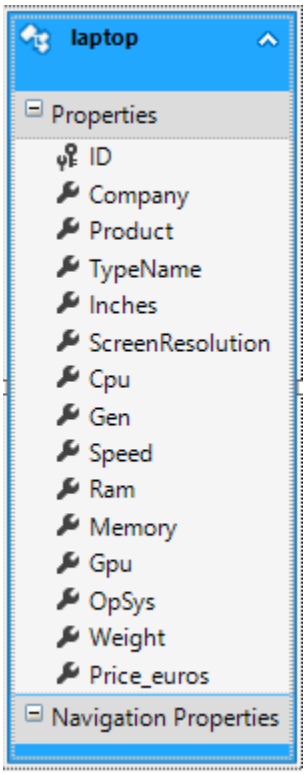
BÀI LAB SỐ 4:

Filter sản phẩm.

B1:

Chạy scripts SQL để cài đặt csdl và sử dụng Entity framework để thêm vào Folder Models trong MVC:



B2:

Tạo Folder Helper và Tạo Interface `IsearchParameters` trong thư mục Helper:

```
public interface ISearchParameters
    {
        string SearchTerm { get; set; }
       List<string> Company { get; set; }
        List<string> ScreenResolution { get; set; }
        SortCriteria SortBy { get; set; }
        List<string> TypeName { get; set; }
        List<float> Inches { get; set; }
        List<string> CPU { get; set; }
        List<float> Speed { get; set; }
        List<string> Memory { get; set; }
        List<int> Ram { get; set; }
        List<string> GPU { get; set; }
        List<string> OS { get; set; }
        double PriceLow { get; set; }
        double PriceHigh { get; set; }
        List<float> Weight { get; set; }
    }
```

Thêm lớp `SearchParameters` trong thư mục helper như sau:

```
public class SearchParameters : ISearchParameters
    {
        public SearchParameters()
        {
            SearchTerm = String.Empty;
            Company = new List<string>();
            ScreenResolution = new List<string>();
            SortBy = SortCriteria.Relevance;
            TypeName = new List<string>();
            Inches = new List<float>();
            CPU = new List<string>();
            Speed = new List<float>();
            Memory = new List<string>();
            Ram = new List<int>();
            GPU = new List<string>();
            OS = new List<string>();
            PriceLow = 0;
            PriceHigh = 0;
            Weight = new List<float>();
        }


        public string SearchTerm { get ; set ; }
        public List<string> Company { get ; set ; }
        public List<string> ScreenResolution { get ; set ; }
        public SortCriteria SortBy { get ; set ; }
        public List<string> TypeName { get ; set ; }
        public List<float> Inches { get ; set ; }
        public List<string> CPU { get ; set ; }
        public List<float> Speed { get ; set ; }
        public List<string> Memory { get ; set ; }
        public List<int> Ram { get ; set ; }
        public List<string> GPU { get ; set ; }
        public List<string> OS { get ; set ; }
        public double PriceLow { get ; set ; }
        public double PriceHigh { get ; set ; }
        public List<float> Weight { get ; set ; }
    }
```

Thêm enum `SortCriteria`

```
using System.ComponentModel;
namespace TMDT4.Helper
{
    public enum SortCriteria
    {
        [Description("Relevance")]
        Relevance = 0,
        [Description("Price: Low to High")]
        PriceLowToHigh = 1,
        [Description("Price: High to Low")]
        PriceHighToLow = 2
    }
}
```

Tạo thêm Lớp SearchBuilder trong thư mục helper như sau:

```
public class SearchBuilder
    {

        private ISearchParameters _searchParameters;
        public SearchBuilder() : this(new SearchParameters()) { }
        public SearchBuilder(ISearchParameters searchParameters)
        {
            _searchParameters = searchParameters;
        }
        public SearchBuilder SetSearchTerm(string searchTerm)
        {
            _searchParameters.SearchTerm = searchTerm;
            return this;
        }
        public SearchBuilder SetCompany(List<string> company)
        {
            _searchParameters.Company = company;
            return this;
        }
        public SearchBuilder SetScreenResolution(List<string> screensolution)
        {
            _searchParameters.ScreenResolution = screensolution;
            return this;
        }
        public SearchBuilder SetSortBy(SortCriteria sortby)
        {
            _searchParameters.SortBy = sortby;
            return this;
        }
        public SearchBuilder SetTypeName(List<string> typename)
        {
            _searchParameters.TypeName = typename;
            return this;
        }
        public SearchBuilder SetInch(List<float> inches)
        {
            _searchParameters.Inches = inches;
            return this;
        }
        public SearchBuilder SetCPU(List<string> cpu)
        {
            _searchParameters.CPU = cpu;
            return this;
        }
        public SearchBuilder SetSpeed(List<float> speed)
        {
            _searchParameters.Speed = speed;
            return this;
        }
        public SearchBuilder SetMemory(List<string> memory)
        {
            _searchParameters.Memory = memory;
            return this;
        }
```

```
        public SearchBuilder SetRam(List<int> ram)
         {
             _searchParameters.Ram = ram;
             return this;
         }
         public SearchBuilder SetGPU(List<string> gpu)
         {
             _searchParameters.GPU = gpu;
             return this;
         }
         public SearchBuilder SetOS(List<string> os)
         {
             _searchParameters.OS = os;
             return this;
         }
         public SearchBuilder SetPriceHigh(double pricehigh)
         {
             _searchParameters.PriceHigh = pricehigh;
             return this;
         }
         public SearchBuilder SetPriceLow(double pricelow)
         {
             _searchParameters.PriceLow = pricelow;
             return this;
         }
         public SearchBuilder SetWeight(List<float> weight)
         {
             _searchParameters.Weight = weight;
             return this;
         }
```

```
public IEnumerable<laptop> Build(TMDTT4Entities Entities)
        {
            var predicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();
            //search term
            if (!String.IsNullOrEmpty(_searchParameters.SearchTerm))
            {
                predicate = predicate.And(e => e.Product.Contains(_searchParameters.SearchTerm));
            }
            //price

            if (_searchParameters.PriceLow > 0 && _searchParameters.PriceHigh > 0)
            {
                predicate = predicate.And(e => e.Price_euros >= _searchParameters.PriceLow &&
e.Price_euros <= _searchParameters.PriceHigh);
            }
            //company
             if (_searchParameters.Company.Count > 0)
            {
                var temppredicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();

                foreach (var item in _searchParameters.Company)
                {
                    temppredicate = temppredicate.Or(e => e.Company == item);
                }
                predicate = predicate.And(temppredicate);
            }
            //ScreenResolution
            if (_searchParameters.ScreenResolution.Count > 0)
            {
                var temppredicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();
                foreach (var item in _searchParameters.ScreenResolution)
                {
                    temppredicate = temppredicate.Or(e => e.ScreenResolution == item);
                }
                predicate = predicate.And(temppredicate);
            }

              //TypeName
```

```
//TypeName
            if (_searchParameters.TypeName.Count > 0)
            {

                var tempppredicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();
                foreach (var item in _searchParameters.TypeName)
                {
                    tempppredicate = tempppredicate.Or(e => e.TypeName == item);
                }
                predicate = predicate.And(tempppredicate);
            }
            //Inches
            if (_searchParameters.Inches.Count > 0)
            {
                var tempppredicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();
                foreach (var item in _searchParameters.Inches)
                {
                    tempppredicate = tempppredicate.Or(e => e.Inches == item);
                }
                predicate = predicate.And(tempppredicate);
            }
            //CPU
            if (_searchParameters.CPU.Count > 0)
            {
                var tempppredicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();
                foreach (var item in _searchParameters.CPU)
                {
                    tempppredicate = tempppredicate.Or(e => e.Cpu == item);
                }
                predicate = predicate.And(tempppredicate);
            }
            //Speed
            if (_searchParameters.Speed.Count > 0)
            {
                var tempppredicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();
                foreach (var item in _searchParameters.Speed)
                {
                    tempppredicate = tempppredicate.Or(e => e.Speed == item);
                }
                predicate = predicate.And(tempppredicate);
            }
            //Memory
            if (_searchParameters.Memory.Count > 0)
            {
                var tempppredicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();
                foreach (var item in _searchParameters.Memory)
                {
                    tempppredicate = tempppredicate.Or(e => e.Memory == item);
                }
                predicate = predicate.And(tempppredicate);
            }
            //Ram
            if (_searchParameters.Ram.Count > 0)
            {
                var tempppredicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();
                foreach (var item in _searchParameters.Ram)
                {
                    tempppredicate = tempppredicate.Or(e => e.Ram == item);
                }
                predicate = predicate.And(tempppredicate);
            }
            //GPU
            if (_searchParameters.GPU.Count > 0)
            {
                var tempppredicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();
                foreach (var item in _searchParameters.GPU)
                {
                    tempppredicate = tempppredicate.Or(e => e.Gpu == item);
                }
                predicate = predicate.And(tempppredicate);
            }
```

```
//OS
            if (_searchParameters.OS.Count > 0)
            {
                var temppredicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();
                foreach (var item in _searchParameters.OS)
                {
                    temppredicate = temppredicate.Or(e => e.OpSys == item);
                }
                predicate = predicate.And(temppredicate);
            }
            //Weight
            if (_searchParameters.Weight.Count > 0)
            {
                var temppredicate = PredicateExtensions.PredicateExtensions.Begin<laptop>();
                foreach (var item in _searchParameters.Weight)
                {
                    temppredicate = temppredicate .Or(e => e.Weight == item);
                }
                predicate = predicate.And(temppredicate);
            }            var records = Entities.laptops.Where(predicate);
            switch (_searchParameters.SortBy)
            {
                case SortCriteria.Relevance:
                    break;
                case SortCriteria.PriceLowToHigh:
                    records = records.OrderBy(e=>e.Price_euros);
                    break;
                case SortCriteria.PriceHighToLow:
                    records = records.OrderByDescending(e => e.Price_euros);
                    break;
                default:
                    break;
            }

            return records;
        }
```

B3: Sửa file `RouteConfig` thành

```
public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");
            routes.MapRoute(
                name: "Home",
                url: "",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional });
            routes.MapRoute(
                name: "Default",
                url: "web/{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id =
UrlParameter.Optional }
            );
        }
```

B4:

Tạo Controller Laptop với nội dung như sau:

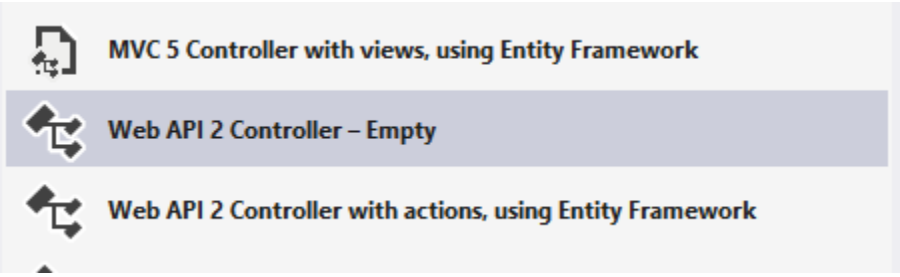
```
public class LaptopController : Controller
    {
        TMDTT4Entities db = new TMDTT4Entities();
        // GET: Laptop
        public ActionResult Index()
        {
            return View();
        }

        public ActionResult getCompany()
        {
            var ListCompany = db.laptops.GroupBy(l => l.Company).Select(l=>l.Key);
            return PartialView(ListCompany);
        }
        public ActionResult getCPU()
        {
            var ListCPU = db.laptops.GroupBy(l => l.Cpu).Select(l => l.Key);
            return PartialView(ListCPU);
        }
        /// Sv tự thêm các thông tin khác
    }
```

Tạo PartialView tương ứng với các Action vừa tạo tương tự như sau:

```
getCompany.cshtml   RouteConfig.cs   SortCriteria.cs
1   @model IEnumerable<string>
2   <div id="2">
3       @foreach (var item in Model)
4       {
5           @Html.CheckBox(item);
6           @Html.Label(item);
7           <b> </b>
8
9       }
10  </div>
```

B5: Tạo một Controller kiểu WebApi đặt tên là `LaptopAPI` và thêm dòng `GlobalConfiguration.Configure(WebApiConfig.Register);`

Vào trong hàm `Application_Start của File : Global.asax`



B6: trong File `LaptopAPI` viết các phương thức sau:

```
        TMDTT4Entities db = new TMDTT4Entities();
         public IEnumerable<laptop> getAll()
         {
             var searchQuery = db.laptops;
             return searchQuery;
         }
```

```
public IHttpActionResult GetProduct(string features_hash)
         {
             SearchParameters seacrhParameters = new SearchParameters();
             string[] ListQuery;
             ListQuery = features_hash.Split('_');
             foreach (var item in ListQuery)
             {
                 string typeQuery = item.Split('-')[0];
                 string QueryData = item.Split('-')[1];
                 switch (typeQuery)
                 {
                     case "1":
                         seacrhParameters.SearchTerm = QueryData;
                         break;
                     case "2":
                         seacrhParameters.Company.Add(QueryData);
                         break;
                     case "3":
                         seacrhParameters.ScreenResolution.Add(QueryData);
                         break;
                     case "4":
                         seacrhParameters.TypeName.Add(QueryData);
                         break;
                     case "5":
                         seacrhParameters.Inches.Add(float.Parse(QueryData));
                         break;
                     case "6":
                         seacrhParameters.CPU.Add(QueryData);
                         break;
                     case "7":
                         seacrhParameters.Speed.Add(float.Parse(QueryData));
                         break;
                     case "8":
                         seacrhParameters.Memory.Add(QueryData);
                         break;
                     case "9":
                         seacrhParameters.Ram.Add(Int32.Parse(QueryData));
                         break;
                     case "10":
                         seacrhParameters.GPU.Add(QueryData);
                         break;
                     case "11":
                         seacrhParameters.OS.Add(QueryData);
                         break;
```

```
                    case "12":
                        seacrhParameters.PriceLow = double.Parse(QueryData);
                        break;
                    case "13":
                        seacrhParameters.PriceHigh = double.Parse(QueryData);
                        break;
                    case "14":
                        seacrhParameters.Weight.Add(float.Parse(QueryData));
                        break;
                    case "15":
                        switch (QueryData)
                        {
                            case "1":
                                seacrhParameters.SortBy = SortCriteria.PriceHighToLow;
                                break;
                            case "2":
                                seacrhParameters.SortBy = SortCriteria.PriceLowToHigh;
                                break;
                            default:
                                seacrhParameters.SortBy = SortCriteria.Relevance;
                                break;
                        }
                        break;
                    default:
                        break;
                }
            }


            var searchQuery = new SearchBuilder().
              SetSearchTerm(seacrhParameters.SearchTerm).//1
              SetCompany(seacrhParameters.Company).//2
              SetScreenResolution(seacrhParameters.ScreenResolution).//3
              SetTypeName(seacrhParameters.TypeName).//4
              SetInch(seacrhParameters.Inches).//5
              SetCPU(seacrhParameters.CPU).//6
              SetSpeed(seacrhParameters.Speed).//7
              SetMemory(seacrhParameters.Memory).//8
              SetRam(seacrhParameters.Ram).//9
              SetGPU(seacrhParameters.GPU).//10
              SetOS(seacrhParameters.OS).//11
              SetPriceLow(seacrhParameters.PriceLow).//12
              SetPriceHigh(seacrhParameters.PriceHigh).//13
              SetWeight(seacrhParameters.Weight)//14
              .SetSortBy(SortCriteria.PriceHighToLow)//15
            .Build(db);
            return Ok(searchQuery);
        }
```

B7: Thay đổi File WebApiConfig.cs thành

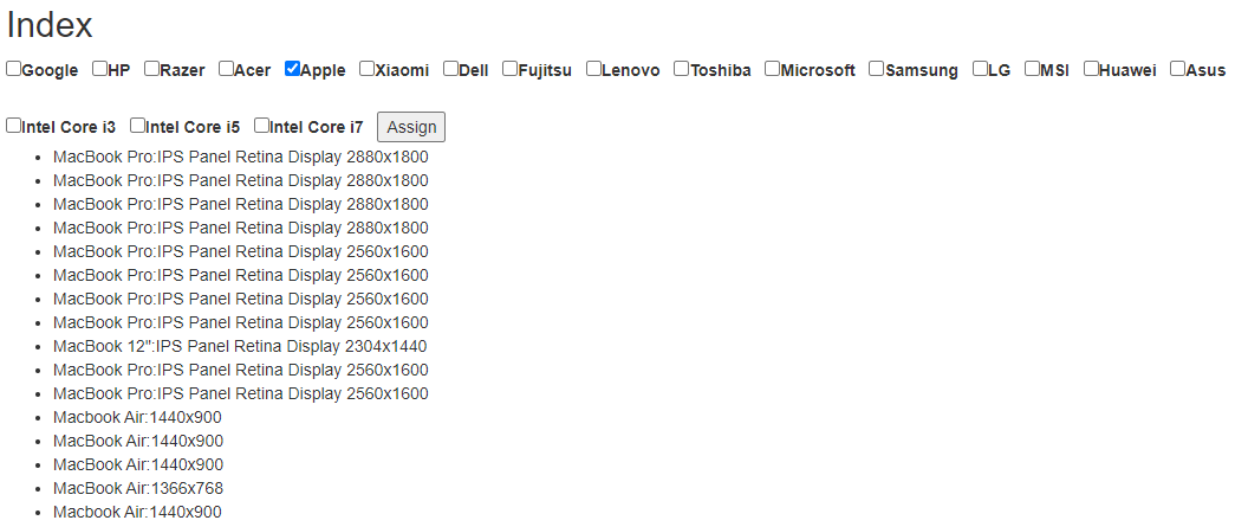
```
public static void Register(HttpConfiguration config)
{
    config.MapHttpAttributeRoutes();

    config.Routes.MapHttpRoute(
        name: "DefaultApi",
        routeTemplate: "api/{controller}/{features_hash}",
        defaults: new { features_hash = RouteParameter.Optional }
    );
```

B8: Quay lại View Index của Controller Laptop và làm như sau:

```
@model IEnumerable<TMDT4.Models.laptop>
@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>
@{Html.RenderAction("getCompany", "Laptop"); }
<br />
@{Html.RenderAction("getCPU", "Laptop"); }
<input type="button" name="name" onclick="find();" value="Assign" />
<ul id="laptop">
</ul>
<script src="~/Scripts/jquery-3.4.1.min.js"></script>
<script>

    var uri = "../../api/LaptopAPI"
    $(document).ready(function () {
        $.getJSON(uri).done(function (data) {
            $.each(data, function (key, item) {
                $('<li>', { text: formatItem(item) }).appendTo($('#laptop'));
            });
        });

    });
    function formatItem(item) {
        return item.Product + ':' + item.ScreenResolution;
    }
    function find() {
        $('#laptop').text('');
        var idString = "";
        var idCompany = $('#2').find('input[type=checkbox]');
        for (var i = 0; i < idCompany.length; i++) {
            if (idCompany[i].checked) {
                if (idString == "") {
                    idString = "2-" + idCompany[i].name;
                }
                else {
                    idString = idString + "_2-" + idCompany[i].name;
                }
            }
        }
        $.getJSON(uri + "/" + idString).done(function (data) {
            $.each(data, function (key, item) {
                $('<li>', { text: formatItem(item) }).appendTo($('#laptop'));
            });
        }).fail(function (jqXHR,textStatus,err) {
            $('#laptop').text('Error:' + err);
        });

    }
</script>
```

Kết quả khi vào đường dẫn: /web/Laptop/Index

Truy vấn thử: và bấm nút Assign



Cấu trúc truy vấn /api/LaptopAPI/2-Apple_2-Hp