Advanced Search

Topics

- ❏ Search w/ Multiple Keywords
- ❏ Deferred Execution and IQueryable<T>
- ❏ HTML5 Autocomplete and Autofill
- ❏ jQuery UI Autocomplete widget
- ❏ Autocomplete w/ Web Service
- ❏ Autocomplete w/ Spellcheck

Searching with multiple keywords

```
public async Task<ActionResult> Index(string q = null, int page = 1, int pageSize = 3)
{
  // Get all the data needed from the tables
  // Joins and .Include() calls should go here
  IQueryable<Thing> query = _db.Things;

  // Build the WHERE clause
  if (!string.IsNullOrWhiteSpace(q))
  {
    string[] keywords = Regex.Split(q, @"\s+");
    foreach (string word in keywords)
    {
      query = query.Where(x => x.Name.Contains(word) ||
                               x.Tags.Contains(word) ||
                               x.Description.Contains(word) ||
                               x.Category.Contains(word));
    }
  }

  // Count the total number of items in the result set
  int totalCount = await query.CountAsync();

  // Download a single page of results
  // Remember to always include an ORDER BY clause
  List<Thing> things =
    await query.OrderBy(x => x.Name)
               .ThenBy(x => x.Id)
               .Skip((page - 1) * pageSize)
               .Take(pageSize).ToListAsync();

  // Build the view model and return it
  var viewModel = new ThingListViewModel
  {
    Things = things,
    Q = q,
    PagingInfo = new PagingInfo
    {
      CurrentPage = page,
      ItemsPerPage = pageSize,
      TotalItems = totalCount
    },
  }
  return View(viewModel);
}
```

Deferred Execution and IQueryable<T>

> "[Deferred Execution] is a subject that developers must understand if they want to use the full power of LINQ. It is not a particularly difficult subject, but it is one that requires us to follow chains of thought that we have not typically pursued when working with imperative code."
> **LINQ and Deferred Execution**
> https://blogs.msdn.microsoft.com/charlie/2007/12/10/linq-and-deferred-execution/

LINQ and Deferred Execution Video Series by Jamie King

C# yield Statement Introduction
https://youtu.be/L6R08ajgZpU
C# LINQ - Introduction to Deferred Execution
https://youtu.be/V3xx5EvPDAE
C# LINQ - Deferred Execution Explained
https://youtu.be/fppy6TM79-c
C# LINQ - Deferred Execution - Assembly Line
https://youtu.be/3P9nDHJMWqI
C# LINQ - Deferred Execution - Runtime Created Assembly Lines
https://youtu.be/6g1C7wEOmIU

IQueryable<T> vs. IEnumerable<T>

What to use when?

IEnumerable<T>

- Prevents deferred execution used to build and execute the query as an expression tree.
- Use if data is available in within the same process i.e. it is stored in an Array or List.
- Use with **Linq to Object** and **Linq to XML.**
- Not suitable for paging like scenarios, because the entire dataset would be loaded and paging done in .NET runtime.

IQueryable<T>

- Fully facilitates deferred execution.
- When querying data from out of process i.e. SQL, LDAP, Files etc.
- Used with **Linq to SQL.**
- By the virtue of support of deferred execution, it is best for queries in paging like scenarios.

All About IEnumerable and IQueryable
https://taagung.com/ienumerable-and-iqueryable/

Autocomplete vs. Autofill

Autocomplete - The browser provides suggestion(s) for how to complete whatever the user has started typing in. By default this is based on previously entered form data, but developers can use a variety of methods to customize this behavior. *Frequently used on search boxes.*

Autofill - The browser automatically fills in values for form fields based on previously entered form data. This typically happens with little to no user interaction. *Frequently used with email addresses, phone numbers, addresses, credit card info, and login credentials.*

Help users get work done faster with Autocomplete and Autofill

> "People hate filling out web forms, especially on mobile devices. They can be slow and frustrating to complete [...] This leads to high user drop-off and frustration. To help make things easier for users, browsers have long been able to autocomplete fields on behalf of the user. Chrome took this a step further in 2011 by introducing Autofill, which fills in entire forms based on a user's Autofill profile."
>
> **Help users checkout faster with Autofill**
> https://developers.google.com/web/updates/2015/06/checkout-faster-with-autofill

HTML5 Autocomplete Attribute

The autocomplete attribute was added to official HTML 5.2 standard in 2017.

By default autocomplete is turned on for all forms and all input fields. Sometimes this is not desirable, especially when you are building your own autocomplete system. You can use the HTML5 **autocomplete** attribute to disable autocomplete for a specific field by setting it to **off**

```
<form action="/Account/Register" method="POST">
  <label for="fname">First Name</label>
  <input type="text" id="fname" name="fname">
  <label for="lname">Last Name</label>
  <input type="text" id="lname" name="lname">
  <label for="email">Email</label>
  <input type="email" id="email" name="email" autocomplete="off">
  <button type="submit">Register</button>
</form>
```

w3schools: Autocomplete Attribute
https://www.w3schools.com/tags/att_input_autocomplete.asp
MDN: Autocomplete Attribute
https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/autocomplete
HTML 5.2 Standard - Autocomplete Attribute
https://www.w3.org/TR/html52/sec-forms.html#autofilling-form-controls-the-autocomplete-attribute

Autofill Field Names

"Until recently, there were no standards when it came to autofill. Each browser implemented their autofill features differently and there was little documentation on how a browser determines what content a field expects. [...]"

"Thankfully, there is a way forward for autofill. [HTML 5.2] recently expanded the **autocomplete** attribute to provide hints to the browser about what content a field expects."

"The autocomplete attribute has been around for several years. It started with two values: **on** and **off.** By default, the autocomplete attribute is set to **on** which means that the browser is free to store values submitted and to autofill them in the form."

"[Recently, additional values have been added as options for the autocomplete attribute. These new options are called **autofill field names.** These names tell the browser what information the field is looking for. For example, one of the autofill field names is **organization.** The HTML5 specification says that organization refers to:]"

> *Company name corresponding to the person, address, or contact information in the other fields associated with this field*

If you wanted to tell the browser to autofill the organization field, your code would look something like this:

```
<input type="text" id="org" name="org" autocomplete="organization">
```

Autofill: What web devs should know, but don't
https://cloudfour.com/thinks/autofill-what-web-devs-should-know-but-dont/

Shipping and Billing Addresses

The value of the autocomplete attribute is actually a space-separated list of tokens. So for example, if you wanted to collect shipping information, you would prepend the autocomplete value with the **shipping** token like so:

```
<textarea name="shipping-address" autocomplete="shipping street-address"></textarea>
<input type="text" name="shipping-city" autocomplete="shipping address-level2">
<input type="text" name="shipping-state" autocomplete="shipping address-level1">
<input type="text" name="shipping-country" autocomplete="shipping country-name">
<input type="text" name="shipping-postal-code" autocomplete="shipping postal-code">
```

The **billing** token works similar to the **shipping** token.

Login Information

Field Name	Meaning
username	A username
email	E-mail address
tel	Full telephone number, including country code
current-password	The current password for the account identified by the username field (e.g. when logging in)
new-password	A new password (e.g. when creating an account or changing a password)
one-time-code	One-time code used for verifying user identity

Credit Cards and Transactions

Field Name	Meaning
transaction-amount	The amount that the user would like for the transaction (e.g. when entering a bid or sale price)
transaction-currency	The currency that the user would prefer the transaction to use
cc-name	Cardholder name
cc-number	Credit card number
cc-exp-month	Month component of the expiration date
cc-exp-year	Year component of the expiration date
cc-csc	Security code

Other Field Names

Field Name	Meaning
off	disable autocomplete
name	Full Name
given-name	Given name (in some Western cultures, also known as the first name)
additional-name	Additional names (in some Western cultures, also known as middle names, forenames other than the first name)
family-name	Family name (in some Western cultures, also known as the last name or surname)
nickname	Nickname, screen name, handle: a typically short name used instead of the full name
organization	Company name corresponding to the person, address, or contact information in the other fields associated with this field
organization-title	Job title (e.g. "Software Engineer", "Senior Vice President", "Deputy Managing Director")
street-address	Street address (multiple lines, newlines preserved)
address-line1	Street address (one line per field)
address-line2	Street address (one line per field)
address-level1	The broadest administrative level in the address, i.e. the province within which the locality is found; for example, in the US, this would be the state; in Switzerland it would be the canton; in the UK, the post town
address-level2	The second administrative level, in addresses with two or more administrative levels; in the countries with two administrative levels, this would typically be the city, town, village, or other locality within which the relevant street address is found
postal-code	Postal code, post code, ZIP code, CEDEX code
country	Country code
country-name	Country name
language	Preferred language
bday	Birthday
sex	Gender identity (e.g. Female, Fa'afafine)

Official Table of Autofill Field Names
https://html.spec.whatwg.org/multipage/form-control-infrastructure.html#inappropriate-for-the-control
MDN: Autocomplete Attribute
https://developer.mozilla.org/en-US/docs/Web/HTML/Attributes/autocomplete

Autocomplete w/ datalist

Using the HTML5 datalist element and input list attribute you can easily set up an input box with a custom autocomplete without jQuery, JavaScript or similar.

Add the **list** attribute to an input field and then create a **datalist** element. The **list** value should be the same as the **datalist's id.**

```
<label for="FavPasta">Favorite Pasta</label>
<input type="text" id="FavPasta" name="FavPast" list="pasta">

<datalist id="pasta">
  <option>Bavette</option>
  <option>Cannelloni</option>
  <option>Fiorentine</option>
  <option>Gnocchi</option>
  <option>Pappardelle</option>
  <option>Penne lisce</option>
  <option>Pici</option>
  <option>Rigatoni</option>
  <option>Spaghetti</option>
  <option>Tagliatelle</option>
</datalist>
```

Native HTML5 autocomplete with datalist
http://www.tipue.com/blog/input-list/
MDN: <datalist>
https://developer.mozilla.org/en-US/docs/Web/HTML/Element/datalist

Autocomplete w/ Usernames and Passwords

If you want to enable auto complete on a login page, that is fairly straight-forward.

```
<form action="/Account/Login" method="POST">
  <label for="email">User Name</label>
  <input type="email" id="email" name="email" autocomplete="email">
  <label for="password">Password</label>
  <input type="password" id="password" name="password" autocomplete="current-password">
  <button type="submit">Login</button>
</form>
```

But to disable it on registration pages requires a few tricks.

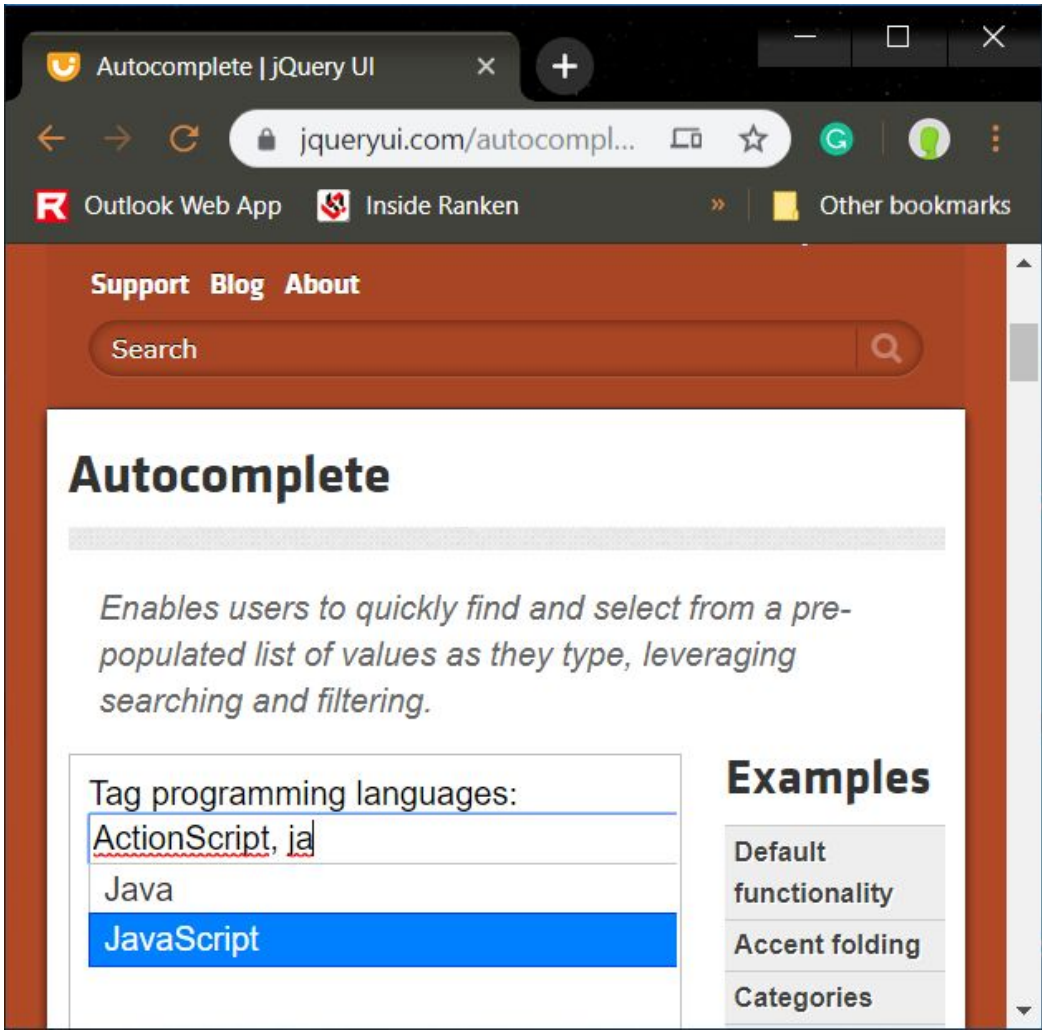
```
<form action="/Account/Register" method="POST">
  <!-- Fake Login Form -->
  <input type="email" style="display: none">
  <input type="password" style="display: none">

  <!-- Actual Register Form -->
  <label for="email">User Name</label>
  <input type="email" id="email" name="email" autocomplete="email">
  <label for="newPassword">Password</label>
  <input type="password" id="newPassword" name="newPassword"
         autocomplete="new-password">
  <label for="confirmPassword">Confirm Password</label>
  <input type="password" id="confirmPassword" name="confirmPassword"
         autocomplete="new-password">
  <button type="submit">Register</button>
</form>
```

Tips & Tricks to disable autocomplete of login information.

- ❏ **autocomplete="off"** is ignored by Chrome and some other browsers.
- ❏ Use **auto-complete="new-password"** on the Password and Confirm Password fields. This should work for any standards-compliant browser.
- ❏ Add a hidden pair of username and password fields before the fields. Using **type=hidden** does not work for this trick, the fields must be hidden with **CSS.** This works on older browsers.
- ❏ To fool **LastPass**, which ignores the **autocomplete** attribute, give both password fields an **id** and **name** other than **password.** Such as **newPassword** and **confirmPassword.**

jQuery UI Autocomplete

The jQuery UI library provides a flexible widget for building custom autocomplete functionality.



Autocomplete Widget: Default Functionality Example
https://jqueryui.com/autocomplete/#default
Autocomplete Widget: Multiple Values Example
https://jqueryui.com/autocomplete/#multiple

*In the below examples, **search.php** receives the search term via a query parameter named **term** and returns the results as a **JSON Array of Strings.***

Autocomplete Widget: Remote Datasource Example
https://jqueryui.com/autocomplete/#remote
Autocomplete Widget: Multiple Values w/ Remote Datasource Example
https://jqueryui.com/autocomplete/#multiple-remote

Using jQuery UI Autocomplete

Official URLs here: https://code.jquery.com/ (remember to use https)

Add the following stylesheet inside the <head>.

```
<link rel="stylesheet" type="text/css"
      href="https://code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
```

Add the following script at the end of the <body>

```
<script
  src="https://code.jquery.com/ui/1.12.1/jquery-ui.min.js"
  integrity="sha256-VazP97ZCwtekAsvgPBSUwPFKdrwD3unUfSGVYrahUqU="
  crossorigin="anonymous"></script>
```

Add the autocomplete widget to an input field with the following JavaScript code:

```
const availablePastas = [
  "Bavette",
  "Cannelloni",
  "Fiorentine",
  "Gnocchi",
  "Pappardelle",
  "Penne lisce",
  "Pici",
  "Rigatoni",
  "Spaghetti",
  "Tagliatelle"
];
$( "#FavPasta" ).autocomplete({ source: availablePastas });
```

Or retrieve the options from a web service:

```
$( "#FavPasta" ).autocomplete({ source: "/Pasta/Names", minLength: 2 });
```

Order of Execution

The order of your stylesheets and scripts is very important. Stylesheets and scripts are executed from top to bottom. So they must be placed in the correct order to work properly.

Below is the suggested order of your stylesheets:

```
<link
  rel="stylesheet" type="text/css"
  href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
  integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
  crossorigin="anonymous">
<link
  rel="stylesheet" type="text/css"
  href="https://code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">
<link
  rel="stylesheet" type="text/css"
  href="~/Content/main.css">
```

Below is the suggested order for your scripts:

```
<script
  src="https://code.jquery.com/jquery-3.4.1.min.js"
  integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
  crossorigin="anonymous"></script>
<script
  src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
  integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1"
  crossorigin="anonymous"></script>
<script
  src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
  integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM"
  crossorigin="anonymous"></script>
<script
  src="https://code.jquery.com/ui/1.12.1/jquery-ui.min.js"
  integrity="sha256-VazP97ZCwtekAsvgPBSUwPFKdrwD3unUfSGVYrahUqU="
  crossorigin="anonymous"></script>
<script src="https://kit.fontawesome.com/[your kit code here].js"></script>

<script src="~/Scripts/jquery.validate.min.js"></script>
<script src="~/Scripts/jquery.validate.unobtrusive.min.js"></script>
<script src="~/Scripts/main.js"></script>
@RenderSection("Scripts", required: false)
```

Styling jQuery UI Autocomplete

While the jQuery UI Autocomplete widget works well out of the box, we typically want to style the popup and input field to be more user-friendly and visually appealing.

Below is the set of CSS rules that worked for me, change the colors to your liking:

```
/* autocomplete popup */
.ui-autocomplete {
  border: none !important;
  padding: 0;
  border-radius: 4px;
  max-height: 10em;
  overflow-x: hidden;
  overflow-y: auto;
  scrollbar-color: #00bc8c #222222;
  scrollbar-width: thin;
}
/* autocomplete menu item */
.ui-autocomplete .ui-menu-item-wrapper {
  padding: 4px;
  padding-left: 16px;
  border-radius: 4px;
}
/* autocomplete active menu item */
.ui-autocomplete .ui-menu-item-wrapper.ui-state-active {
  background-color: #00bc8c !important;
  border-color: #00bc8c !important;
}
```

And the HTML to go with it (including bootstrap classes):

```
<label for="q" class="sr-only form-label">Keywords</label>
<div class="input-group">
  <input type="text" id="q" name="q" placeholder="keywords" class="form-control">
  <div class="input-group-append">
    <button type="submit" class="btn btn-outline-primary">
      <i class="fa-fw fas fa-search"></i>
      Search
    </button>
  </div>
</div>
```

Building a web service to suggest search terms

Often we need to access the database to determine what search terms to suggest to the user. In this case we build a web service that accepts a single search term and returns a list of suggested search terms.

In the example below, we use the categories and tags of products in the database to build that list of suggested terms.

C#

```
public async Task<ActionResult> Keywords(string term)
{
  term = term.ToLower();

  var products =
      await _db.Products
                .Where(x => x.Category.Contains(term) || x.Tags.Contains(term))
                .Select(x => new { Category = x.Category.ToLower(),
                                   Tags = x.Tags.ToLower() })
                .ToListAsync();

  IEnumerable<string> categories =
      products.Select(x => x.Category);
  IEnumerable<string> tags =
      products.SelectMany(x => Regex.Split(x.Tags, @"\s*,\s*"));

  IEnumerable<string> keywords =
      categories.Union(tags)
                .Where(x => x.Contains(term))
                .OrderBy(x => x)
                .Distinct();

  return Json(keywords, JsonRequestBehavior.AllowGet);
}
```

JavaScript

```
$("#q").autocomplete({ source: "@Url.Action("Keywords")", minLength: 2 });
```

Spell Checking

A big part of providing good search suggestions is using a spell checker to correct spelling errors in the search query. This is often implemented using **Natural Language Processing (NLP)** tools. A common tool in spellchecking is the **Levenshtein Distance** metric which measures how different a word is compared to a valid word. The **Levenshtein Distance** counts the number of characters that you would need to insert, remove, replace, or swap to convert one word into another word. The correct spelling of a word usually has a small distance from the misspelled word.

Wikipedia: Levenshtein distance
https://en.wikipedia.org/wiki/Levenshtein_distance

NHunspell
https://github.com/hunspell/hunspell
http://www.crawler-lib.net/nhunspell (dead link)

1. Install nuget package **NHunspell**
2. Download dictionary files **en_US.aff** and **en_US.dic**
   (I used the files from libreoffice: https://cgit.freedesktop.org/libreoffice/dictionaries/tree/en)
3. Copy the dictionary files into the **C:\Program Files (x86)\IIS Express** folder.
4. Open the project folder and add the DLLs to git **Hunspellx64.dll** and **Hunspellx86.dll**
5. Add code to your autocomplete web service to use the spellchecker.

Example

```
var spelling = new NHunspell.Hunspell("en_US.aff", "en_US.dic");
var suggestions =
  spelling.Spell(term) ?
    new List<string>() { term } :
    spelling.Suggest(term).Take(5);

...

keywords = suggestions.Union(keywords);
```

Apache Solr
https://lucene.apache.org/solr/features.html

Apache Solr is another option which runs as a separate search server. Solr is out of scope for this class, but it is a good tool for enterprise applications.