|  |  |  |
| --- | --- | --- |
| 1 | using | System; |
| 2 | using | System.Collections; |
| 3 | using | System.Collections.Generic; |
| 4 | using | System.ComponentModel; |
| 5 | using | System.IO; |
| 6 | using | System.Linq; |
| 7 | using | System.Text; |
| 8 | using | System.Threading.Tasks; |
| 9 | using | System.Windows.Forms; |
| 10 | using | Nihulon2.Model; |
| 11 | using | Nihulon2.Model.DbAccess; |
| 12 | using | OfficeOpenXml; |
| 13 | using | OfficeOpenXml.Style; |
| 14 |  |  |

15 namespace Nihulon2.SupervisorsAdministration 16 {

17 /\*

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 exams 22 \*/

23 public class SupervisorsAdministration\_Controller 24 {

1. private ISupervisorsAdministrationView \_view; // instance of the view for data visualization
2. private DbConnector dbConn; // An instance of the class that provides connection to the DB

27

1. // The list of exams that are binded to the exam table
2. private SortableBindingList<Exam> examsList;
3. // The list with exams that have time overlaps
4. private SortableBindingList<Exam> examsWithOverlap; 32

33 #region Properties 34

35 /\*

36 \* Filter properties that define what data we need to get from the DB 37 \*/

38

1. // When we need to show exams for a specific period of time,
2. // this property is a date of the start of that period
3. public string dateFromFilter

42 {

1. get;
2. set;

45 }

1. // When we need to show exams for a specific period of time,
2. // this property is a date of the start of that period
3. public string dateToFilter

49 {

1. get;
2. set;

52 }

53

54 // If true, show only the exams that refer to a certain division

|  |  |  |  |
| --- | --- | --- | --- |
| 55 | public string divisionFilter | |  |
| 56 | { | |
| 57 | get; | |
| 58 | set; | |
| 59 | } | |
| 60 | // If true, show only the exams that refer to a certain course | |
| 61 | public string courseFilter | |
| 62 | { | |
| 63 | get; | |
| 64 | set; | |
| 65 | } | |
| 66 | // If true, show only the exams that refer to a certain room | |
| 67 | public string roomFilter | |
| 68 | { | |
| 69 | get; | |
| 70 | set; | |
| 71 | } | |
| 72 | // If true, show the disabled exams | |
| 73 | public bool showDisabledFilter | |
| 74 | { | |
| 75 | get; | |
| 76 | set; | |
| 77 | } | |
| 78 | // If true, show the exams that were added today | |
| 79 | public bool showNewFilter | |
| 80 | { | |
| 81 | get; | |
| 82 | set; | |
| 83 | } | |
| 84 |  | |
| 85 | // Flag specifies in which mode to show exams, All or Overlaps | | only |
| 86 | public bool showOverlaps | |  |
| 87 | { | |  |
| 88 | get; | |  |
| 89 | set; | |  |
| 90 | } | |  |
| 91 |  | |  |
| 92 | #endregion | |  |
| 93 |  | |  |
| 94 | // The constructor that gets the instance of the view | |  |
| 95 | // that will be calling to the controller | |  |
| 96 | public SupervisorsAdministration\_Controller | |  |
|  | (ISupervisorsAdministrationView view) | |  |
| 97 | { |  | |
| 98 |  | // Initializing connection to DB | |
| 99 |  | dbConn = DbConnector.Instance; | |
| 100 |  | examsList = new SortableBindingList<Exam>(); | |
| 101 |  | // Binding with the view | |
| 102 |  | \_view = view; | |
| 103 |  | \_view.setController(this); | |
| 104 |  |  | |
| 105 |  | this.loadView(); | |
| 106 |  |  | |
| 107 |  | dbConn.foundOverlaps = false; | |
| 108 | } |  | |
| 109 |  |  | |

|  |  |  |
| --- | --- | --- |
| 110 | #region Interface |  |
| 111 |  |
| 112 | // Reload all comboboxes and fill the table |
| 113 | public void reload() |
| 114 | { |
| 115 | reloadFiltersComboBoxes(); |
| 116 | fillTable(); |
| 117 | } |
| 118 |  |
| 119 | // Get the exams from the DB and bind the list of exams with the | table |
| 120 | at the view  public void fillTable() |  |
| 121 | { |  |
| 122 | List<Exam> list; |  |
| 123 |  |  |
| 124 | // Show all exams according to the filters |  |
| 125 | if (!showOverlaps) |  |
| 126 | { |  |
| 127 | // Get exams form DB according to filters |  |
| 128 | list = dbConn.getExams(dateFromFilter, dateToFilter, |  |

divisionFilter,

1. courseFilter, roomFilter, showDisabledFilter, showNewFilter);
2. list.Reverse();
3. // Convert the list of exams to the Binding list
4. examsList = new SortableBindingList<Exam>(list);
5. // Ask the view to bind its table with controller's list of exams
6. \_view.bindExamsWithTable(ref examsList);

135 }

1. // Show the exams that have time overlaps
2. else

138 {

1. // Get exams with time overlaps
2. list = dbConn.getExamsWithOverlaps();
3. // Convert the list of exams to the Binding list
4. examsList = new SortableBindingList<Exam>(list);
5. \_view.bindExamsWithTable(ref examsList);

144 }

145 }

146

1. // Add a new exam to the DB and reload the table
2. public void addExam(Exam newExam)

149 {

1. // Insert the new exam
2. dbConn.insertExam(newExam);
3. // Fill the table with exams
4. fillTable();

154 }

155

1. // Delete an exam from the DB
2. public void deleteExam(int examId)

158 {

159 dbConn.removeExam(examId);

160 }

161

1. // Save changed exam to the DB and reload the table
2. public void changeExam(Exam changedExam)

164 {

1. dbConn.updateExam(changedExam);
2. // Check time overlaps if the flag "Show overlaps" is true
3. if(this.showOverlaps == true)
4. dbConn.markExamsWithOverlap();
5. fillTable();

170 }

171

1. // Takes a name of Excel file and create the file
2. // with exam table
3. public void createExcel(string excelName)

175 {

1. string filePath;
2. if(excelName != "")

178 {

1. // the path to the excel file
2. filePath = @"./Excel files/" + excelName + ".xlsx"; 181
3. // Creating the directory for Excel files if doesn't exists
4. if (!Directory.Exists(@"./Excel files/"))
5. Directory.CreateDirectory(@"./Excel files/"); 185
6. // Create an instance of the excelPackage, create the file and fill it with the exams
7. using (ExcelPackage excel = new ExcelPackage())

188 {

1. // create a workSheet
2. excel.Workbook.Worksheets.Add("Worksheet1");
3. // get the created workSheet
4. ExcelWorksheet excelWorksheet = excel.Workbook.Worksheets ["Worksheet1"];
5. // create the row of headers
6. List<string[]> headerRow = new List<string[]>()

196 {

197

new string[] { "זמן

סיום" ",תוספת

שעת", "

שעת

" ",מגמה/קורס" ",קבוצה" ",מקצוע" ",חדר" ",התייצבות

198

משגיח/ה" ",חטיבה }, "זמן" ",הערה

};

בחינה" ",שם

" ",בוטל" ",מס" ",תאריך

1. // Insert header row data
2. excelWorksheet.Cells["A1"].LoadFromArrays(headerRow);
3. // Build the data for inserting from the exams at the table
4. List<string[]> examsData = this.getExamsForExcel();
5. excelWorksheet.Cells["A2"].LoadFromArrays(examsData); 204
6. // format the workSheet
7. this.formatExcel(excelWorksheet, examsData.Count); 207
8. FileInfo excelFile = new FileInfo(filePath);
9. excel.SaveAs(excelFile);

210 }

211 }

212 }

213

1. // Load exams from Excel file
2. // and save the changes into the DB
3. public void loadFromExcel(string path)

217 {

1. // Create an instance of the excelPackage from the Excel file
2. using (ExcelPackage excel = new ExcelPackage(new FileInfo(path)))

220 {

1. int i;
2. // Get worksheet
3. ExcelWorksheet workSheet = excel.Workbook.Worksheets[1]; 224
4. // Read the row of headers from the excel
5. char ch = 'A'; // Start from A 227
6. // Fill the array with headers
7. string[] headers = new string[14];
8. for(i = 1; i <= headers.Length; i++)

231 {

1. if (workSheet.Cells[$"{(char)(ch+i-1)}1"].Value != null)
2. headers[i-1] = workSheet.Cells[$"{(char)(ch + i - 1)} 1"].Value.ToString();
3. else
4. headers[i-1] = "";

236 }

237

1. // Check if the headers at correct order and format
2. if (this.checkExcelFormat(headers))

240 {

1. // Save the exams from that worksheet to the DB
2. this.loadExamsFromExcelToDB(workSheet);

243 }

244 }

245 }

246

1. // Finds exams with time overlaps
2. public void checkOverlaps()

249 {

1. dbConn.markExamsWithOverlap();
2. this.fillTable();

252 }

253

254 #endregion 255

256

1. #region Private methods for internal use
2. // Get names of related items from DB and fill the combo boxes of the filters in the view
3. private void reloadFiltersComboBoxes()

260 {

261 string[] divisions, courses, rooms; 262

1. // Get the names of related items from the DB
2. divisions = dbConn.getRelatedItemsNamesByType("חטיבות");
3. courses = dbConn.getRelatedItemsNamesByType("מגמות");
4. rooms = dbConn.getRelatedItemsNamesByType("חדרים"); 267
5. // Clear combo boxes
6. \_view.clearComboBoxes(); 270
7. // Fill the combo boxes with names of related items
8. \_view.fillDivisionsCbo(divisions);
9. \_view.fillCoursesCbo(courses);
10. \_view.fillRoomsCbo(rooms);

275 }

276

1. // Set default values for the properties
2. private void setDefaultFilters()

279 {

1. this.dateFromFilter = "";
2. this.dateToFilter = "";
3. this.divisionFilter = "הכול";
4. this.courseFilter = "הכול";
5. this.roomFilter = "הכול";
6. this.showDisabledFilter = false;

286 }

287

1. // Builds rows with exams for inserting into the excel file
2. private List<string[]> getExamsForExcel()

290 {

291 List<string[]> data = new List<string[]>(); 292

1. // Go through the exams at the table and fill the rows data
2. foreach(Exam exam in examsList)

295 {

296 string[] row = new string[14]; 297

1. row[0] = exam.hasExtraTime ? "יש" : "";
2. row[1] = exam.EndingTime;
3. row[2] = exam.StartTime;
4. row[3] = exam.room;
5. row[4] = exam.DisciplineName;
6. row[5] = exam.GroupName;
7. row[6] = exam.course;
8. row[7] = exam.division;
9. row[8] = exam.SupervisorName;
10. row[9] = exam.Date;
11. row[10] = exam.Id.ToString();
12. row[11] = exam.isCanceled ? "בוטל" : "";
13. row[12] = exam.Comments; 311

312 // Calculate difference between the start time and the ending time

|  |  |
| --- | --- |
| 313 | // for getting the time scale of the exam |
| 314 | DateTime fromTime = DateTime.Parse(exam.StartTime); |
| 315 | DateTime toTime = DateTime.Parse(exam.EndingTime); |
| 316 | TimeSpan hours = (toTime - fromTime); |
| 317 | row[13] = hours.ToString(); |
| 318 |  |
| 319 | data.Add(row); |

|  |  |  |
| --- | --- | --- |
| 320 | } |  |
| 321 | return data; |
| 322 | } |
| 323 |  |
| 324 | // Gets workSheet of Excel with exams and formats its cells |
| 325 | private void formatExcel(ExcelWorksheet excelWorksheet, int | numRows) |
| 326 | { |  |
| 327 | int i; |  |
| 328 |  |  |
| 329 | excelWorksheet.View.RightToLeft = true; |  |

330

1. // Set width for the columns
2. excelWorksheet.Column(1).Width = 7;
3. excelWorksheet.Column(2).Width = 10;
4. excelWorksheet.Column(3).Width = 10;
5. excelWorksheet.Column(4).Width = 12;
6. excelWorksheet.Column(5).Width = 15;
7. excelWorksheet.Column(6).Width = 10;
8. excelWorksheet.Column(7).Width = 25;
9. excelWorksheet.Column(8).Width = 10;
10. excelWorksheet.Column(9).Width = 17;
11. excelWorksheet.Column(10).Width = 13;
12. excelWorksheet.Column(11).Hidden = true;
13. excelWorksheet.Column(12).Width = 7;
14. excelWorksheet.Column(13).Width = 25;
15. excelWorksheet.Column(14).Width = 10; 346

347 // Set wrap text

348 for (i = 1; i <= 14; i++)

349 excelWorksheet.Column(i).Style.WrapText = true; 350

1. // Set border style
2. excelWorksheet.Cells[$"A1:N{numRows + 1}"].Style.Border.Left.Style

= OfficeOpenXml.Style.ExcelBorderStyle.Thin;

1. excelWorksheet.Cells[$"A1:N{numRows + 1}"].Style.Border.Right.Style = OfficeOpenXml.Style.ExcelBorderStyle.Thin;
2. excelWorksheet.Cells[$"A1:N{numRows + 1}"].Style.Border.Top.Style

= OfficeOpenXml.Style.ExcelBorderStyle.Thin;

1. excelWorksheet.Cells[$"A1:N{numRows + 1}"].Style.Border.Bottom.Style = OfficeOpenXml.Style.ExcelBorderStyle.Thin;
2. excelWorksheet.Cells[$"A1:N{numRows + 1}"].Style.Border.BorderAround (OfficeOpenXml.Style.ExcelBorderStyle.Thick);
3. // Paint yellow the empty cells at the column with names of supervisors

|  |  |  |
| --- | --- | --- |
| 358 | for | (i = 2; i <= numRows + 1; i++) |
| 359 | { |  |
| 360 |  | if(excelWorksheet.Cells[$"I{i}"].Value == null) |
| 361 |  | { |

1. excelWorksheet.Cells[$"I{i}"].Style.Fill.PatternType = OfficeOpenXml.Style.ExcelFillStyle.Solid;
2. excelWorksheet.Cells[$"I

{i}"].Style.Fill.BackgroundColor.SetColor (System.Drawing.Color.Yellow);

|  |  |
| --- | --- |
| 364 | } |
| 365 | } |
| 366 | } |
| 367 |  |
| 368 | // Checks format of the excel file |
| 369 | private bool checkExcelFormat(string[] headers) |
| 370 | { |
| 371 | string[] patternHeaders = { "זמן תוספת", "סיום שעת", " שעת |

",קבוצה" ",מקצוע" ",חדר" ",התייצבות

372

משגיח/ה" ",חטיבה" ",מגמה/קורס" }; "זמן" ",הערה" ",בוטל"

בחינה" ",שם

",מס" ",תאריך

373 // Check the header row if the columns at correct order 374 for(int i = 0; i < 14; i++)

375 {

1. if (headers[i] != patternHeaders[i])
2. return false;

378 }

379 return true;

380 }

381

1. // Sets default filters and fill the table with exams
2. private void loadView()

384 {

1. // Set default filters
2. setDefaultFilters();
3. // Get names of related items and fill the combo boxes of the filters in the view
4. reloadFiltersComboBoxes(); 389
5. // Bind the event handler that catch time overlaps with the processing method of the view
6. dbConn.onOverlapsStateChanged += \_view.showOverlapsWarning; 392
7. // Fill the table with exams
8. fillTable();

395 }

396

1. // Takes an Excel worksheet with exams data as a parameter
2. // and saves the exams from that worksheet to the DB
3. private void loadExamsFromExcelToDB(ExcelWorksheet workSheet)

400 {

401 // Array for saving id, supervisor name and comments

402 string[] examData = new string[3];

403 // Go through all rows at the excel file and save the changes into the DB

|  |  |  |
| --- | --- | --- |
| 404 | for | (int i = 2; i <= workSheet.Dimension.End.Row; i++) |
| 405 | { |  |
| 406 |  | // Get data from the excel |
| 407 |  | // Id |
| 408 |  | if (workSheet.Cells[$"K{i}"].Value != null) |

409 examData[0] = workSheet.Cells[$"K{i}"].Value.ToString();

410 else

411 examData[0] = "";

|  |  |  |
| --- | --- | --- |
| 412 | // | Supervisor name |
| 413 | if | (workSheet.Cells[$"I{i}"].Value != null) |

414 examData[1] = workSheet.Cells[$"I{i}"].Value.ToString();

|  |  |  |  |
| --- | --- | --- | --- |
| 415 |  |  | else |
| 416 |  |  | examData[1] = ""; |
| 417 |  |  | // Comments |
| 418 |  |  | if (workSheet.Cells[$"M{i}"].Value != null) |
| 419 |  |  | examData[2] = workSheet.Cells[$"M{i}"].Value.ToString(); |
| 420 |  |  | else |
| 421 |  |  | examData[2] = ""; |
| 422 |  |  |  |
| 423 |  |  | // Save the changes into the DB |
| 424 |  |  | dbConn.updateExamFromExcel(examData); |
| 425 |  |  | } |
| 426 |  |  | } |
| 427 |  |  |  |
| 428 |  |  | #endregion |
| 429 |  |  |  |
| 430 |  | } |  |
| 431 | } |  |  |
| 432 |  |  |  |