1. using System;
2. using System.Collections.Generic;
3. using System.Linq;
4. using System.Text;
5. using System.Threading.Tasks;
6. using System.Configuration; 7

8

9 using Nihulon2.Model;

10 using Nihulon2.Model.DbAccess; 11

12 namespace Nihulon2.RelationsList 13 {

14 /\*

1. \* The class that assumes the layer between the view and DataBase.
2. \* It provides methods that get data from the dbConnector and
3. \* calls the view methods to fill its controls with the new data.
4. \* This controller works with the items related to the exam
5. \* like divisions, courses and rooms 20 \*/

21 public class RelationsList\_Controller 22 {

1. private IRelationsListView \_view; // An instance of the view for data visualization
2. private DbConnector dbConn; // An instance of the class that provides connection to the DB
3. private RelatedItem[] relatedItems; // Items that are in the data grid of RelationsList\_View
4. private string selectedType; // The type that is selected in the combo box of the view

27

1. // Defines if disabled related items are going to be shown
2. public bool ShowDisabled

30 {

1. get;
2. set;

33 }

34

35 /\*

1. \* The constructor. It initializes instances of the dbConnector and
2. \* the view classes, and sets default values for flags and comboBoxes
3. \* view - The instance of a view that will work with this controller 39 \*/

40 public RelationsList\_Controller(IRelationsListView view) 41 {

1. // Initializing connection to DB
2. dbConn = DbConnector.Instance; 44

45 ShowDisabled = false; // Don't show disabled related items by default

46

1. // Binding with view
2. \_view = view;
3. \_view.setController(this);
4. // set divisions as default related items and fill the data grid of the view

51

52

53 }

54

55

selectedType = "חטיבות";

\_view.setRelatedItemsType(selectedType);

56 #region Interface 57

58 /\*

1. \* Gets array of items from DB according to the type and
2. \* calls the method of the view that fills grid with the items
3. \* itemType - The type of items that will be loaded from the DB 62 \*/

63 public void loadDataByType(string itemType) 64 {

1. selectedType = itemType;
2. /\* Get an array of related items from DB
3. itemType - The type of needed items (Divisions, Courses, Rooms)
4. ShowDisabled - True or false. If we need the array including disabled items\*/
5. relatedItems = dbConn.GetRelatedItemsByType(itemType, ShowDisabled);

70

1. \_view.clearDataGrid();
2. \_view.fillDataGrid(relatedItems); 73 }

74

75 /\*

1. \* Adds new related item to the DB
2. \* newItemName - The name of the new item

78 \*/

79 public void addNewItem(string newItemName) 80 {

81 try

82 {

1. if(!string.IsNullOrEmpty(newItemName))
2. dbConn.insertRelatedItem(newItemName, selectedType); // Insert the new item

85

86

87

88

89 }

90

}

catch { };

loadDataByType(selectedType); // reload the data grid after the new item was inserted

1. // Takes a name of item and changes its status (disabled or not) at the DB and reload the data grid
2. public void changeStatusOfItem(string nameOfItem) 93 {

94 if(!string.IsNullOrEmpty(nameOfItem)) // if the instance was found 95 {

1. dbConn.changeStatusToRelatedItem(nameOfItem, selectedType);
2. loadDataByType(selectedType); // reload the data grid after the item was changed

98 }

99 }

100

|  |  |  |  |
| --- | --- | --- | --- |
| 101 |  |  | // Get data from the DB and reload the view |
| 102 |  |  | public void reload() |
| 103 |  |  | { |
| 104 |  |  | this.loadDataByType(selectedType); |
| 105 |  |  | } |
| 106 |  |  |  |
| 107 |  |  | #endregion |
| 108 |  | } |  |
| 109 | } |  |  |
| 110 |  |  |  |