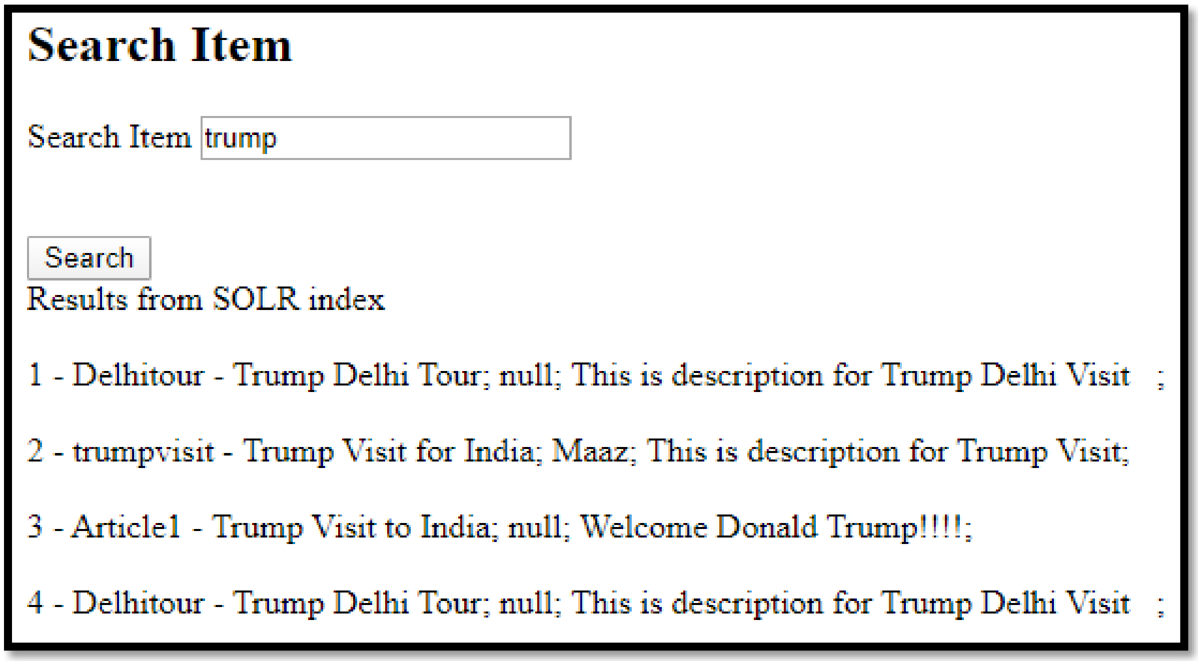
Sitecore with Solr Search

1. Create Search Component



Results from SOLR index

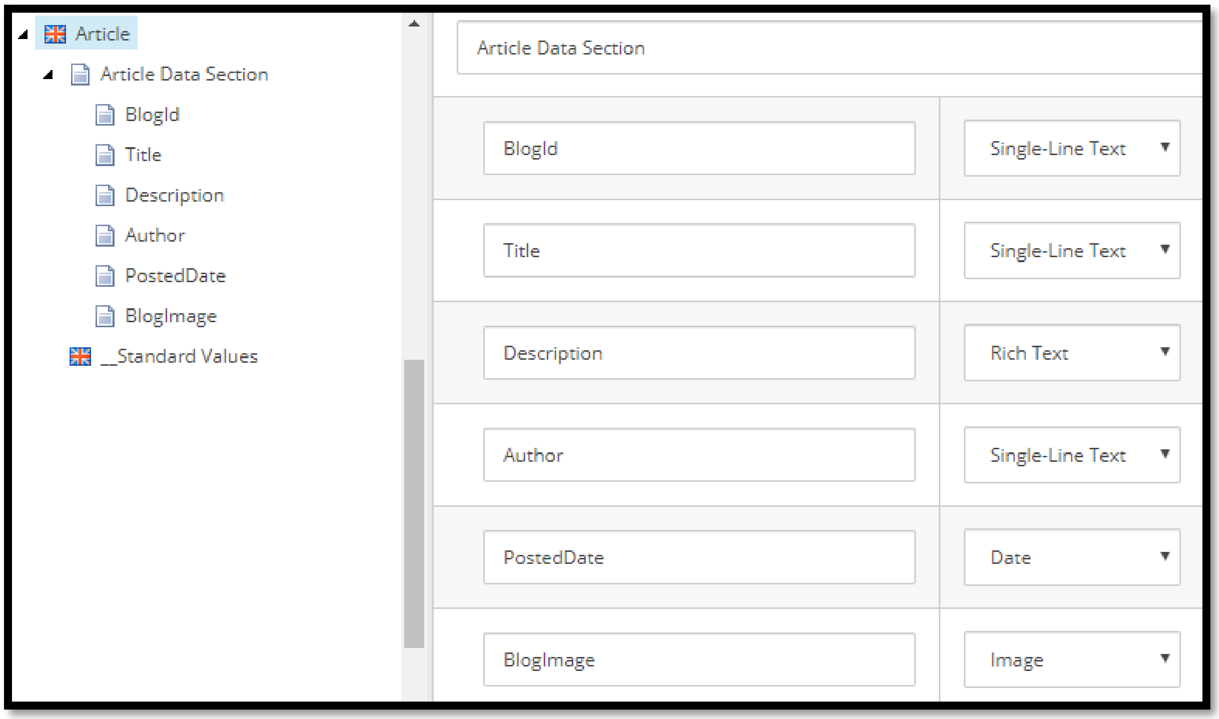
1 - Delhitour - Trump Delhi Tour; null; This is description for Trump Delhi Visit ;

2 - trumpvisit - Trump Visit for India; Maaz; This is description for Trump Visit;

3 - Article1 - Trump Visit to India; null; Welcome Donald Trump!!!!;

4 - Delhitour - Trump Delhi Tour; null; This is description for Trump Delhi Visit ;

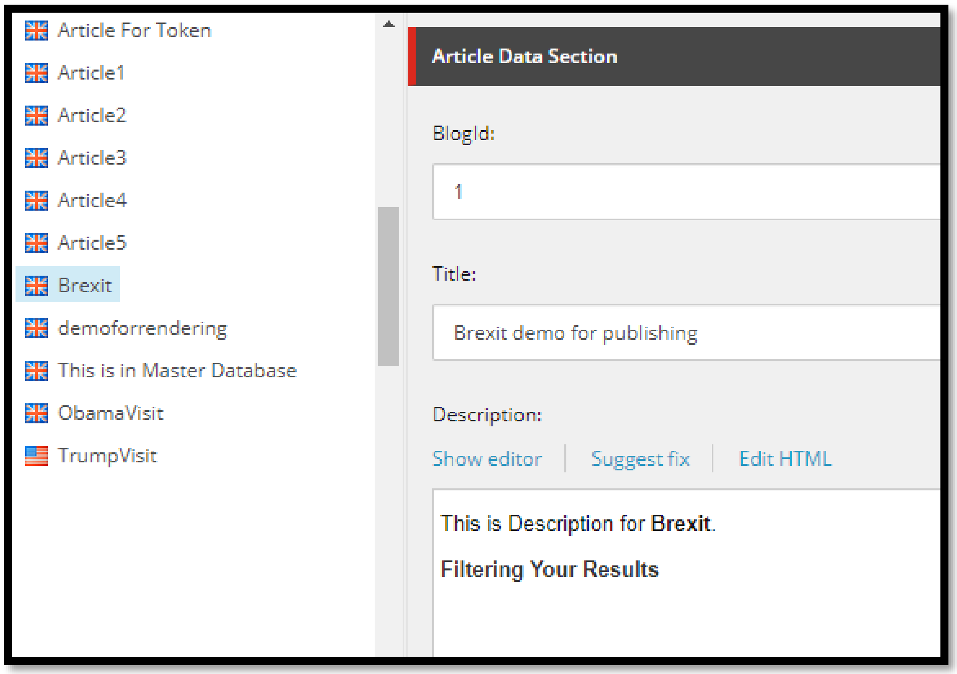
2. Check SOLR is Running at https://localhost:8983/solr
3. Create a template in Sitecore



Article Data Section

BlogId	Single-Line Text
Title	Single-Line Text
Description	Rich Text
Author	Single-Line Text
PostedDate	Date
BlogImage	Image

4. By using the same template create few items and put some data.



Article Data Section

BlogId: 1

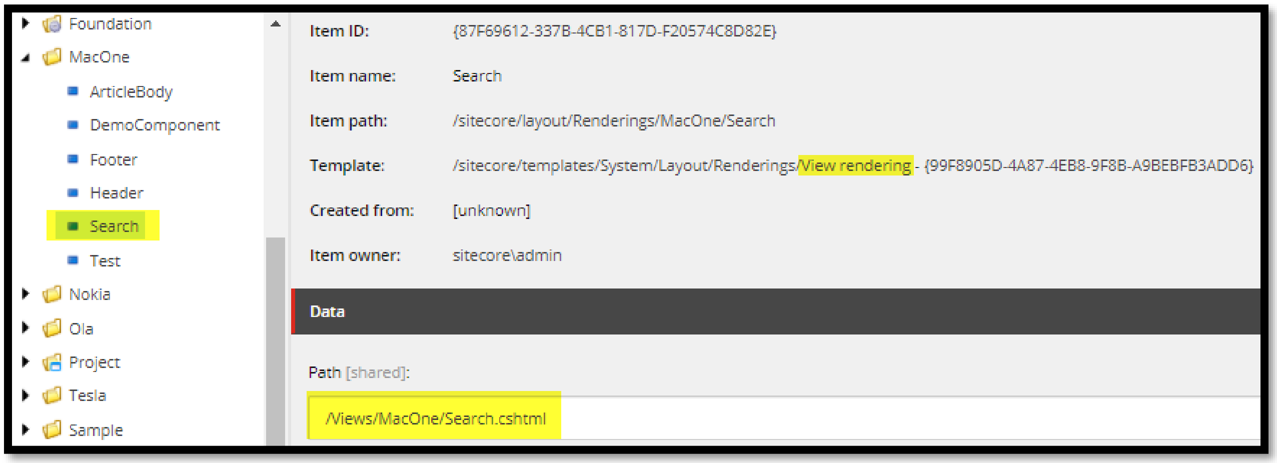
Title: Brexit demo for publishing

Description:

This is Description for **Brexit**.

Filtering Your Results

5. Create a View Rendering and assign it on HOME page.



Item ID: {87F69612-337B-4CB1-817D-F20574C8D82E}

Item name: Search

Item path: /sitecore/layout/Renderings/MacOne/Search

Template: /sitecore/templates/System/Layout/Renderings/View rendering - {99F8905D-4A87-4EB8-9F8B-A9BEBFB3ADD6}

Created from: [unknown]

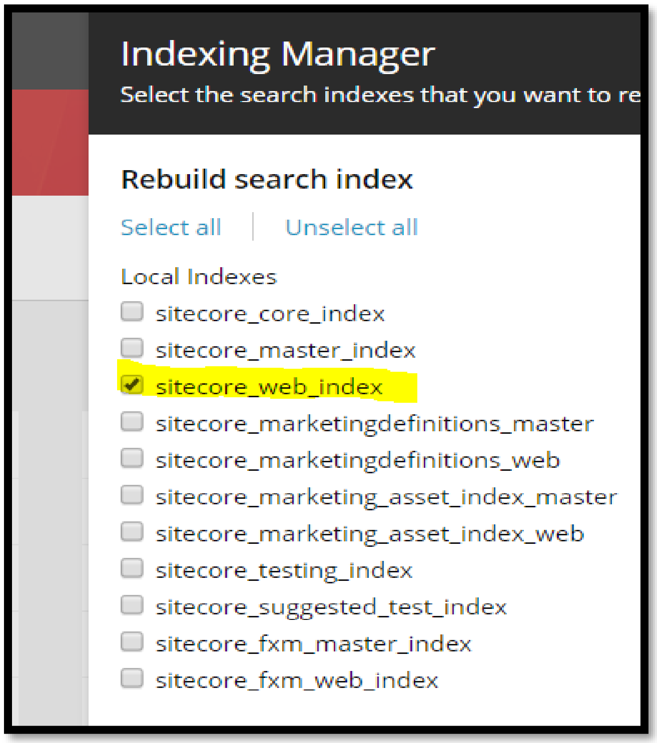
Item owner: sitecore\admin

Data

Path [shared]:

/Views/MacOne/Search.cshtml

6. Go to Sitecore Control Panel then Index manager. If you are using any existing indexes, then rebuild it. Or you can add a new index in the solution same core name add in Solr admin and rebuild that.



Indexing Manager

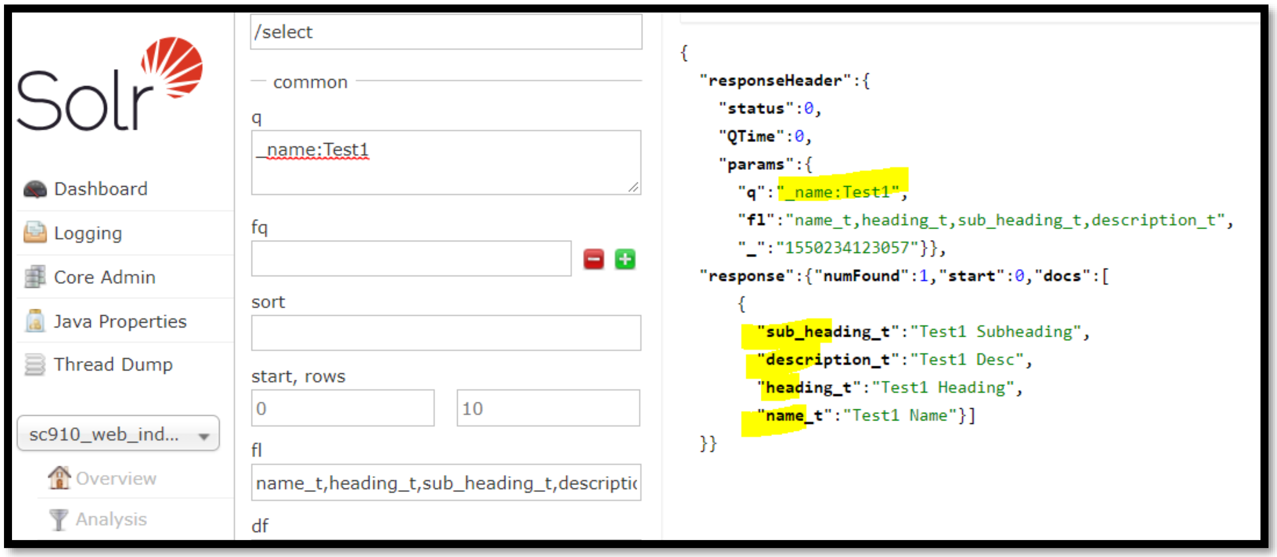
Select the search indexes that you want to re

Rebuild search index

Local Indexes

- [ ] sitecore_core_index
- [ ] sitecore_master_index
- [✓] sitecore_web_index
- [ ] sitecore_marketingdefinitions_master
- [ ] sitecore_marketingdefinitions_web
- [ ] sitecore_marketing_asset_index_master
- [ ] sitecore_marketing_asset_index_web
- [ ] sitecore_testing_index
- [ ] sitecore_suggested_test_index
- [ ] sitecore_fxm_master_index
- [ ] sitecore_fxm_web_index

7. After indexing check the data is present in the index.



/select

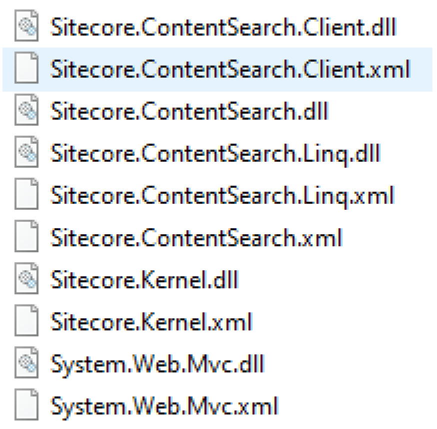
q: _name:Test1

start, rows: 0 10

fl: name_t,heading_t,sub_heading_t,descriptic

```
{
  "responseHeader":{
    "status":0,
    "QTime":0,
    "params":{
      "q":"_name:Test1",
      "fl":"name_t,heading_t,sub_heading_t,description_t",
      "_":"1550234123057"}},
  "response":{"numFound":1,"start":0,"docs":[
      {
        "sub_heading_t":"Test1 Subheading",
        "description_t":"Test1 Desc",
        "heading_t":"Test1 Heading",
        "name_t":"Test1 Name"}]
  }}
```

8. Open the Visual Studio.
9. Create an empty MVC application. Remove reference "***System.Web.Mvc.dll***" and add below references



Sitecore.ContentSearch.Client.dll
Sitecore.ContentSearch.Client.xml
Sitecore.ContentSearch.dll
Sitecore.ContentSearch.Linq.dll
Sitecore.ContentSearch.Linq.xml
Sitecore.ContentSearch.xml
Sitecore.Kernel.dll
Sitecore.Kernel.xml
System.Web.Mvc.dll
System.Web.Mvc.xml

Create a Model

Code

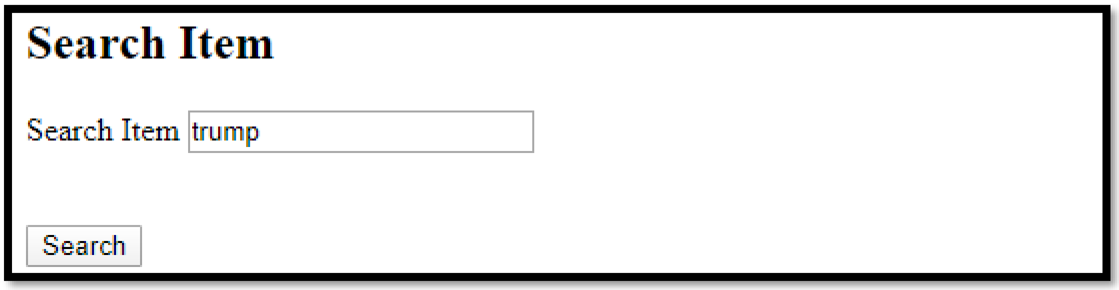
```
using Sitecore.ContentSearch;
using Sitecore.ContentSearch.SearchTypes;
using System.Collections.Generic;


namespace SitecoreSolrIndexing.Models
{

    public class SearchModel : SearchResultItem
    {
        [IndexField("_name")]
        public virtual string ItemName { get; set; }

        [IndexField("author_t")]
        public virtual string Author { get; set; }

        [IndexField("description_t")]
        public virtual string Description { get; set; } // Custom field on my template

        [IndexField("title_t")]
        public virtual string Title { get; set; } // Custom field on my template
    }


    public class SearchResult
    {
        public string ItemName { get; set; }

        public string Title { get; set; }
        public string Author { get; set; }

        public string Description { get; set; }
    }


    /// <summary>
    /// Custom search result model for binding to front end
    /// </summary>
    public class SearchResults
    {
        public List<SearchResult> Results { get; set; }
    }
}
```

Create a Controller.

Code:

```
using Sitecore.ContentSearch;
using Sitecore.ContentSearch.Linq;
using Sitecore.ContentSearch.Linq.Utilities;
using SitecoreSolrIndexing.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Linq.Expressions;
using System.Web.Mvc;

namespace SitecoreSolrIndexing.Controllers
{
    public class SolrIndexController : Controller
    {
        public ActionResult DoSearch(string searchText)
        {
            var myResults = new SearchResults
            {
                Results = new List<SearchResult>()
            };
            var searchIndex = ContentSearchManager.GetIndex("sitecore_web_index"); // Get
the search index
            var searchPredicate = GetSearchPredicate(searchText); // Build the search
predicate
            using (var searchContext = searchIndex.CreateSearchContext()) // Get a
context of the search index
            {
                //Select * from Sitecore_web_index Where Author="searchText" OR
Description="searchText" OR Title="searchText"
                var searchResults =
searchContext.GetQueryable<SearchModel>().Where(searchPredicate); // Search the index for
items which match the predicate

// This will get all of the results, which is not reccomended
                var fullResults = searchResults.GetResults();
                // This is better and will get paged results - page 1 with 10 results per
page
                //var pagedResults = searchResults.Page(1, 10).GetResults();
                foreach (var hit in fullResults.Hits)
                {
                    myResults.Results.Add(new SearchResult
                    {
                        Description = hit.Document.Description,
                        Title = hit.Document.Title,
                        ItemName = hit.Document.ItemName,
                        Author = hit.Document.Author
                    });
                }
                return new JsonResult { JsonRequestBehavior =
JsonRequestBehavior.AllowGet, Data = myResults };
            }
        }

        /// <summary>
        /// Search logic
        /// </summary>
        /// <param name="searchText">Search term</param>
        /// <returns>Search predicate object</returns>
        public static Expression<Func<SearchModel, bool>> GetSearchPredicate(string
searchText)
        {
            var predicate = PredicateBuilder.True<SearchModel>(); // Items which meet the
predicate
                                                                  // Search the whole
phrase - LIKE
                                                                  // predicate =
predicate.Or(x => x.DispalyName.Like(searchText)).Boost(1.2f);
                                                                  // predicate =
predicate.Or(x => x.Description.Like(searchText)).Boost(1.2f);
                                                                  // predicate =
predicate.Or(x => x.Title.Like(searchText)).Boost(1.2f);
                                                                  // Search the whole
phrase - CONTAINS
            predicate = predicate.Or(x => x.Author.Contains(searchText)); //
.Boost(2.0f);
            predicate = predicate.Or(x => x.Description.Contains(searchText)); //
.Boost(2.0f);
            predicate = predicate.Or(x => x.Title.Contains(searchText)); // .Boost(2.0f);
            //Where Author="searchText" OR Description="searchText" OR Title="searchText"
            return predicate;
        }
    }
}
```

Create a View.



Code:

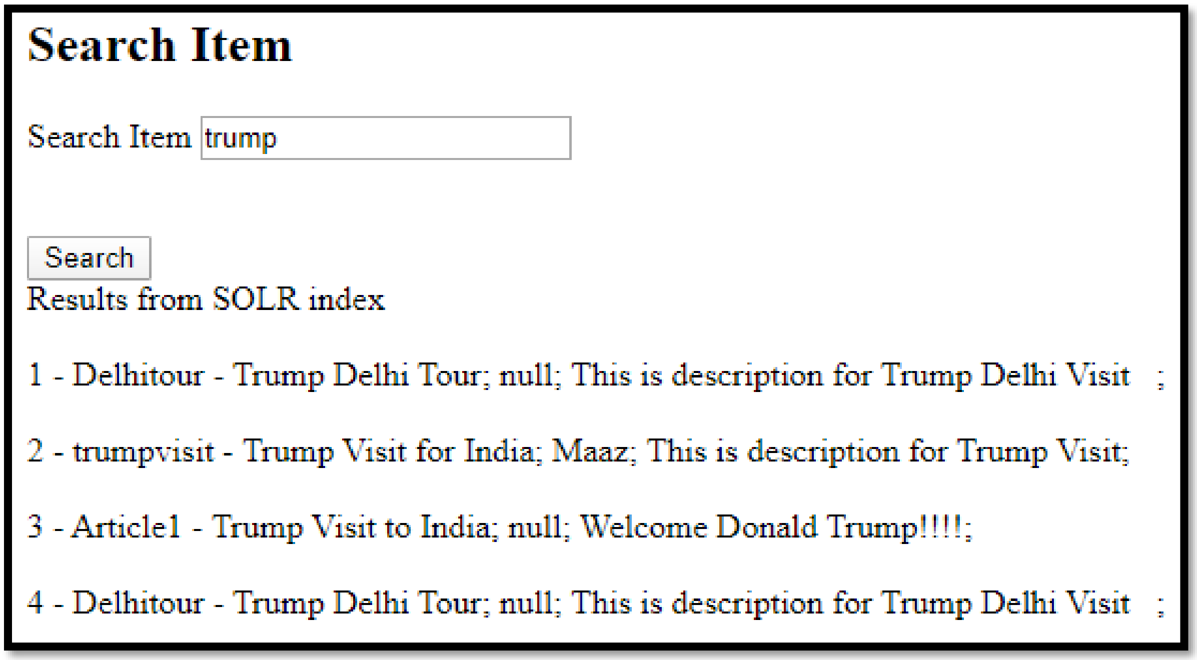
```
@{
    ViewBag.Title = "SOLR Search Component";
}

<h2>Search Item</h2>
<div class="form-example">
    <label for="name">Search Item </label>
    <input type="text" name="name" id="searchInput" required>
</div>
<br />
<br />

<div class="form-example">
    <button class="favorite styled" type="button" id="searchButton">
        Search
    </button>
</div>

<div id="resultsItem"></div>

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js"
type="text/javascript"></script>
<script>
              $(document).ready(function () {
                     $("#searchButton").click(function (e) {
                            // debugger;
                            e.preventDefault();
                            $.ajax({
                                   type: "GET",
                                   datatype: "JSON", url: "@Url.Action("DoSearch",
"AirtelSearchIndex")",

                                   contentType: "application/json",
                                   data: {
                                          searchText: $("#searchInput").val()
                                   },
                                   success: function (result) {
                                          var resultText = "";
                                          $('#resultsItem').text(resultText);
                                          $('#resultsItem').append("Results from SOLR
index <BR><BR>");

                                          $(result.Results).each(function (index, item) {
                                                 // each iteration
                                                 var Description = item.Description;
                                                 var Title = item.Title;
                                                 var Author = item.Author;
```

```
                                                    var ItemName = item.ItemName;
                                                    resultText += (index + 1) + " - " +
ItemName + Title + "; " + Author + "; " + Description + "; " + "<BR><BR>";
                                             });
                                             $('#resultsItem').append(resultText);
                                      },
                                      error: function (result) {
                                             alert('error');
                                      }
                               });
                        });
                 });
</script>
```

10. Build and Publish whole solution.

11. Assign view rendering to some item and try to access that item via browser, you should get below output.

12. In textbox, I put “trump” text to search. You get the search results.



Search Item

Search Item trump

Search

Results from SOLR index

1 - Delhitour - Trump Delhi Tour; null; This is description for Trump Delhi Visit ;

2 - trumpvisit - Trump Visit for India; Maaz; This is description for Trump Visit;

3 - Article1 - Trump Visit to India; null; Welcome Donald Trump!!!!;

4 - Delhitour - Trump Delhi Tour; null; This is description for Trump Delhi Visit ;

Optional step – To get intelligence in Views

1. Add references “Sitecore.Mvc.dll” and “Sitecore.Mvc.Analytics.dll”.
2. Add below entry in “Views/web.config” as

```
   <add namespace="Sitecore.Mvc"/>
</namespaces>
```

Pipeline – Page Not Found

1. **Create a Custom class**

Code:

```
using Sitecore.Data.Items;
using Sitecore.Pipelines.HttpRequest;

namespace MacOneSearch
{
    public class HttpRequestProcessor404 : HttpRequestProcessor
    {
        public override void Process(HttpRequestArgs args)
        {
            if (Sitecore.Context.Item != null || Sitecore.Context.Site == null ||
Sitecore.Context.Database == null || Sitecore.Context.Database.Name == "core")
            {
                return;
            }
            var pageNotFound = Sitecore.Context.Database.GetItem(@"{F2A6639A-F28D-4C50-
B8DB-B68E0BA8164A}");
            if (pageNotFound == null)
                return;
            args.ProcessorItem = pageNotFound;
            Sitecore.Context.Item = pageNotFound;
            args.HttpContext.Response.StatusCode = 404;
        }

    }
}
```

2. **Create a Config File:**

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/">
  <sitecore>
    <pipelines>
      <httpRequestBegin>
        <processor type="MacOneSearch.HttpRequestProcessor404,MacOneSearch"

patch:after="processor[@type='Sitecore.Pipelines.HttpRequest.ItemResolver,
Sitecore.Kernel']"/>
      </httpRequestBegin>
    </pipelines>
  </sitecore>
</configuration>
```

3. Build and Publish whole solution.

Pipeline

1. **Create a Custom Route for Sitecore URL under App_Start**

Code:

```
using Sitecore.Pipelines;
using System.Web.Mvc;
using System.Web.Routing;

namespace SitecoreSolrIndexing.CustomRoutes
{
    public class RegisterSolrCustomRoute
    {
        public virtual void Process(PipelineArgs args)
        {
            Register();
        }

        public static void Register()
        {
            RouteTable.Routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            //http://xp9p2.sc/api/solrsearchtesting/index/DoSearch?searchItem=corona
            RouteTable.Routes.MapRoute("MySearchRoute",
"api/solrsearchtesting/{controller}/{action}");
        }

    }
}
```

2. **Create a Config File:**

```
<?xml version="1.0" encoding="utf-8"?>
<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/">
       <sitecore>
              <pipelines>
                     <initialize>
        <processor type="SitecoreSolrIndexing.CustomRoutes.RegisterSolrCustomRoute,
SitecoreSolrIndexing"
patch:before="processor[@type='Sitecore.Mvc.Pipelines.Loader.InitializeRoutes,
Sitecore.Mvc']"/>
                     </initialize>
              </pipelines>
       </sitecore>
</configuration>
```

3. Build and Publish whole solution.

4. For getting JSON result, access API by
   http://mysitecoredomain.com/api/solrsearchtesting/index/DoSearch?searchItem=corona