PRIMO PROGETTO | Diario di lavoro - 24.10.2018

Matan Davidi

Trevano, 24.10.2018

Lavori svolti

Orario	Lavoro svolto
13:15 - 14:45	Fine implementazione - creazione pagina ReadCheck

L'ultima lezione ho avuto il problema di "Riuscire a far passare i dati da un controller all'altro.", ma ho trovato durante questa lezione che è più facile costruire il modello nel ReadCheck controller creando un nuovo oggetto CSVHelper che si riferisca a "Registrazioni_aaaa_mm_gg.csv" e ne legga i dati. A questo proposito ho creato una classe chiamata "UserData" che contiene i dati che vengono letti dal file, compresi con la data di inserimento, che viene usata come modello della view ReadCheck.

Inizialmente, come ho scritto l'ultima lezione, avevo deciso di effettuare la validazione del campo "Data di nascita" tramite jQuery, quindi ho scritto tutto nel site.js che si trova nella cartella wwwroot/js.

Per quanto riguarda il controllo dell'intestazione dei file CSV, creo prima di tutto uno StringBuilder chiamato header che contiene l'intestazione, quindi le colonne del CSV separate dal separatore ed effettuo il controllo confrontando la prima riga del file, usando il seguente codice:

```
File.ReadAllLines(Path,
Encoding).FirstOrDefault().Equals(header.ToString().TrimEnd(Environment.NewLine.To
CharArray()))
```

dove Path e Encoding sono proprietà della classe da definire nel costruttore, la prima contiene il percorso verso il file CSV e la seconda definisce la codifica con cui leggere il file.

Ho poi creato un metodo

```
public static void AddToBeginning(string text, string path)
```

che, come si può intuire dal nome, scrive del testo nella prima riga del file. Ho deciso di fare un metodo statico in modo da poterlo richiamare anche senza dover creare un'istanza, che potrebbe tornarmi molto utile in futuro.

```
public static void AddToBeginning(string text, string path)
{
    string tempfile = System.IO.Path.GetTempPath() + Guid.NewGuid().ToString() + ".csv";
    using (var writer = new StreamWriter(tempfile))
    using (var reader = new StreamReader(path))
    {
        writer.WriteLine(text);
        while (!reader.EndOfStream)
            writer.WriteLine(reader.ReadLine());
    }
    File.Copy(tempfile, path, true);
}
```

Concretamente questo metodo crea un nuovo file temporaneo al quale viene aggiunto il testo contenuto nel parametro text e, in seguito, il testo del file al quale punta il parametro path. Questo file temporaneo viene poi copiato al posto del file che si trova nel percorso path.

Questa soluzione è stata trovata su stackoverflow.com

Inoltre, oggi ho creato una nuova view Index all'interno della cartella creata in precedenza ReadCheck, che contiene una tabella all'interno della quale verranno inseriti i dati letti dal file "Registrazioni_aaaa_mm_dd.csv" relativi all'ultimo inserimento effettuato, ossia quello dell'utente corrente. Sotto alla tabella si trova un bottone che riporta alla pagina principale dell'applicazione.

I dati da passare a questa view sono letti, appunto, dal file "Registrazioni_aaaa_mm_dd.csv" all'interno del ReadCheckController, più precisamente nel metodo

```
public IActionResult Index()
```

dentro il quale viene creata una nuova istanza di CSVHelper esattamente come quella che scrive sul file nel SignupController, quindi:

```
CSVHelper csv = new CSVHelper(
    string.Format(
        "wwwroot/Registrazioni/Registrazioni_{0}_{1}_{2}.csv",
        DateTime.Now.Year, DateTime.Now.Month, DateTime.Now.Day),
    ";",
    new string[] {
        "data", "nome", "cognome", "data_nascita", "via", "numero_civico",
        "citta", "nap", "telefono", "email", "sesso", "hobby", "professione"
    });
```

di cui viene chiamato il metodo ReadAllLines() e viene creato un oggetto di tipo UserData tramite uno degli overload del suo costruttore che accetta un array di stringhe, ossia uno degli elementi della lista che viene ritornata con il metodo ReadAllLines. Visto che l'elemento più recente viene scritto per ultimo, vado a prendere l'ultimo elemento della lista.

```
UserData model = new UserData(csv.ReadAllLines().LastOrDefault());
```

Infine viene ritornata la view "Index" con l'istanza di UserData "model" come modello, in modo che la view venga popolata con i suoi dati.

Problemi riscontrati e soluzioni adottate

Punto della situazione rispetto alla pianificazione

Sono in orario con la pianificazione

Programma di massima per la prossima giornata di lavoro

Proseguire con la documentazione dell'implementazione e i test case.