system setting. */
getMaximumDailyActiveSections: function() {
    var myVariable;
    var callback = function(value) {
        myVariable = value;
    };
    this.Terrasoft.SysSettings.querySysSettingsItem("GymsNumber", callback, this)
    if (myVariable === undefined) {
        return;
    }
    else {
        this.set("maximumDailyActiveSections", myVariable);
    }
}
},
modules: /**SCHEMA_MODULES*/{}/**SCHEMA_MODULES*/,
/* No changes. */
details: /**SCHEMA_DETAILS*/{

// ...

}/**SCHEMA_DETAILS*/,
businessRules: /**SCHEMA_BUSINESS_RULES*/{}/**SCHEMA_BUSINESS_RULES*/,
dataModels: /**SCHEMA_DATA_MODELS*/{}/**SCHEMA_DATA_MODELS*/,

```

```

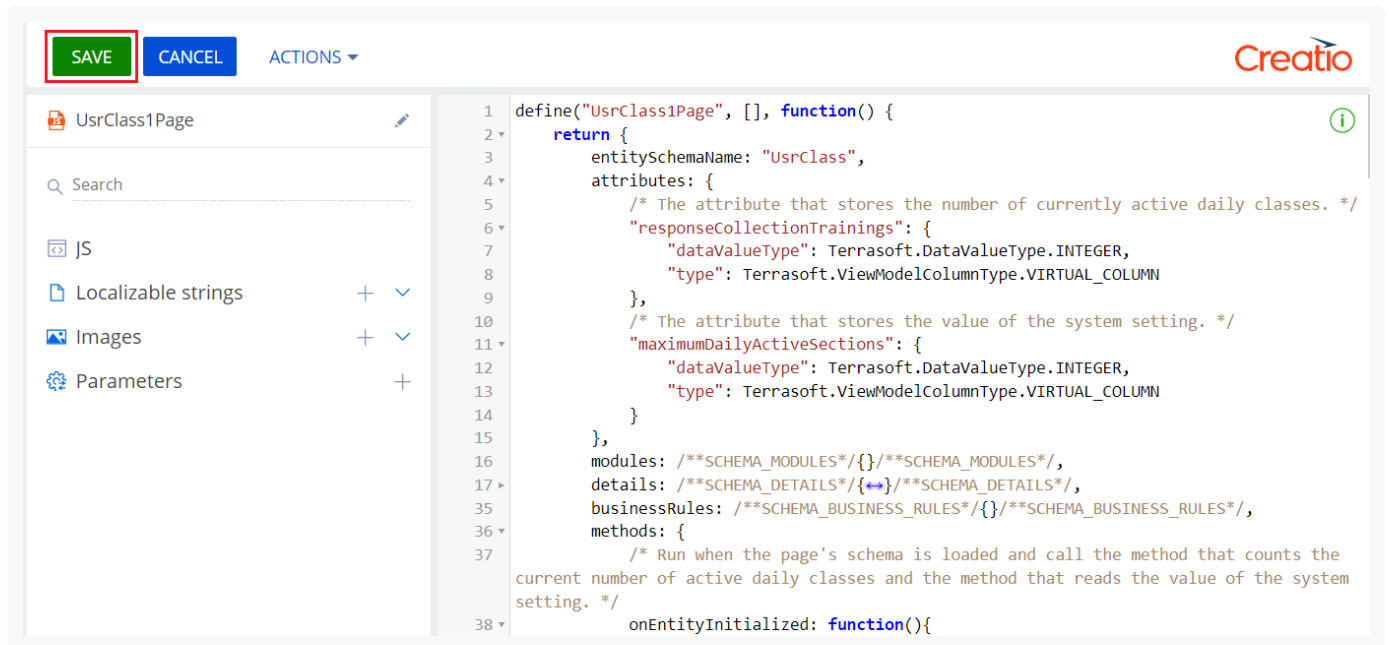
    /* No changes. */
    diff: /**SCHEMA_DIFF*/[

    // ...

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

```

5. Click the [ Save ] button to save the schema.



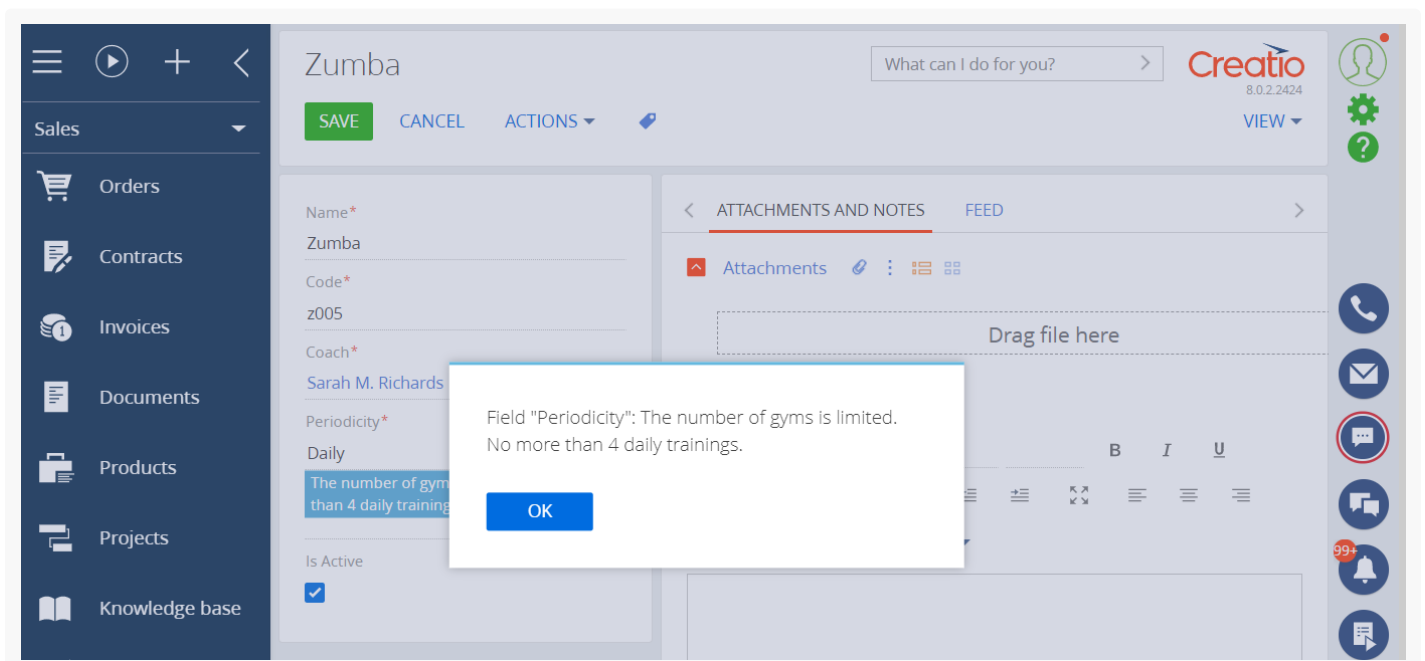
The screenshot shows the Creatio development environment. At the top, there are buttons for 'SAVE' (highlighted with a red box), 'CANCEL', and 'ACTIONS'. Below these is a sidebar with a search bar and a list of components: JS, Localizable strings, Images, and Parameters. The main area is a code editor showing the schema definition for 'UsrClass1Page'. The code is as follows:

```

1 define("UsrClass1Page", [], function() {
2   return {
3     entitySchemaName: "UsrClass",
4     attributes: {
5       /* The attribute that stores the number of currently active daily classes. */
6       "responseCollectionTrainings": {
7         "dataValueType": Terrasoft.DataValueType.INTEGER,
8         "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN
9       },
10      /* The attribute that stores the value of the system setting. */
11      "maximumDailyActiveSections": {
12        "dataValueType": Terrasoft.DataValueType.INTEGER,
13        "type": Terrasoft.ViewModelColumnType.VIRTUAL_COLUMN
14      }
15    },
16    modules: /**SCHEMA_MODULES*/{/**SCHEMA_MODULES*/,
17    details: /**SCHEMA_DETAILS*/{/**SCHEMA_DETAILS*/,
35    businessRules: /**SCHEMA_BUSINESS_RULES*/{/**SCHEMA_BUSINESS_RULES*/,
36    methods: {
37      /* Run when the page's schema is loaded and call the method that counts the
38      current number of active daily classes and the method that reads the value of the system
      setting. */
      onEntityInitialized: function(){

```

The result of the changes:



The screenshot shows the Creatio user interface. The top navigation bar includes the 'Zumba' record name, a search bar, and the 'Creatio 8.0.2.2424' logo. Below the navigation bar, there are buttons for 'SAVE', 'CANCEL', and 'ACTIONS'. The main content area shows the 'Zumba' record form with the following fields:

- Name\*: Zumba
- Code\*: z005
- Coach\*: Sarah M. Richards
- Periodicity\*: Daily
- The number of gyms: The number of gyms is limited. No more than 4 daily trainings.
- Is Active:

A validation error message is displayed in a dialog box:

Field "Periodicity": The number of gyms is limited.  
No more than 4 daily trainings.

The background shows the 'ATTACHMENTS AND NOTES' section with a 'FEED' tab and a 'Drag file here' area. The bottom right corner of the interface features a vertical toolbar with various icons, including a notification bell with '99+' and a help icon.

As a result, we implemented the required business logic of the page. On the [next step](#), implement the population of the group class timetable.