ADVANCED INTERNET SERVICES TERM PROJECT REAL TIME GAMING

For Advanced Internet Services term project I have developed a online real-time memory game which can be played with 2 players consecutively via web browser. Players can log in with their name and create sessions. These players will be waited on the game pool. When another user log in the game and create session then game engine will matches these two players who are currently not playing a game and starts a new match. Besides that they can also send chat messages to each other instantly.

1. USED TECHONOLOGIES

In this project I have used Microsoft Visual Studio as IDE. For providing real time functionalities I have used Javascript and ASP.NET SignalR technology. ASP.NET SignalR is a library for .NET that adds real-time / bidirectional communication to your web application. Also i have used cardflip css and handlebars javascript library for user interface representation.

2. ABOUT SIGNALR

ASP.NET SignalR is a new library for ASP.NET developers that makes it incredibly simple to add real-time web functionality to your applications. What is "real-time web" functionality? It's the ability to have your server-side code push content to the connected clients as it happens, in real-time.

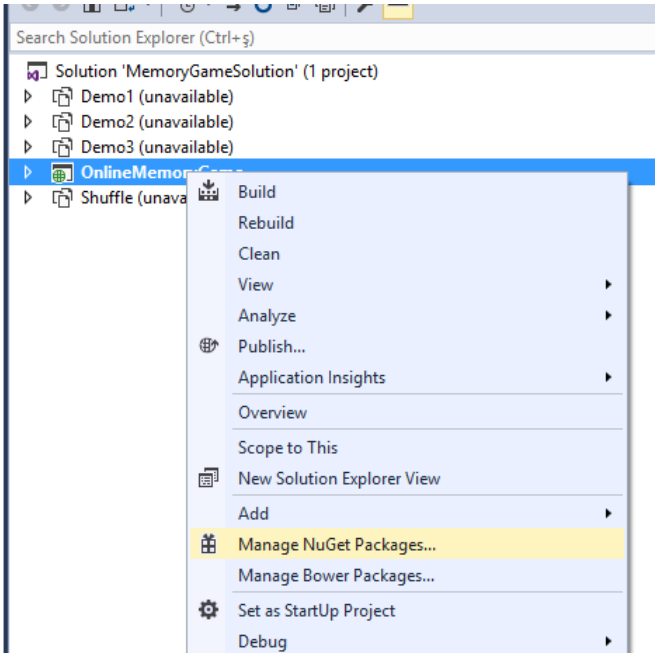
You may have heard of WebSockets, a new HTML5 API that enables bi-directional communication between the browser and server. SignalR will use WebSockets under the covers when it's available, and gracefully fallback to other techniques and technologies when it isn't, while your application code stays the same.

SignalR also provides a very simple, high-level API for doing server to client RPC (call JavaScript functions in your clients' browsers from server-side .NET code) in your ASP.NET application, as well as adding useful hooks for connection management, e.g. connect/disconnect events, grouping connections, authorization.

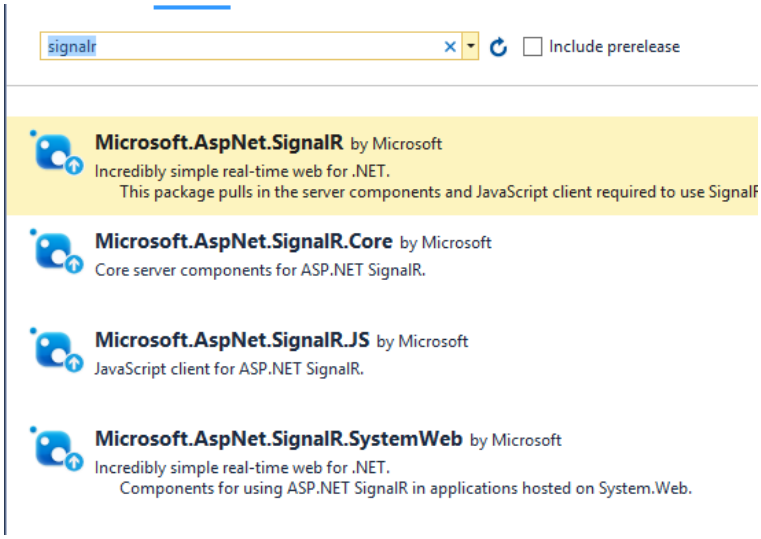
SignalR can be used to add any sort of "real-time" web functionality to your ASP.NET application. While chat is often used as an example, you can do a whole lot more. Any time a user refreshes a web page to see new data, or the page implements Ajax long polling to retrieve new data, is candidate for using SignalR.

It also enables completely new types of applications, that require high frequency updates from the server, e.g. real-time gaming.

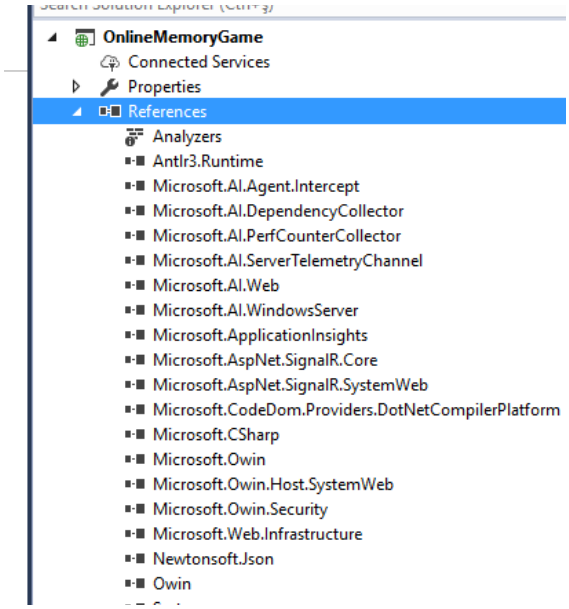
How to add this framework to your project?



Right click on your project and select manage NuGet Packages. type signalr and download and install the listed packages.



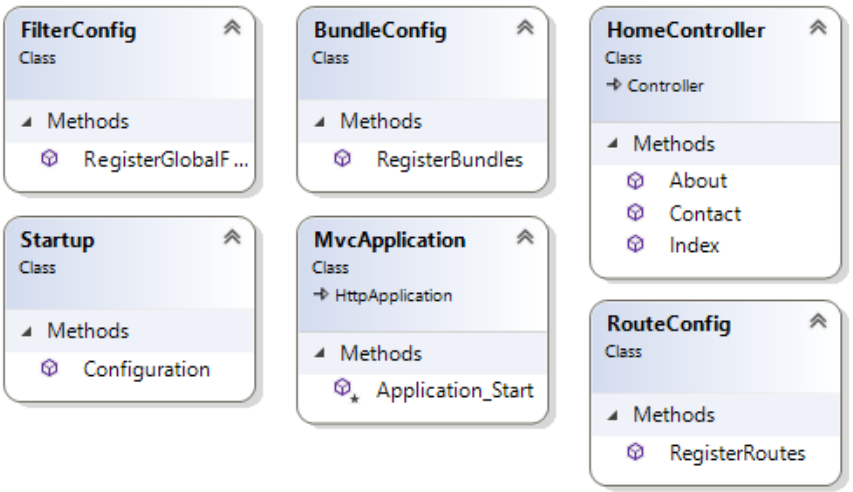
As you see below NuGet will install these required packages into our project solution then we will be able to use these libraries in our project.



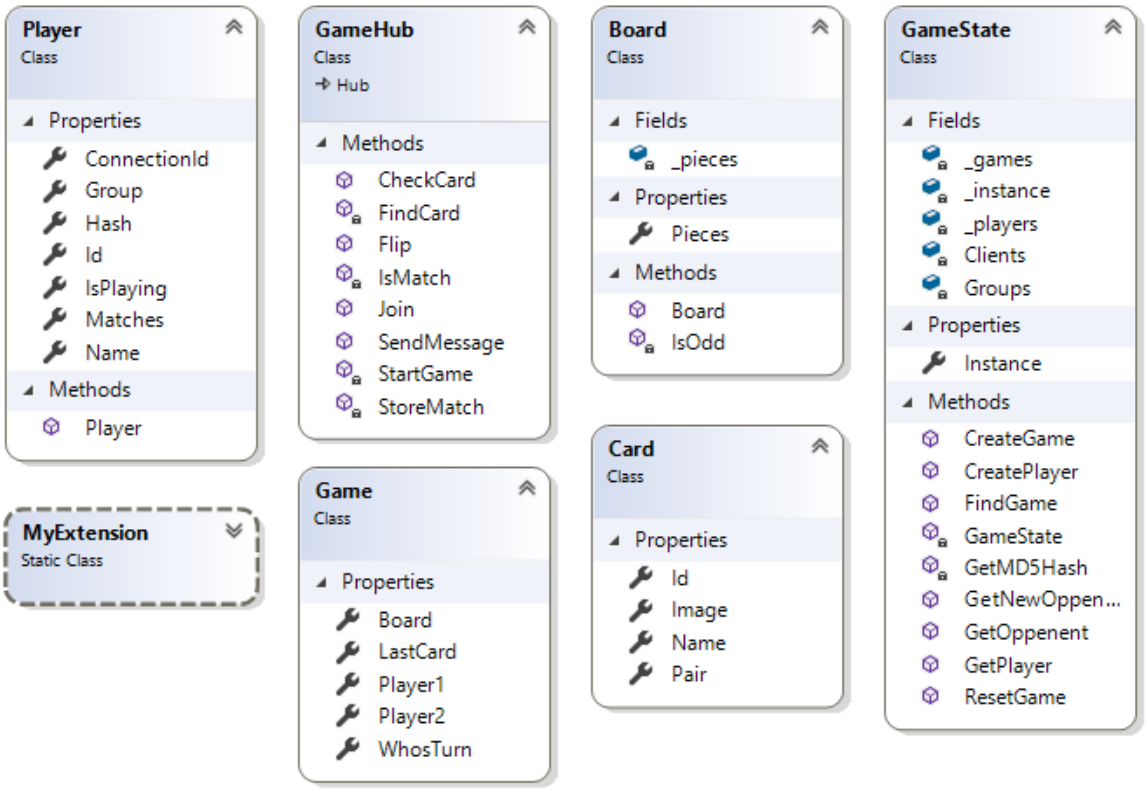
3. DESIGNING BACK-END OF REAL-TIME MEMORY GAME

I have implemented a card matching game as a memory game which can be played as real-time. In visual studio I have created and ASP.NET Web Application. In this solution I have created necessarily classes which we will need while creating game. Let's see the class diagram and give detail about it.

First part here is classes which are created automatically for asp.net web application. We will not go in details about these classes. They are used for start up, running, registering configuration files, routing web pages, controlling models and views, configuration security etc.



Second part here is our game project code classes which are more important for us. We will go in detail for each class. Let's have a look at class diagram.



Player.cs:

Player.cs class is prepared for the users who log in the web game and plays match between other users. It has a ConnectionId which one is unique for every log in from every user. A Group property for grouping players who are currently play a match with each other. A Hash property to encrypt user name and store it in the session. Id for providing uniqueness of palyers as a GUID. IsPlaying property is used for player plays a match or not currently. Matches property is a list of matched card id numbers.

Card.cs:

Card.cs class is created to keep cards details. It has a Id property to be unique in other cards as a GUID. Image property is used image name which is kept for the storage of the computer directory location. A Pair property is used to keep Id of the same card picture.

Board.cs:

Board.cs class is prepared to create a game board with card pieces for each game. Normally the cards in every match will start at the same position and it will be easy to find the same card

while playing more and more. So task of this board class is to shuffle cards for every match start.

Game.cs:

Game.cs class creates two players are Player1 and Player2 also a Board and a Last Card for pairing with the same one and a WhosTurn flag to hold that who's turn currently.

GameState.cs:

GameState.cs class is our basic game engine for managing all games and all players, matching players, creating and resetting games etc. This class is prepared based on Singleton Design Pattern. Because in our application we will just need one object in the memory. In this class's construction we get the HubContext as a parameter for getting all clients and groups of these clients. This class will be used in GameHub which is used to online communication between players and games. Let's have a look to implemented methods in this class.

```
public Player CreatePlayer(string userName)
```

Creates a new player via using Player.cs constructer.

```
public Player GetPlayer(string userName)
```

Gets the player from player list using by userName parameter.

```
public Player GetNewOppenent(Player player)
```

Gets a player who is currently not playing a game other than requesting user.

```
public Player GetOppenent(Player player, Game game)
```

Using game parameter this method returns the player who is on the game other than requesting user.

```
public Game CreateGame(Player player1, Player player2)
```

This method creates a game which includes two players and a board. For new game this method creates a group and set the players' group property to this group id.

```
public Game FindGame(Player player, out Player opponent)
```

Using player parameter this method finds the player's exist game and finds the player's opponent in this game.

```
public void ResetGame(Game game)
```

This method deletes the parameter gaming group and deletes the players who are included in this group. Also the players and the game will be deleted from players and game lists.

GameHub.cs:

GameHub.cs class is our another basic game engine for communicating players and managing online matches. This class manage sessions for players logging in the game, starts games, builds game boards, flips cards, pairs cards or not, sending chat messages etc.

```
public bool Join(string userName)
```

Using GameState.GetPlayer method this method queries the player list and if there is no player creates one with this user name. Besides that it calls a client JavaScript function to update its view.

```
private bool StartGame(Player player)
```

Firstly using GameState.FindGame it queries that a game exists or not for the player. If there is an existing game then it calls a javascript function for the group of this game to build their game board. If there is no game and there is no available player then it sends the player to waiting list. If there is an available player who is currently available for a game then it creates a new game and build this two players' game board.

```
public bool Flip(string cardName)
```

Flip method is called by a client. it controls the turn of the game that equals the caller client or not. If turn is on the caller client then server side request a javascript flip function to visual flip effect on the players.

```
public bool CheckCard(string cardName)
```

CheckCard method firstly finds the players and the game. Then controls the turn of the game. After that determine the flip sequence that first or second flip. If it is first flip then sets the lastCard property to flipped card. If the flipping card is the second one then controls it matches or not via IsMatch method. If it is matched then calls showMatch javascript funtion of the group to visual effect. If the caller player's match count is higher than half of the card pairs then it calls the winner javascript function of the game group.

```
public void SendMessage(string userName, string message)
```

This method is used for clients to send chat messages to each other.

```
private void StoreMatch(Player player, Card card)
```

This method is used store the for pair cards of the player.

```
private bool IsMatch(Game game, Card card)
```

Checks the parameter card is the same card with the game's last card or not.

```
private Card FindCard(Game game, string cardName)
```

Finds the card of the parameter game with parameter card name.

4. DESIGNING FRONT-END OF REAL-TIME MEMORY GAME

We have completed the back-end part of our project. Let's examine the frond-end part of the game project. As a ubiquitous standart web technology we have used HTML, CSS and Javascript. For UI controls we used HTML. For styling these HTML tags we have used bootstrap and for programming these controls we have used JQuery.

For the card pieces we used handlebars javascript library to get a table like view. Handlebars has got templates that can be used like variables in loops.

Setting Up a client connection with javascript by the client:

```
var gameHub = $.connection.gameHub;
```

Calling a sample server-side method with javascript by the client:

gameHub.server.sendMessage(userName, message);

Hub Start Function:

When hub connection starts then this start function is called. We can code required initial statements here:

```
$.connection.hub.start().done(function () {

            $('#join').removeAttr('disabled');

        });
```

A sample client javascript function for server-side calls:

```
gameHub.client.waitingList = function () {

      $('#alert').html("At this time there is not an oppenent. As soon as one joins game will begin.");

        };
```

From server-side calling a client javascript function can be done easily:

```
Clients.Group(player.Group).buildBoard(game);
```

We have some javascript functions:

Log-in button:

Gets the user information and sends it to GameHub class via using RPC (Remote Procedure Call)

addMessage:

Gets a message from a client and adds this message to all client html pages.

playerJoined:

It is called from server to client when a player joined the game.

waitingList:

It is called from server to client when a player joined the game but there is no avaliable opponent.

buildBoard:

It is called from server to client when a player joined the game and there is already a player who is in waiting list. So for these to clients of the group are called this method to create their board to start a new game.

flipCard:

This method is called from server to client to flip a card from back to front in a motion view using by cardflip.css

resetFlip:

This method is called from server to client when user tries to find a match by flipping the second card but cannot find the match. So the two card will be flipped to back in a delayed 1,5 s time interval.

showMatch:

This method is called from server to client when a user finds the card match. Front of the cards will be stable. And displays a message that the user finds a match.

Winner:

This method is called from server to client when a user finds the card match and also this is the winning move. Front of the cards will be stable and will display a win-loss message to 2 players.