INFS3204/7204 Assignment 4 - NBA League REST API

The goal of this assignment is to explore REST API development. All assignments will have to be developed with Microsoft Visual Studio 2015 using **C#** as the programming language. No other languages will be accepted. This assignment contributes to 10% of your overall grade. **You must present this assignment to your lab tutor during your scheduled lab session in week 12 (20/10/2016 to 21/10/2016). No late demonstration is allowed.** You also need to submit your code on blackboard before 11:59pm Sunday 23/10/2016.

Preparation

Before attempting to do this assignment, you should have a basic knowledge of how to use Microsoft Visual Studio 2015 to create ASP.NET Web Applications and Web Services, as well as a basic knowledge of C# programming**. Please note that you need to choose .NET Framework 4.5/4.6 for your assignments.** You also need to have Windows 7 (or later versions) installed on your computer.

Notes: Please be advised that you need to meet the basic requirements for this assignment, which are mentioned in this document. However, you may try to design and implement your work more interestingly and comprehensively.

You are responsible for all exception handling in both service and client sides in all required functions. During the demonstration, if your application crashes because of unhandled exception (either service or client side), you will lose marks for that function. For input validation, only the data types listed are accepted, no other types are allowed.

NBA League Web API

You need to create the player class in your project as follows:

- **Player (2 Marks)**
  - String **Registration_ID**
  - String **Player_name**
  - String **Team_name**
  - DateTime **Date_of_birth**

You need to provide all the following RESTful APIs:

- GetAllPlayers. **(2 Marks)**

  Use RESTful approach to implement this operation on server side that supports Http GET method (1 mark), and use Ajax to call this operation from a client side web page (1 mark).

  This operation reads information about all players from players.txt (Download from Blackboard) and returns a list of players.

  All Players

  - p1111, Kobe Bryant, Los Angeles Lakers, 1978-08-23
  - p2222, Tim Duncan, San Antonio Spurs, 1976-04-25
  - p3333, LeBron James, Cleveland Cavaliers, 1984-12-30

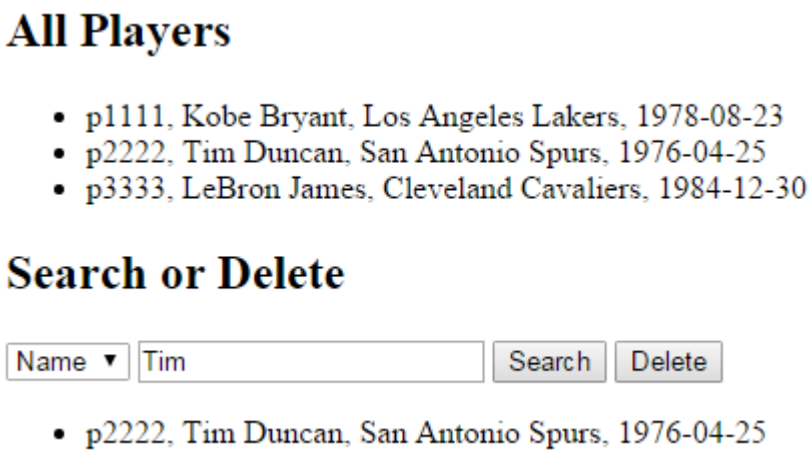
- GetPlayerInfo. **(2 Marks)**

  Use RESTful approach to implement this operation on server side that supports Http GET method (1 mark), and use Ajax to call this operation on client side (1 mark).

  This operation receives a field and a corresponding value of such field, and returns a list of players that satisfy this condition. For example, when the field is ID and the value is p2222, it displays all the players that match the condition (in this case, the information about Tim Duncan).

  Note that: The supported fields should include PlayerID and PlayerFullName (Using a Dropdownlist in a web form). Input validation is required, and the input

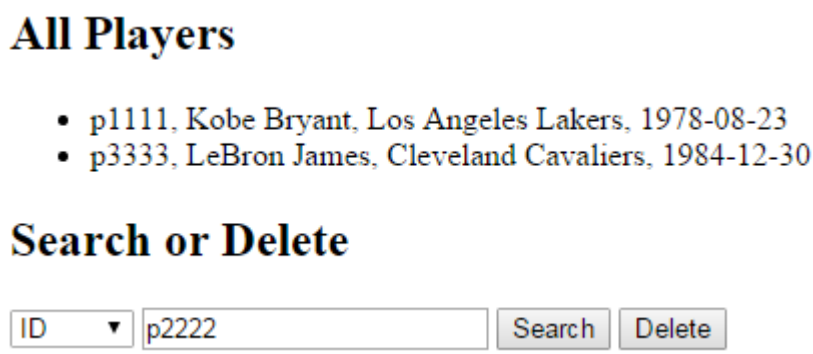
cannot be null. For Name, you need to support both upper and lower cases, and you also need to support partial match.



- DeletePlayer. **(2 Marks)**

  Use RESTful approach to implement this operation on server side that supports Http DELETE method (1 mark), and use Ajax to call this operation on client side (1 mark).

  This operation receives a field and a corresponding value of such field. For example, the field is ID, and the value is p2222. Then, it deletes all the players that match the condition, and saves the rest of the players into the players.txt. The supported fields should include PlayerID and PlayerName. If the operation fails due to deletion of an inexistent player, it should report a NotFound error message. Note that, once you delete a player, the list of players needs to be updated on the web page.



- PlayerRegistration. **(2 Mark)**

  Use RESTful approach to implement this operation on server side that supports Http POST method (1 mark), and use Ajax to call this operation on client side (1 mark).

  This operation receives the information of a player and creates an instance of the class Player. It will then add this player into the existing player list and saves into the players.txt file.

  Please note that if the ID of the player you want to register already exists, you then need to **update** this player's information in the players.txt file instead.

  Note that: The new or updated player list after this operation needs to be displayed on top of the page. Input validation is needed.