HomeController.cs

```
using Practice.DB;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;
using Practice.Models;

namespace Practice.Controllers
{
    public class HomeController : Controller
    {
        //
        // GET: /Home/
        AppDbContext ctx;
        public HomeController()
        {
            ctx = new AppDbContext();
        }
        public ActionResult Index()
        {
            var model = ctx.Products.ToList();
            return View(model);
        }

        //
        // GET: /Home/Details/5
        public ActionResult Details(int id)
        {
            return View();
        }

        //
        // GET: /Home/Create
        public ActionResult Create()
        {
            return View();
        }

        //
        // POST: /Home/Create
        [HttpPost]
        public ActionResult Create(Products0029 cnt)
        {
            try
            {
                // TODO: Add insert logic here
                ctx.Products.Add(cnt);
                ctx.SaveChanges();
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Home/Edit/5
        public ActionResult Edit(int id)
        {
            return View();
        }

        //
        // POST: /Home/Edit/5
        [HttpPost]
        public ActionResult Edit(int id, Products0029 cnt)
        {
            try
            {
                // TODO: Add update logic here
                ctx.Entry(cnt).State = System.Data.Entity.EntityState.Modified;
                ctx.SaveChanges();
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }

        //
        // GET: /Home/Delete/5
        public ActionResult Delete(int id)
        {
            var keyid = ctx.Products.Find(id);
            return View(keyid);
        }

        //
        // POST: /Home/Delete/5
        [HttpPost]
        public ActionResult Delete(int id, Products0029 cnt)
        {
            var keyid = ctx.Products.Find(id);

            try
            {
                // TODO: Add delete logic here
                ctx.Products.Remove(keyid);
                ctx.SaveChanges();
                return RedirectToAction("Index");
            }
            catch
            {
                return View();
            }
        }
        private bool IsUserLoggedIn()
        {
            return Session["Username"] != null && Session["Password"] != null;
        }

        public ActionResult Login()
        {
            if (IsUserLoggedIn())
            {
                return RedirectToAction("Index");
            }

            return View();
        }
        [HttpPost]
        public ActionResult CheckAnswer(string Username, string Password)
        {
            var correctname = "aliyan";
            var correctpass = "12";
            if (Username == correctname && Password == correctpass)
            {
                Session["Username"] = "Aliyan";
                Session["Password"] = "12";
                return RedirectToAction("Index");

            }
            else
            {
                return View();
            }
        }
        private List<Products0029> GetCart()
        {
            if (Session["Cart"] == null)
            {
                Session["Cart"] = new List<Products0029>();
            }

            return (List<Products0029>)Session["Cart"];
        }

        private void SetCart(List<Products0029> cart)
        {
            Session["Cart"] = cart;
        }
        public ActionResult AddToCart(int id)
        {
            if (!IsUserLoggedIn())
            {
                return RedirectToAction("Login");
            }

            var product = ctx.Products.Find(id);

            if (product != null)
            {
                var cart = GetCart();
                cart.Add(product);
                SetCart(cart);
            }

            return RedirectToAction("Index");
        }

        // GET: Home/RemoveFromCart/5
        public ActionResult RemoveFromCart(int id)
        {
            if (!IsUserLoggedIn())
            {
                return RedirectToAction("Login");
            }

            var cart = GetCart();
            var product = cart.FirstOrDefault(p => p.Id == id);

            if (product != null)
            {
                cart.Remove(product);
                SetCart(cart);
            }

            return RedirectToAction("Cart");
        }

        // GET: Home/Cart
        public ActionResult Cart()
        {
            if (!IsUserLoggedIn())
            {
                return RedirectToAction("Login");
            }

            var cart = GetCart();
            return View(cart);
        }
    }
}
```

Products0029.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace Practice.Models
{
    public class Products0029
    {
        [Key]
        public int Id { get; set; }
        public int Code { get; set; }
        public string Name { get; set; }
        public int Price { get; set; }
        public string Color { get; set; }
        public string Dimension { get; set; }
        public string Description { get; set; }
        public string Brand { get; set; }
    }
}
```

AppDbContext.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Practice.Models;
using System.Data.Entity;

namespace Practice.DB
{
    public class AppDbContext : DbContext
    {
        public AppDbContext()
            : base("name=conn")
        {
        }
        public DbSet<Products0029> Products { get; set; }

        public static void SeedData(AppDbContext dbContext)
        {
            var products123 = new List<Products0029>
    {
        new Products0029
        {
            Id = 1,
            Code = 2565,
            Name = "ABC",
            Price = 200,
            Color = "Green",
            Dimension = "2x5x3",
            Description = "Etc",
            Brand = "Logo"
        },
        new Products0029
        {
            Id = 2,
            Code = 2566,
            Name = "EFG",
            Price = 250,
            Color = "Red",
            Dimension = "5x5x5",
            Description = "Etc",
            Brand = "Service"
        },
        new Products0029
        {
            Id = 3,
            Code = 2567,
            Name = "HIJ",
            Price = 275,
            Color = "Blue",
            Dimension = "4x5x3",
            Description = "Etc",
            Brand = "Bata"
        }
    };

            dbContext.Products.AddRange(products123);
            dbContext.SaveChanges();
        }
    }
}
```

install-package EntityFramework -version 6.4

Web.config

```
<connectionStrings>

  <add name="conn" connectionString="Data Source=AdvWebClass.mssql.somee.com;Initial Catalog=AdvWebClass;
User ID=hammadali562002_SQLLogin_1;Password=7ittcy6x85" providerName="System.Data.SqlClient"/>

</connectionStrings>
```

enable-migrations

```
    AutomaticMigrationsEnabled = true;
```

Add-Migration

update-database

Index.cshtml

```
@model IEnumerable<Practice.Models.Products0029>

@{
  ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
  @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
  <tr>
    <th>
      @Html.DisplayNameFor(model => model.Code)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Name)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Price)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Color)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Dimension)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Description)
    </th>
    <th>
      @Html.DisplayNameFor(model => model.Brand)
    </th>
    <th></th>
  </tr>

@foreach (var item in Model) {
  <tr>
    <td>
      @Html.DisplayFor(modelItem => item.Code)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.Name)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.Price)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.Color)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.Dimension)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.Description)
    </td>
    <td>
      @Html.DisplayFor(modelItem => item.Brand)
    </td>
    <td>
      @Html.ActionLink("Edit", "Edit", new { id = item.Id }) |
      @Html.ActionLink("Details", "Details", new { id = item.Id }) |
      @Html.ActionLink("Delete", "Delete", new { id = item.Id })
      @Html.ActionLink("AddToCart", "AddToCart", new { id = item.Id }) |
      @Html.ActionLink("RemoveFromCart", "RemoveFromCart", new { id = item.Id }) |
      @Html.ActionLink("Cart", "Cart", new { id = item.Id })
    </td>
  </tr>
}

</table>
```

Login.cshtml

```
@{
  ViewBag.Title = "Login";
}

<h2>Login</h2>
<form action="/Home/CheckAnswer" method="post">
  <label for="username">Username:</label>
  <input type="text" name="username" />
  <label for="password">Password:</label>
  <input type="password" name="password" />
  <input type="submit" value="Submit" />
</form>
```

Cart.cshtml

```
@model List<Practice.Models.Products0029>

@{
  ViewBag.Title = "Cart";
}

<h2>Cart</h2>
@if (Model != null && Model.Count > 0)
{
  <table class="table">
    <thead>
      <tr>
        <th>Product Name</th>
        <th>Price</th>
        <th></th>
      </tr>
    </thead>
    <tbody>
      @foreach (var item in Model)
      {
        <tr>
          <td>@item.Name</td>
          <td>@item.Price</td>
          <td>
            @Html.ActionLink("Remove", "RemoveFromCart", new { id = item.Id })
          </td>
        </tr>
      }
    </tbody>
  </table>

  <p>
    @Html.ActionLink("Continue Shopping", "Index")
  </p>
}
else
{
  <p>Your cart is empty.</p>
  <p>
    @Html.ActionLink("Continue Shopping", "Index")
  </p>
}
```

CheckAnswer.cshtml

```
@{
  ViewBag.Title = "CheckAnswer";
}
<center>
  <h2 style="color:red;">Username Or Password Incorrect</h2>
  @Html.ActionLink("Return to Login Page", "Login")
</center>
```

RemoveFromCart.cshtml

```
@{
  ViewBag.Title = "Remove From Cart";
  Layout = null;
}

<h2>Remove From Cart</h2>

<p>Product removed from the cart successfully.</p>

@Html.ActionLink("Back to Cart", "Cart")
```

Data Annotations

```
using System;
using System.ComponentModel.DataAnnotations;

namespace Practice.Models
{
   public class Products0029
   {
      [Key]
      public int Id { get; set; }

      [Required(ErrorMessage = "Code is required.")]
      [Range(1000, 9999, ErrorMessage = "Code must be a 4-digit number.")]
      public int Code { get; set; }

      [Required(ErrorMessage = "Name is required.")]
      [StringLength(50, ErrorMessage = "Name cannot exceed 50 characters.")]
      public string Name { get; set; }

      [Required(ErrorMessage = "Price is required.")]
      [Range(0, int.MaxValue, ErrorMessage = "Price must be a non-negative number.")]
      public int Price { get; set; }

      [Required(ErrorMessage = "Color is required.")]
      public string Color { get; set; }

      [Required(ErrorMessage = "Dimension is required.")]
      [RegularExpression(@"^\d+(\.\d{1,2})?$", ErrorMessage = "Positive number with up to 2 decimal places.")]
      public string Dimension { get; set; }

      [StringLength(200, ErrorMessage = "Description cannot exceed 200 characters.")]
      public string Description { get; set; }

      [Required(ErrorMessage = "Brand is required.")]
      [StringLength(50, ErrorMessage = "Brand cannot exceed 50 characters.")]
      public string Brand { get; set; }
   }
}
```

1. **DataType:** Specifies the type of data expected for a property. For example, you can use `
[DataType(DataType.EmailAddress)]` to validate that a property contains a valid email address.
2. **RegularExpression:** Defines a regular expression pattern that the property value must match. For instance, `
[RegularExpression(@"^[A-Za-z\s]+$", ErrorMessage = "Name must only contain letters and spaces.")]`
ensures that a name property contains only letters and spaces.
3. **StringLength:** Sets the maximum and minimum lengths for a string property. For example, `
[StringLength(100, MinimumLength = 5, ErrorMessage = "Description must be between 5 and 100 characters.")]`
specifies that a description property should be between 5 and 100 characters long.
4. **Range:** Specifies the range of acceptable values for a numeric property. For instance, `
[Range(1, 100, ErrorMessage = "Value must be between 1 and 100.")]` ensures that a property falls within the specified range.
5. **Required:** Marks a property as required, meaning it must have a value. For example, `
[Required(ErrorMessage = "Please provide a name.")]` ensures that a name property is not left empty.
6. **Compare:** Compares the value of a property with another property in the model. For instance, `
[Compare("Password", ErrorMessage = "Passwords do not match.")]` can be used to confirm that a "Confirm Password"
field matches the "Password" field.
7. **Display:** Specifies the display name for a property that will be shown in the user interface.
For example, `[Display(Name = "Product Name")]` changes the display name of a property to "Product Name"
instead of using the default property name.

WebController.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace WebApplication1.Controllers
{
   public class WebController : Controller
   {
      //
      // GET: /Web/
      public ActionResult Index()
      {
         return View();
      }


      [HttpPost]
      public ActionResult Index(HttpPostedFileBase updfile)
      {
         //---------------Read Text File ----------------------------
         //string FileContent = "";
         //using (var reader = new StreamReader(updfile.InputStream))
         //{
         //    FileContent = reader.ReadToEnd();
         //}
         //string updData = FileContent.ToUpper();
         //string NewFileName = "M_TextFile.txt";
         //string path = Server.MapPath("~/UploadedFiles/") + NewFileName;

         //System.IO.File.WriteAllText(path, updData);
         //ViewBag.Content = updData;


         //string path = Server.MapPath("~/UploadedFiles/") + updfile.FileName;
         //updfile.SaveAs(path);


         //---------------------- Read CSV File -----------------------
         var csvData = new List<string[]>();
         using (var reader = new StreamReader(updfile.InputStream))
         {
            while (!reader.EndOfStream)
            {
               string line = reader.ReadLine();
               string[] row = line.Split(',');

               csvData.Add(row);
            }

         }

         ViewBag.Content = csvData;

         return View();
      }

       }
}
```

index.cshtml

```
@{
   ViewBag.Title = "Index";
}

<h2>Index</h2>

<form method="post" enctype="multipart/form-data">
   <label>Upload File</label>
   <input type="file" name="updfile" />
   <br />
   <input type="submit" name="Submit" />
</form>
<div>
   @ViewBag.Content
</div>
```

Order.cshtml

```
@model IEnumerable<MyEcommerceAdmin.Models.Order>

@{
    ViewBag.Title = "Order";
    Layout = "~/Views/Shared/_Layout.cshtml";
}

<link rel="stylesheet" href="https://cdn.datatables.net/1.13.5/css/jquery.dataTables.min.css" />

<div class="row">
    <div class="col-md-12">
        <table id="example" class="table table-bordered table-hover table-striped">
            <thead>
                <tr>
                    <th>Order ID</th>
                    <th>Customer ID</th>
                    <th>Payment ID</th>
                    <th>Shipping ID</th>
                    <th>Discount</th>
                    <th>Taxes</th>
                    <th>Total Amount</th>
                    <th>Order Date</th>
                    <th>Action</th>
                </tr>
            </thead>
            <tbody>
                @foreach (var item in Model)
                {
                    <tr>
                        <td>@Html.DisplayFor(modelItem => item.OrderID)</td>
                        <td>@Html.DisplayFor(modelItem => item.Customer.CustomerID)</td>
                        <td>@Html.DisplayFor(modelItem => item.Payment.PaymentID)</td>
                        <td>@Html.DisplayFor(modelItem => item.ShippingDetail.ShippingID)</td>
                        <td>@Html.DisplayFor(modelItem => item.Discount)</td>
                        <td>@Html.DisplayFor(modelItem => item.Taxes)</td>
                        <td>@Html.DisplayFor(modelItem => item.TotalAmount)</td>
                        <td>@Html.DisplayFor(modelItem => item.OrderDate)</td>
                        <td>
                            @Html.ActionLink("View Details", "Details", new { id = item.OrderID }, new { @class = "btn btn-info" })
                        </td>
                    </tr>
                }
            </tbody>
        </table>
    </div>
</div>

@section Scripts {
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdn.datatables.net/1.13.5/js/jquery.dataTables.min.js"></script>
    <script>
        $(document).ready(function() {
            $('#example').DataTable();
        });
    </script>
}
```

Sure! Here are some examples of using DataTables with jQuery in a web application:

1. Include the necessary libraries:
```html
<!-- jQuery library -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<!-- DataTables CSS -->
<link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.11.0/css/jquery.dataTables.min.css">

<!-- DataTables JS -->
<script src="https://cdn.datatables.net/1.11.0/js/jquery.dataTables.min.js"></script>
```

2. Initialize and configure a DataTable:
```javascript
$(document).ready(function() {
  $('#myTable').DataTable({
    // Configuration options
  });
});
```

3. Create an HTML table with an ID:
```html
<table id="myTable">
  <thead>
    <tr>
      <th>Column1</th>
      <th>Column2</th>
      <th>Column3</th>
    </tr>
  </thead>
  <tbody>
    <!-- Table rows will be added dynamically -->
  </tbody>
</table>
```

4. Load data into the DataTable dynamically (e.g., using an AJAX call):
```javascript
$(document).ready(function() {
  $('#myTable').DataTable({
    ajax: {
      url: 'data.json', // URL to fetch data from
      dataSrc: 'data' // Specify the data source property
    },
    columns: [
      { data: 'Column1' },
      { data: 'Column2' },
      { data: 'Column3' }
    ]
  });
});
```
In this example, the DataTable is populated with data from a JSON file (data.json). The JSON file should have a property named 'data' that contains an array of objects representing the table rows.

5. Refresh or redraw the DataTable:
```javascript
// Refresh the DataTable with new data (e.g., after modifying the underlying data source)
$('#myTable').DataTable().ajax.reload();

// Redraw the DataTable (e.g., after modifying the table structure)
$('#myTable').DataTable().draw();
```

6. Destroy the DataTable:
```javascript
// Destroy the DataTable and restore the original HTML table
$('#myTable').DataTable().destroy();
```

These examples demonstrate the basic usage of DataTables with jQuery. You can explore the DataTables documentation for more advanced features and customization options: https://datatables.net/.

Feature enable / disable

```
new DataTable('#example', {
   info: false,
   ordering: false,
   paging: false
});
```

Default ordering (sorting)

```
 new DataTable('#example', {
    order: [[3, 'desc']]
 });
```

Multi-column ordering

```
new DataTable('#example', {
   columnDefs: [
      {
         targets: [0],
         orderData: [0, 1]
      },
      {
         targets: [1],
         orderData: [1, 0]
      },
      {
         targets: [4],
         orderData: [4, 0]
      }
   ]
});
```

Multiple tables

```
 new DataTable('table.display');
```

Hidden columns

```
 new DataTable('#example', {
    columnDefs: [
       {
          target: 2,
          visible: false,
          searchable: false
       },
       {
          target: 3,
          visible: false
       }
    ]
 });
```

State saving

```
new DataTable('#example', {
   stateSave: true
});
```

State saving

```
new DataTable('#example', {
   stateSave: true
});
```

Alternative pagination

```
new DataTable('#example', {
   pagingType: 'full_numbers'
});
```

numbers
Page number buttons only
simple
Previous' and 'Next' buttons only
simple_numbers
Previous' and 'Next' buttons,
plus page numbers
full
First'Previous', 'Next' and 'Last' buttons
full_numbers
First', 'Previous', 'Next' and 'Last' buttons,
plus page numbers
first_last_numbers
First' and 'Last' buttons, plus page numbers

Scroll - vertical

```
new DataTable('#example', {
   paging: false,
   scrollCollapse: true,
   scrollY: '200px'
});
```

Scroll - vertical, dynamic height

```
new DataTable('#example', {
   paging: false,
   scrollCollapse: true,
   scrollY: '50vh'
});
```

Scroll - horizontal

```
new DataTable('#example', {
   scrollX: true
});
```

Scroll - horizontal and vertical

```
new DataTable('#example', {
   scrollX: true,
   scrollY: 200
});
```

Data rendering

```
new DataTable('#example', {
   ajax: '../ajax/data/objects_salary.txt',
   columns: [
      {
         data: 'name'
      },
      {
         data: 'position',
         render: function (data, type) {
            if (type === 'display') {
               let link = 'http://datatables.net';

               if (data[0] < 'H') {
                  link = 'http://cloudtables.com';
               }
               else if (data[0] < 'S') {
                  link = 'http://editor.datatables.net';
               }

               return '<a href="' + link + '">' + data + '</a>';
            }

            return data;
         }
      },
      {
         className: 'f32', // used by world-flags-sprite library
         data: 'office',
         render: function (data, type) {
            if (type === 'display') {
               let country = '';

               switch (data) {
                  case 'Argentina':
                     country = 'ar';
                     break;
                  case 'Edinburgh':
                     country = '_Scotland';
                     break;
                  case 'London':
                     country = '_England';
                     break;
                  case 'New York':
                  case 'San Francisco':
                     country = 'us';
                     break;
                  case 'Sydney':
                     country = 'au';
                     break;
                  case 'Tokyo':
                     country = 'jp';
                     break;
               }

               return '<span class="flag ' + country + '"></span> ' + data;
            }

            return data;
         }
      },
      {
         data: 'extn',
         render: function (data, type, row, meta) {
            return type === 'display'
               ? '<progress value="' + data + '" max="9999"></progress>'
               : data;
         }
      },
      {
         data: 'start_date'
      },
      {
         data: 'salary',
         render: function (data, type) {
            var number = DataTable.render
               .number(',', '.', 2, '$')
               .display(data);

            if (type === 'display') {
               let color = 'green';
               if (data < 250000) {
                  color = 'red';
               }
               else if (data < 500000) {
                  color = 'orange';
               }

               return `<span style="color:${color}">${number}</span>`;
            }

            return number;
         }
      }
   ]
});
```

Language

```
new DataTable('#example', {
    language: {
        decimal: ',',
        thousands: '.'
    }
});
```

DOM / jQuery events

```
let table = new DataTable('#example');

table.on('click', 'tbody tr', function () {
    let data = table.row(this).data();

    alert('You clicked on ' + data[0] + "'s row");
});
```

POST data

```
new DataTable('#example', {
    ajax: {
        url: 'scripts/post.php',
        type: 'POST'
    },
    columns: [
        { data: 'first_name' },
        { data: 'last_name' },
        { data: 'position' },
        { data: 'office' },
        { data: 'start_date' },
        { data: 'salary' }
    ],
    processing: true,
    serverSide: true
});
```

Ajax data source (arrays)

```
new DataTable('#example', {
    ajax: 'data/arrays.txt'
});
```

Server-side processing

```
new DataTable('#example', {
    ajax: 'scripts/server_processing.php',
    processing: true,
    serverSide: true
});
```

```
@section Scripts {
    <script>
        $(document).ready(function() {
            $('#example').DataTable({
                processing: true,
                serverSide: true,
                ajax: {
                    url: '@Url.Action("GetOrders", "YourControllerName")',
                    type: 'POST'
                },
                columns: [
                    { data: 'OrderID' },
                    { data: 'Customer.CustomerID' },
                    { data: 'Payment.PaymentID' },
                    { data: 'ShippingDetail.ShippingID' },
                    { data: 'Discount' },
                    { data: 'Taxes' },
                    { data: 'TotalAmount' },
                    { data: 'OrderDate' },
                    {
                        data: 'OrderID',
                        render: function(data, type, row) {
                            return '<a href="' + '@Url.Action("Details", "
YourControllerName")' +
'?id=' + data + '" class="btn btn-info">View Details</a>';
                        }
                    }
                ]
            });
        });
    </script>
}
```

Return key to search

```
new DataTable('#example', {
    ajax: 'scripts
/server_processing.php',
    processing: true,
    search: {
        return: true
    },
    serverSide: true
});
```

Ajax data source (objects)

```
new DataTable('#example', {
    ajax: 'data/objects.txt',
    columns: [
        { data: 'name' },
        { data: 'position' },
        { data: 'office' },
        { data: 'extn' },
        { data: 'start_date' },
        { data: 'salary' }
    ]
});
```

Automatic addition of row ID attributes

```
new DataTable('#example', {
    ajax: 'scripts/ids-objects.php',
    columns: [
        { data: 'first_name' },
        { data: 'last_name' },
        { data: 'position' },
        { data: 'office' },
        { data: 'start_date' },
        { data: 'salary' }
    ],
    processing: true,
    serverSide: true
});
```

Custom HTTP variables

```
new DataTable('#example', {
    ajax: {
        url: 'scripts/server_processing.php',
        data: function (d) {
            d.myKey = 'myValue';
            // d.custom = $('#myInput').val();
            // etc
        }
    },
    processing: true,
    serverSide: true
});
```

Nested object data (arrays)

```
new DataTable('#example', {
    ajax: 'data/objects_subarrays.txt',
    columns: [
        { data: 'name[, ]' },
        { data: 'hr.0' },
        { data: 'office' },
        { data: 'extn' },
        { data: 'hr.2' },
        { data: 'hr.1' }
    ]
});
```

Nested object data (objects)

```
new DataTable('#example', {
    ajax: 'data/objects_deep.txt',
    columns: [
        { data: 'name' },
        { data: 'hr.position' },
        { data: 'contact.0' },
        { data: 'contact.1' },
        { data: 'hr.start_date' },
        { data: 'hr.salary' }
    ],
    processing: true
});
```

Object data source

```
new DataTable('#example', {
    ajax: 'scripts/ids-objects.php',
    columns: [
        { data: 'first_name' },
        { data: 'last_name' },
        { data: 'position' },
        { data: 'office' },
        { data: 'start_date' },
        { data: 'salary' }
    ],
    processing: true,
    serverSide: true
});
```

Row details

```
function format(d) {
   return (
      'Full name: ' +
      d.first_name +
      ' ' +
      d.last_name +
      '<br>' +
      'Salary: ' +
      d.salary +
      '<br>' +
      'The child row can contain any data you wish, including links
, images, inner tables etc.'
   );
}

const table = new DataTable('#example', {
   ajax: 'scripts/ids-objects.php',
   columns: [
      {
         class: 'dt-control',
         orderable: false,
         data: null,
         defaultContent: ''
      },
      { data: 'first_name' },
      { data: 'last_name' },
      { data: 'position' },
      { data: 'office' }
   ],
   order: [[1, 'asc']],
   processing: true,
   serverSide: true
});

// Array to track the ids of the details displayed rows
const detailRows = [];

table.on('click', 'tbody td.dt-control', function () {
   let tr = event.target.closest('tr');
   let row = table.row(tr);
   let idx = detailRows.indexOf(tr.id);

   if (row.child.isShown()) {
      tr.classList.remove('details');
      row.child.hide();

      // Remove from the 'open' array
      detailRows.splice(idx, 1);
   }
   else {
      tr.classList.add('details');
      row.child(format(row.data())).show();

      // Add to the 'open' array
      if (idx === -1) {
         detailRows.push(tr.id);
      }
   }
});

// On each draw, loop over the `detailRows` array and show any child rows
table.on('draw', () => {
   detailRows.forEach((id, i) => {
      let el = document.querySelector('#' + id + ' td.dt-control');

      if (el) {
         el.dispatchEvent(new Event('click', { bubbles: true }));
      }
   });
});
```

Deferred loading of data

```
new DataTable('#example', {
   ajax: 'scripts/server_processing.php',
   deferLoading: 57,
   processing: true,
   serverSide: true
});
```