Using AJAX In ASP.NET MVC

What is Ajax?

As we all know, AJAX means Asynchronous JavaScript and XML. It is a client-side script that communicates to and from a server/database without the need for a postback or a complete page refresh. The Ajax speeds up response time.

In other words, Ajax is the method of exchanging data with a server, and updating parts of a web page, without reloading the entire page.

Benefits of Ajax

* Callbacks
* Making Asynchronous Calls
* User-Friendly
* Improve the speed, performance and usability of a web application

**Implementation of Ajax can be done in two way in ASP.Net Application**

* using Update Panel and,
* using jQuery

**What Advances have Been Made to Ajax?**

JavaScript is the client-side scripting language and XML is a mark-up language to define data. And we have, JSON(JavaScript Object Notation) as another mark-up language to define data as well. JSON is much easier to use with JavaScript than XML. After the combination of JavaScript and Ajax, the XML Web Services are being replaced by JSON Web Services.

Another major advance to JavaScript and Ajax is the JavaScript object library called jQuery, which is the free, open-source software. It is a wrapper for JavaScript. jQuery is used to write the JavaScript to navigate and manipulate a page and make asynchronous Ajax callbacks.

Hence, Ajax callbacks have become standard programming practices by using jQuery and JSON Web Services for designing and developing web applications.

Demonstration: Implementation of Ajax using jQuery

**Step 1**

Create a new Project and choose ASP.NET MVC web application.

**Step 2**

Just Ignore the built-in Models and Controllers and make your own model.

Here I am creating a model called "Students" having properties studentID, studentName and studentAddress as shown below,

public class Student

{

[Key]

public int studentID { get; set; }

[Required]

public string studentName { get; set; }

[Required]

public string studentAddress { get; set; }

}

C#

Copy

Add "using System.ComponentModel.DataAnnotations;" Attributes to Validate Model Data and then build the project once.

**Step 3**

Let's create another model by inheriting the DbContext. It is a class that manages all the database operations, like database connection, and manages various entities of the Entity Model.

We can also say DbContext is a wrapper of ObjectContext. So, DbContext is a lightweight version of the ObjectContext and is exposes only the common features that are really required in programming.

Here I am creating a model called "StudentContext" as shown below,

public class StudentContext : DbContext

{

public DbSet<Student> Students { get; set; }

}

C#

Copy

Add "using System.Data.Entity;" that provides access to the core functionality of the entity framework.

**Step 4**

Now create a controller to written the code for inserting data into database, displaying data into view.

Here I am creating "Student" controller. Inside the Controller, I am creating an object of StudentContext for inserting and retrieving data from database. Also add the necessary namespace.

StudentContext context = new StudentContext();

C#

Copy

**Step 5**

Now I'm creating the action methods for Inserting and retrieving the data to/from the database.

Here I am creating an [HttpPost] action method "createStudent" for inserting the JSON-Formatted data to database. I am Using [HttpPost] attribute to Save/Post the data as below:

[HttpPost]

public ActionResult createStudent(Student std)

{

context.Students.Add(std);

context.SaveChanges();

string message = "SUCCESS";

return Json(new { Message = message, JsonRequestBehavior.AllowGet });

}

C#

Copy

Similarly, I am also creating JSON method "getStudent" to retrieve data from database and returning raw JSON data to be consumed by JavaScript in view as shown below,

public JsonResult getStudent(string id)

{

List<Student>students = new List<Student>();

students = context.Students.ToList();

return Json(students, JsonRequestBehavior.AllowGet);

}

C#

Copy

**Step 6**

Now add a view to display the data and data inserting field. Here, I am adding a view named "Index.cshtml". Then, write the HTML codes for making the data input field, submit button and also for displaying the data in same page. You can use bootstrap classes for designing. Here I am using bootstrap. My designing code is as below:

<div class="col-md-12">

<div class="panel panel-primary">

<div class="panel-heading">

<h3 class="panel-title">Please enter the details below.</h3>

</div>

<div class="panel-body">

<div class="form-group col-md-5">

<label>Student Name</label>

<input type="text" name="StudentName" id="StudentName" class="form-control" placeholder="Enter Student Name" required="" />

</div>

<div class="form-group col-md-5">

<label>Student Address</label>

<input type="text" name="StudentAddress" id="StudentAddress" class="form-control" placeholder="Enter Student Address" required="" />

</div>

<div class="form-group col-md-1">

<div style="float: right; display:inline-block;">

<input class="btn btn-primary" name="submitButton" id="btnSave" value="Save" type="button">

</div>

</div>

</div>

</div><hr />

<table id="tblStudent" class="table table-bordered table-striped table-responsive table-hover">

<thead>

<tr>

<th align="left" class="productth">ID</th>

<th align="left" class="productth">Student Name</th>

<th align="left" class="productth">Student Address</th>

</tr>

</thead>

<tbody></tbody>

</table>

</div>

Markup

Copy

We can simply use the bootstrap class and call jQuery functions in ASP.NET MVC because during the project creation it will by default added to project and also linked to the template.

**Step 7**

Now I will write the script for inserting the data as well as retrieving and displaying it to the view. Just after the HTML code finished add the script shown below,

@section Scripts

{

<script type="text/javascript">

$(function () {

LoadData();

$("#btnSave").click(function () {

//alert("");

var std = {};

std.studentName = $("#StudentName").val();

std.studentAddress = $("#StudentAddress").val();

$.ajax({

type: "POST",

url: '@Url.Action("createStudent")',

data: '{std: ' + JSON.stringify(std) + '}',

dataType: "json",

contentType: "application/json; charset=utf-8",

success: function () {

// alert("Data has been added successfully.");

LoadData();

},

error: function () {

alert("Error while inserting data");

}

});

return false;

});

});

function LoadData() {

$("#tblStudent tbody tr").remove();

$.ajax({

type: 'POST',

url: '@Url.Action("getStudent")',

dataType: 'json',

data: { id: '' },

success: function (data) {

var items = '';

$.each(data, function (i, item) {

var rows = "<tr>"

+ "<td class='prtoducttd'>" + item.studentID + "</td>"

+ "<td class='prtoducttd'>" + item.studentName + "</td>"

+ "<td class='prtoducttd'>" + item.studentAddress + "</td>"

+ "</tr>";

$('#tblStudent tbody').append(rows);

});

},

error: function (ex) {

var r = jQuery.parseJSON(response.responseText);

alert("Message: " + r.Message);

alert("StackTrace: " + r.StackTrace);

alert("ExceptionType: " + r.ExceptionType);

}

});

return false;

}

</script>

}

JavaScript

Copy

We are using Ajax to refresh "tblStudent" so it refreshes only the particular table, rather than refreshing entire page. After clicking the "submitButton" the data from the input fields are taken in variable and redirected to action "createStudent" of "StudentController" to insert into the database. And, I have written a function "LoadData" to display the data from database to view continuously after inserting data. Inside "LoadData" function I am calling "getStudent" method which returns result in JSON and the JSON-Formatted data are presented in html by the underlying statements.

**Step 8**

Now, configure the Database connectionStrings in "web.config" file of the particular project as shown below,

<connectionStrings>

<add name="StudentContext" connectionString="Data Source=RAVI-KANDEL;Initial Catalog=Student;Integrated Security=True"

providerName="System.Data.SqlClient" />

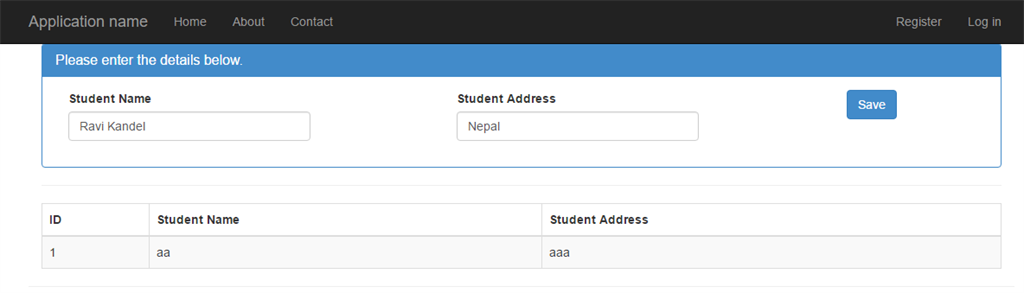
</connectionStrings>

Markup

Copy

In the above connectingString "StudentContext" is the name of class inherited from abstract Class "DbContext" in "StudentContext.cs" file. "RAVI-KANDEL" is the name of the Database Server and "Student" is the name of the Database. Change "RAVI-KANDEL" with your Database Server name. Finally, run your application and navigate to Student controller. Also you can configure "RouteConfig.cs" and set the controller to "Student" for direct navigation to the Student controller when loading the application.

You can add the records and can see added data in the table after the form-control as shown below,



Also, you can open the database server and see the data stored in database over there.

Please feel free to comment/feedback.

Happy Coding!

An Introduction to AJAX in ASP.NET: All You Need to Know

Lesson 7 of 15[By Ravikiran A S](https://www.simplilearn.com/authors/ravikiran-a-s)

Last updated on Jul 26, 2024247545



[Previous](https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/view-state-in-asp-dot-net)[Next](https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/c-sharp-arrays)

The usage of AJAX in ASP.net is fast growing these days as it helps you to create more dynamic, responsive, and better web applications. By using AJAX in ASP.net, the performance of an application is highly increased as the response time is reduced and the traffic between the server and the client is reduced as well.

You will now learn what AJAX is in ASP.net, its advantages and disadvantages, and how to transfer the data through AJAX in ASP.net.

Unleash Your Career as a Full Stack Developer!

Full Stack Developer - MERN Stack[EXPLORE COURSE](https://www.simplilearn.com/full-stack-developer-course-mern-certification-training?source=GhPreviewCTABanner)



What Is AJAX in ASP.net?

AJAX in ASP.net stands for Asynchronous JavaScript and XML. This technique is used to develop rich web applications that are interactive, responsive, and dynamic in nature.

By the method of receiving and sending data to the server in the background, the web pages are updated using AJAX in [ASP.net](https://www.simplilearn.com/tutorials/asp-dot-net-tutorial/what-is-asp-dot-net). Without even reloading the web page, you will be able to update the website part by part as well. This update of the website is done asynchronously using AJAX in ASP.net.

In AJAX, usually, multiple technologies are used together to create these dynamic web pages. Some of these technologies include -

* XSLT
* [XHTML](https://www.simplilearn.com/tutorials/html-tutorial/what-is-html)
* [CSS](https://www.simplilearn.com/tutorials/html-tutorial/html-vs-css)
* [JavaScript](https://www.simplilearn.com/reasons-to-learn-javascript-article)
* Document Object Model
* XML
* XMLHttpRequest object

The interactive animation that you see on modern web pages is mainly because of AJAX. In common web pages that do not use AJAX, one needs to reload the entire web page in order to reflect content changes in any parts or all. Now, you are going to discuss the advantages, disadvantages of AJAX in ASP.net and how to transfer data through AJAX in ASP.net.

Unleash Your Career as a Full Stack Developer!

Full Stack Developer - MERN Stack[EXPLORE COURSE](https://www.simplilearn.com/full-stack-developer-course-mern-certification-training?source=GhPreviewCTABanner)



Advantages of AJAX in ASP.net

You will explore some advantages of AJAX in ASP.net. Some of these advantages of AJAX in ASP.net are described below -

* Ajax in ASP.net improves the responsiveness and interactivity of a website.
* Ajax in ASP.net helps in reducing the traffic between server and client.
* Ajax in ASP.net reduces the cross-browser issues as it is cross-browser friendly.
* In Ajax in ASP.net, you can use a single web page or SPA to be able to handle several features and apply multi-purpose applications to it.
* In Ajax in ASP.net, you can use APIs or Application Programming Interfaces and because these work seamlessly with JavaScript and HTTP methods, it makes it an enormous advantage to building dynamic web applications.

Disadvantages of AJAX in ASP.net

Let us discuss some disadvantages of AJAX in ASP.net. Some of these disadvantages of AJAX in ASP.net are described below -

* The size of a data request is largely increased as all these data requests are URL encoded.
* It highly depends on JavaScript as it is a JavaScript built-in. Therefore, if a user disables JavaScript in the browser, then AJAX stops working.
* Indexing of an AJAX application cannot be done using Google-like search engines.
* Since all the files are downloaded on the client-side in an AJAX application, security is scarce in these applications.
* Within the AJAX, the server information is completely inaccessible.

How to Transfer the Data Through AJAX?

Now, look at the implementation of transferring data through AJAX in ASP.net.

There are two ways in which we can transfer data through AJAX -

1. Server to Client
2. Client to Server

Now, you will see how to apply and implement both of these ways.

Consider an entity - Student which contains two fields: ID and Name.

Code Snippet

public class Student

{

   public int ID { get; set; }

   public string Name { get; set; }

}

1. Server to Client

In ASP.net MVC, we add code in the Controller action. The Controller here is the Student and the action name here is GetStudList which takes a list of students as parameters.

Code Snippet

public List<Student> GetStudList()

{

    var studList = new List<Student>()

    {

         new Student { ID=1, Name="Dina"},

         new Student { ID=2, Name="Chester"}

    };

    return studList;

}

Following this, to get the list of Students, you need to send a request to the Server. To do so-

1. You need to load the jQuery library. To enable this, you are going to add a jQuery CDN (Content Delivery Network) reference.
2. To get a list of students from the Server, you have to write and execute a code block and then display it.

Code Snippet

<script type="text/javascript" src="http://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script>

<script type="text/javascript">

    function getStudents() {

        $.ajax({

            type: "GET",

            url: Student/GetStudList',

            data: {},

            contentType: "application/json; charset=utf-8",

            dataType: "json",

            beforeSend: function(){

                Show(); // the loader icon is shown here

            },

            success: function (response) {

                // we will loop over the list and then display it

                $.each(response, function (index, stud) {

                    $('#display').append('<p>Id: ' + stud.ID + '</p>' +

                                        '<p>Id: ' + stud.Name + '</p>');

                });

            },

            complete: function(){

                Hide(); // the loader icon is shown here

            },

            failure: function (jqXHR, textStatus, errorThrown) {

                alert("HTTP Status: " + jqXHR.status + "; Error Text: " + jqXHR.responseText);    //error message is displayed

            }

        });

    }

</script>

In the above code snippet, “GET” is the method type that gets data as a response parameter in the success properties. This method does the following - loops over the student list collects student data and binds the whole as a div.

In order to display the loader icon when the AJAX request is being sent to the Server and when the AJAX completes, the loader icon is hidden.

The following code does the job -

<body>

    <div id="div\_Loader" style="padding-left: 400px; top: 500px">

        <img src="loadingicon.gif" width="50px" height="50px" alt="loader icon" />

    </div>

    <div id="display">

    </div>

</body>

2. Client to Server

To send data from Server to Client using AJAX in ASP.net. Let us see how to do the same from Client to Server using AJAX in ASP.net.

To save the list of students to the database, you have to send these lists of students to the Server. To do so, you need to add the following code to the Controller’s action. The Controller is Student and the action name is SaveStudList that takes in the parameter - list of Students.

Code Snippet

public static bool SaveStudList(List<Student> studList)

{

    try

    {

        // block of statements to save student list to database

    }

    catch (Exception ex)

    {

        // if logging error arises

        return false;

    }

    return true;

To send complex objects converted to a list of students to the Server, you have to create an array and then push a JSON object into it. Following this, using the JSON.stringify( ) method, you have to convert this array into JSON string in order to effectively send the data.

<script type="text/javascript">

    $(function () {

        var results = new Array();

        var stud1 = { "ID": "1", "Name": "Mina" };

        results.push(stud1);

        var stud2 = { "ID": "2", "Name": "Shina" };

        results.push(stud2);

        var postData = { studList: results };

        $.ajax({

            url: 'WebForm1.aspx/SaveStudList',

            data: JSON.stringify(postData),

            type: 'POST',

            contentType: 'application/json',

            dataType: 'json',

            beforeSend: function(){

                Show(); // loader icon shown

            },

            success: function (result) {

                alert(result);

            },

            complete: function(){

                Hide(); // loader icon  is hidden

            },

            failure: function (jqXHR, textStatus, errorThrown) {

                alert("Status: " + jqXHR.status + "; Error: " + jqXHR.responseText);

 //error message displayed

       }

        });

    });

    $(document).ready(function () {

        $("#div\_Loader").hide();

    });

    function Show() {

        $('#div\_Loader').show();

    }

    function Hide() {

        $('#div\_Loader').hide();

    }

</script>

Through this, using $.Ajax, you can implement AJAX in jQuery.

Conclusion

AJAX is used to create dynamic web pages that do not require page reloading when any part of the whole web page content or the whole web page content is changed. The server data exchange is asynchronous in nature and AJAX in ASP.net uses multiple technologies like XSLT, XHTML, CSS, JavaScript, etc.

To master and learn more about AJAX in ASP.net and all its related technologies properly and well versed to get into full-stack development, one might consider referring and learning in-depth from various resources, study materials, and course books.

If you are interested in understanding and acquiring knowledge on AJAX in ASP.net and all its related technologies in order to become a full-stack web and desktop application developer, Simplilearn offers an exclusive [Full Stack Developer - MERN Stack](https://www.simplilearn.com/full-stack-developer-course-mern-certification-training?source=GhPreviewCoursepages) to master both backend and frontend with tools, like SpringBoot, AngularMVC, JSPs, and Hibernate to start your career as a full-stack developer.

If you are not up to enrolling yourself into the full certification course yet, Simplilearn also offers [free online skill-up courses](https://www.simplilearn.com/skillup-free-online-courses?referrer=top-nav#Software-D) in several domains, from data science and business analytics to software development, AI, and machine learning. Become a [full-stack web developer](https://www.simplilearn.com/full-stack-web-developer-article) today!

If you have any queries leave them in the comments section of this article and our experts will get back to you on the same, ASAP!