AJAX pointed task

19/1/2017

Time: 90 minutes (you have to *send* solution *before* 11:50) Points: 25

1 Sending the solution

Pack **source** files (including csproj file) into a zip archive and name it login.zip, substitute login with your login in faculty network. Do not add any compiled files to the archive.

Send via email:

`To:` jan.karwowski@mini.pw.edu.pl
`Subject:` `[WebApp] AJAX` `login`
(substitute login with your login in faculty network)
`Attachments:` login.zip
`Content:` empty

2 The task: coffee drinking competition system

You have to develop Webapp for managing player lists in coffee drinking competition and simulate the tournament.

You are provided with sample html file which can be used as a base for the frontend and C# file implementing backend functionality of the app (storing and retrieving players, performing competition and returning its result).

2.1 General requirements

You have to develop a web application using AJAX. Client side have to consist only of static content (html, javascript, css). No html or other files for the browser can be generated dynamically. You can use provided files as a base for client side part of the app. The dynamic content of the page has to be provided by server side using WebAPI technology and loaded into the browser using javascript. Server side API has do be designed according to REST design pattern.

2.2 Detailed Requirements

1. 3 points Server side provides REST-style API
2. 3 points *Add player* button adds a new player to the server's list of players.
3. 2 points When website is loaded a list in Competitors section is populated with content loaded from server. The list contains a start number and two buttons for each player. The list is **not** refreshed automatically (e.g. when its content changes after adding, deleting, ...). See Figure 1.
4. 3 points The info button requests detailed info about given player and populates it into Player block on the right. See Figure 2.
5. 2 points Delete button removes given player from server if exists.

Figure 1: Players' list filled with data

6. 2 points Refresh button requests a competitor list from server and replaces Competitors section list with new contents (i.e. refreshes data loaded at page load).

7. 4 points After each request that does not have immediate visible effect: adding player, deleting player and after failure of any request a respective message is appended to the list in Messages section. Add player request fails when a player with given start number already exists or if fields are incorrect (Player constructor throws an exception). Remove player request fails when trying to remove nonexistent player. Player info request fails when there is no player with given number. All information about (non-)existence of players are returned by provided backend functions. The list is not cleared and all messages after loading page are retained until the next reload. See Figure 3.

8. 6 points *Start competition* button requests performing three competitions in parallel (by performing three separate request to a server). And after all of them are finished populates *Results* section with list of numbers of all winners. Until all request are finished a text *Waiting ...* is visible instead of the list. Any earlier results are cleared. See Figure 4.

   Note: The method for performing competition in C# code contains artificial (random `Thread.sleep()`) which will cause requests to take different amount of time. This is intentional.

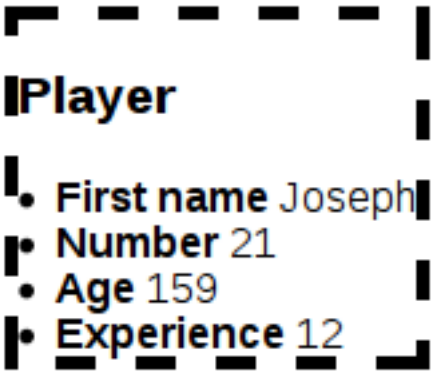

Figure 2: Populated player info box

Messages

1. Error adding player
2. PlayerAdded
3. Player 7 removed
4. Error getting player info
5. Error getting player info
6. Error when removing player

Figure 3: Message log

Results

Players who have received a point:

- 2
- 6
- 4
- 6
- 4
- 8
- 6
- 2
- 21

Figure 4: Coffee drinking competition results