Sessions & Attendees

# Document Control

## Record of Changes

|  |  |  |
| --- | --- | --- |
| **Author** | **Date** | **Description** |
| **Mohamed Salah** | 4/6/2018 |  |
|  |  |  |

## Distribution

|  |  |
| --- | --- |
| **Name** | **Designation** |
|  |  |
|  |  |

## Approvals

|  |  |  |
| --- | --- | --- |
| **Name** | **Designation** | **Signature** |
|  |  |  |
|  |  |  |

## Related Documentation

|  |  |  |
| --- | --- | --- |
| **Title** | **Version &/or Date** | **Description of Relationship** |
|  |  |  |
|  |  |  |

Contents

[1 Document Control 1](#_Toc516328472)

[1.1 Record of Changes 1](#_Toc516328473)

[1.2 Distribution 1](#_Toc516328474)

[1.3 Approvals 1](#_Toc516328475)

[1.4 Related Documentation 1](#_Toc516328476)

[2 Introduction 3](#_Toc516328477)

[2.1 The problem descritopn 3](#_Toc516328478)

[2.2 Objective: 3](#_Toc516328479)

[3 List of Technologies and Patterns 5](#_Toc516328480)

[3.1 I used the following technologies 5](#_Toc516328481)

[3.2 Patterns 5](#_Toc516328482)

[4 System Modules 6](#_Toc516328483)

[5 Class Diagram 7](#_Toc516328484)

[5.1 Sesion.Model 7](#_Toc516328485)

[5.2 Session.Repository 8](#_Toc516328486)

[5.3 Session.Service 9](#_Toc516328487)

[6 Database 10](#_Toc516328488)

[7 UI Views 11](#_Toc516328489)

[8 Missing due to time limitation 14](#_Toc516328490)

# Introduction

This documentation is descripting the work Project May 2018May.

Assume you are in a college and there are many sessions to be attended by attendees, and you need to know who attended it and want to create some reports to grant certificates to the attendees based on their attendance.

## The problem descritopn

Develop the following 2 pages:

Sessions page:

* • page to insert the sessions with all the session details required in addition to a grid to show the inserted sessions. In this page, you will create two sections:
* 1. To insert the sessions with all the session details required.
* 2. Show sessions in a grid.

Upload Attendee CSV Page:

* • A page to upload a CSV file that contains the attendees list with sessions they attended in addition to a grid that shows the imported data of the attendees and their attended sessions. In this page, you will create two sections:
* 1. Upload CSV File
* 2. Show attendees in a grid with a data filter above the grid
* • Upload CSV File:
* 1. Mandatory columns in the CSV are (Attendee ID, Session Time ID), validation on the CSV file
* 2. Attendee ID→generate attendee IDs - static numbers (1,2,3,4,5,6,7,……)
  1. 3. Session Time ID Comes from sessions page o Other than (Attendee ID, Session Time ID) columns are dynamic, the user can use any numbers of columns or any column names.

## Objective:

1. Keep a clean separation of concerns.
2. Make unit testing and integration testing easy.
3. A complete example on Repository and Unit of Work with IoC and Unit Testing.

# List of Technologies and Patterns

## I used the following technologies

1. MVC 5
2. Entity Framework 6.0 (Approche Code First)
3. Sql Server (LocalDB)
4. Visual Studio 2015
5. Jquery

## Patterns

1. MVC Pattern
2. Repositorty pattern: To create an abstract data access layer for the application which allows to centralise all data access logic in one place. With generic feature, we can reduce the amount of code we need for common scenarios
3. UnitOfWork pattern:  To combine a set of interactions and commit them at once using a transaction.
4. Dependency Injection pattern: Basically, instead of having your objects creating a dependency or asking a factory object to make one for them, you pass the needed dependencies in to the constructor or via property setters, and you make it somebody else's problem (an object further up the dependency graph, or a dependency injector that builds the dependency graph)
5. IOC desing pattern: is used to increase modularity of the program and make it extensible

# System Modules

There are 5 modules in the system

1. Session: is the web MVC project that contains UI and the controlers of the project that take the action from the UI and parameters form model.
2. Session.Model: is data access layer of the system that contains Entity module
3. Session.Repository: abstraction layer provides an in-memory like collection interface for accessing domain objects (AttendeeFullDataRepository, SessionAttendee …). Therefore, as far as the consuming component is concerned, it uses the repository just like a collection when working with Domain objects.
4. Session.Service: is the BLL (Business logic layer) of the system,
5. Session.Tests: (for unit testing, mocking …etc)

Session.Repository

Database

Unit Of work

Repository

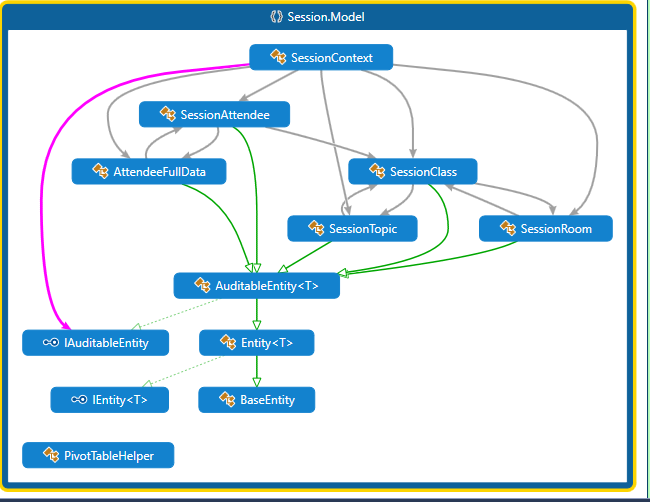
DBContext

Session.Service (BLL)

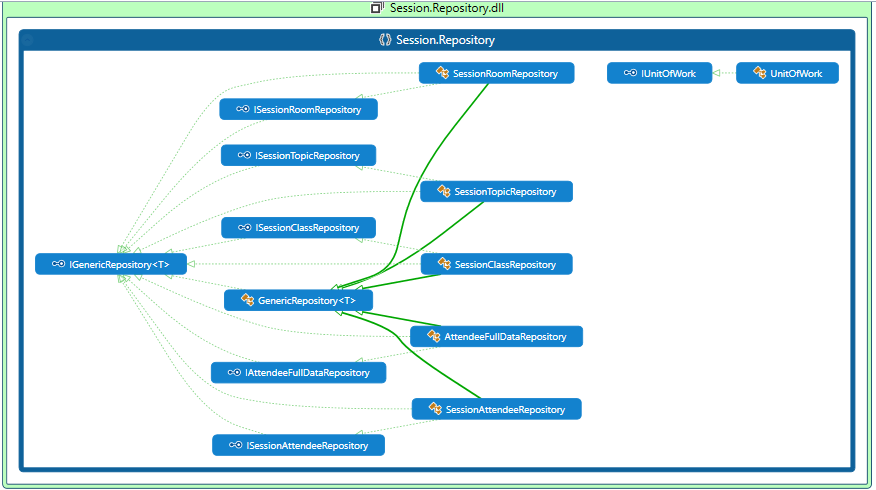
Session

# Class Diagram

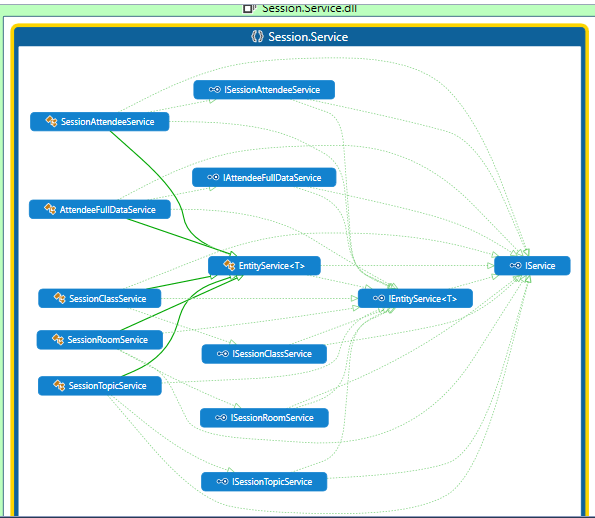
## Sesion.Model



## Session.Repository

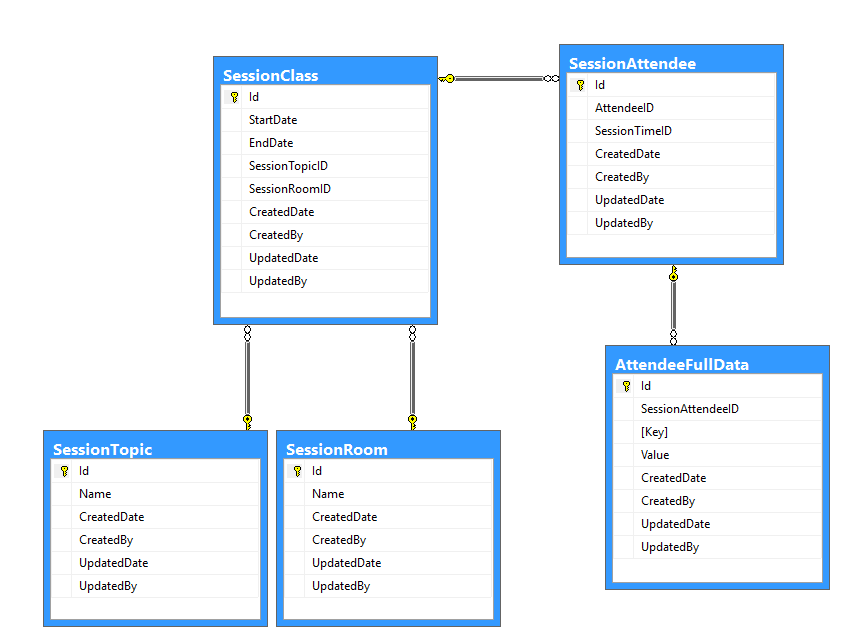


## Session.Service



# Database

Databas Cosists of 5 Tables as the following



|  |  |
| --- | --- |
| Table Name | Usage |
| SessionTopic | Contais topics names {Oracel, WebApi, ...} |
| SessionRoom | Contais Room names {Hall, Conference Court, Room C, …} |
| SessionClass | Contains the session data { Topic , Room , from data , to date} |
| SessionAttendee | Contains the data between the attendee and classes |
| AttendeeFullData | Key value table contains the dynamic data for full attendee details. |

**Note**: AttendeeFullData is key Value table to deal with the idea of dynamic columns. And on run time

Code convert it to columns instead of rows

For example

|  |  |  |  |
| --- | --- | --- | --- |
| Id | SessionAttendeeID | Key | Value |
| 273 | 44 | First Name | Mahmoud Attendee1 |
| 274 | 44 | Last Name | Elmasry1 |
| 275 | 44 | tITlE | President |
| 276 | 44 | Company | imahmoud |

On run time it convert to

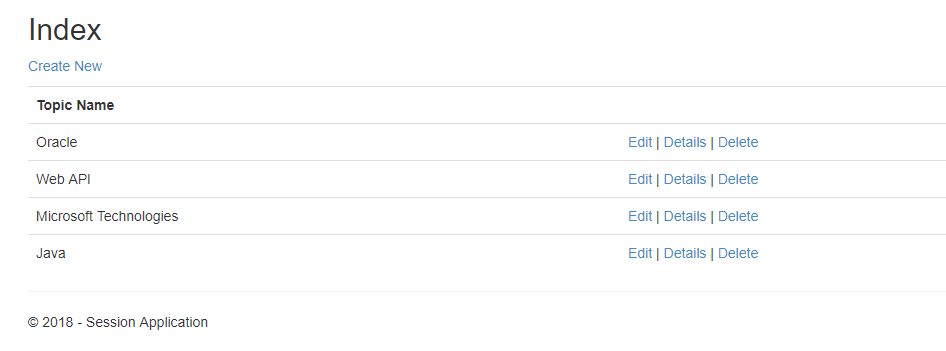
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SessionAttendeeID | First Name | Last Name | tITlE | Company |
| 44 | Mahmoud Attendee1 | Elmasry1 | President | imahmoud |

In addition, join with SessionAttendee so that grid of attendee can shown with full data.

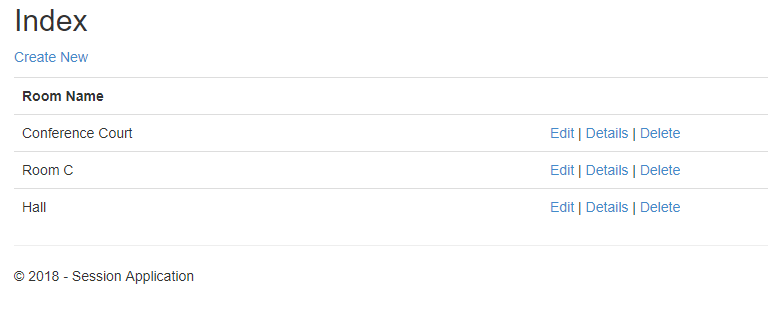
# UI Views

There are five views in the demo app each as following

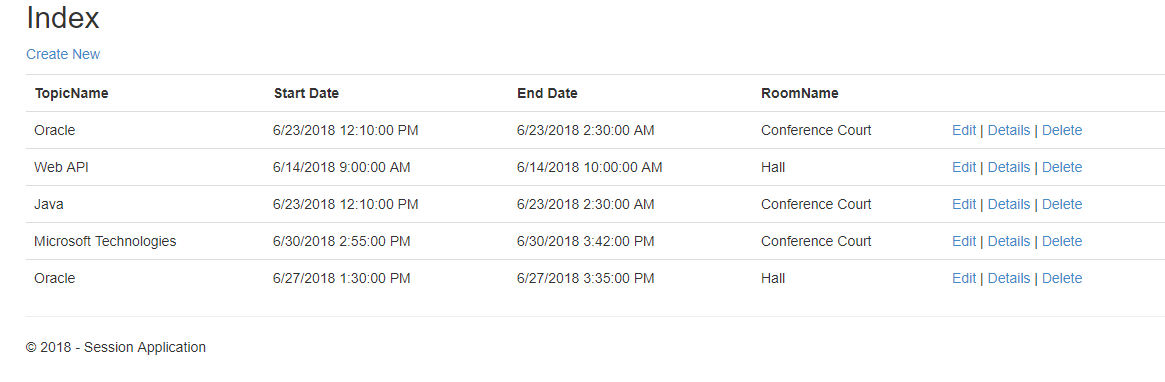
1. *Topic*  
   contains the list of app topic that can used to make session with avialvility to view details, Edit, add and delete.



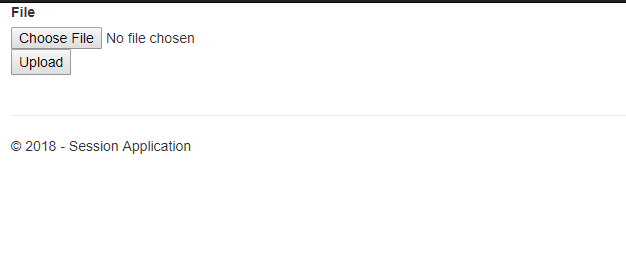
1. *Room*  
   contains the list of app room that can used to make session with avialvility to view details, Edit, add and delete.



1. *Session*  
   contais list of the app sessions with avialvility to view details, Edit, add and delete.

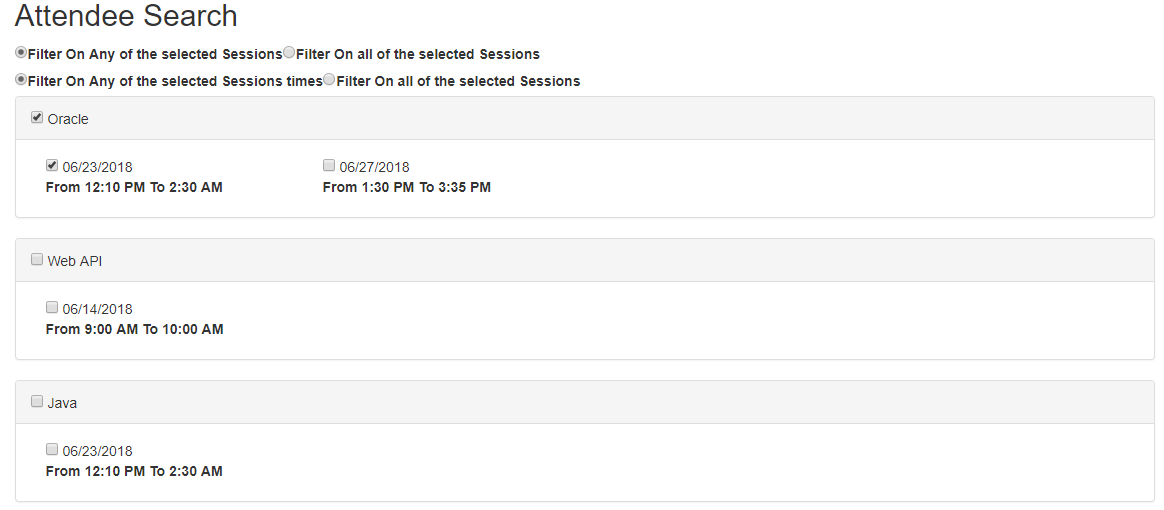


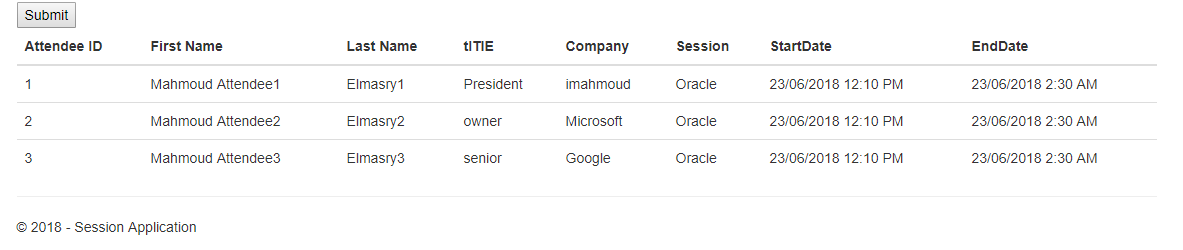
1. *Upload* *Attnedee*



1. *Attendee* *Grid*

Dasdas





# Missing due to time limitation

Due to the time delivery limitation, I was hopping to add the following feature to the demo

1. Logging module: so, that I can track the system for any issue happened or results not sent to the users   
   and in that module, I will use log4net or make custom loging in the database
2. Localization module: so, that I can avoid make any hard-codded strings and also support multi-languages