# AJAX Introduction

JAX is a developer's dream, because you can:

* Update a web page without reloading the page
* Request data from a server - after the page has loaded
* Receive data from a server - after the page has loaded
* Send data to a server - in the background

## AJAX Example Explained

### HTML Page

<!DOCTYPE html>  
<html>  
<body>  
  
<div id="demo">  
  <h2>Let AJAX change this text</h2>  
  <button type="button" onclick="loadDoc()">Change Content</button>  
</div>  
  
</body>  
</html>

The HTML page contains a <div> section and a <button>.

The <div> section is used to display information from a server.

The <button> calls a function (if it is clicked).

The function requests data from a web server and displays it:

### Function loadDoc()

function loadDoc() {  
  var xhttp = new XMLHttpRequest();  
  xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
     document.getElementById("demo").innerHTML = this.responseText;  
    }  
  };  
  xhttp.open("GET", "ajax\_info.txt", true);  
  xhttp.send();  
}

## What is AJAX?

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

AJAX is not a programming language.

AJAX just uses a combination of:

* A browser built-in XMLHttpRequest object (to request data from a web server)
* JavaScript and HTML DOM (to display or use the data)

AJAX is a misleading name. AJAX applications might use XML to transport data, but it is equally common to transport data as plain text or JSON text.

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

## How AJAX Works



* 1. An event occurs in a web page (the page is loaded, a button is clicked)
* 2. An XMLHttpRequest object is created by JavaScript
* 3. The XMLHttpRequest object sends a request to a web server
* 4. The server processes the request
* 5. The server sends a response back to the web page
* 6. The response is read by JavaScript
* 7. Proper action (like page update) is performed by JavaScript

# AJAX - The XMLHttpRequest Object

[❮ Previous](https://www.w3schools.com/xml/ajax_intro.asp)[Next ❯](https://www.w3schools.com/xml/ajax_xmlhttprequest_send.asp)

The keystone of AJAX is the XMLHttpRequest object.

## The XMLHttpRequest Object

All modern browsers support the XMLHttpRequest object.

The XMLHttpRequest object can be used to exchange data with a server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page.

## Create an XMLHttpRequest Object

All modern browsers (Chrome, Firefox, IE7+, Edge, Safari Opera) have a built-in XMLHttpRequest object.

Syntax for creating an XMLHttpRequest object:

*variable*= new XMLHttpRequest();

### Example

var xhttp = new XMLHttpRequest();

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_xmlhttp)

## Access Across Domains

For security reasons, modern browsers do not allow access across domains.

This means that both the web page and the XML file it tries to load, must be located on the same server.

The examples on W3Schools all open XML files located on the W3Schools domain.

If you want to use the example above on one of your own web pages, the XML files you load must be located on your own server.

## Old Versions of Internet Explorer (IE5 and IE6)

Old versions of Internet Explorer (IE5 and IE6) use an ActiveX object instead of the XMLHttpRequest object:

*variable*= new ActiveXObject("Microsoft.XMLHTTP");

To handle IE5 and IE6, check if the browser supports the XMLHttpRequest object, or else create an ActiveX object:

### Example

if (window.XMLHttpRequest) {  
    // code for modern browsers  
    xmlhttp = new XMLHttpRequest();  
 } else {  
    // code for old IE browsers  
    xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");  
}

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_xmlhttp_ie)

## XMLHttpRequest Object Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| new XMLHttpRequest() | Creates a new XMLHttpRequest object |
| abort() | Cancels the current request |
| getAllResponseHeaders() | Returns header information |
| getResponseHeader() | Returns specific header information |
| open(method,url,async,user,psw) | Specifies the request  method: the request type GET or POST url: the file location async: true (asynchronous) or false (synchronous) user: optional user name psw: optional password |
| send() | Sends the request to the server Used for GET requests |
| send(string) | Sends the request to the server. Used for POST requests |
| setRequestHeader() | Adds a label/value pair to the header to be sent |

## XMLHttpRequest Object Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| onreadystatechange | Defines a function to be called when the readyState property changes |
| readyState | Holds the status of the XMLHttpRequest. 0: request not initialized  1: server connection established 2: request received  3: processing request  4: request finished and response is ready |
| responseText | Returns the response data as a string |
| responseXML | Returns the response data as XML data |
| status | Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the [Http Messages Reference](https://www.w3schools.com/tags/ref_httpmessages.asp) |
| statusText | Returns the status-text (e.g. "OK" or "Not Found") |

# AJAX - Send a Request To a Server

[❮ Previous](https://www.w3schools.com/xml/ajax_xmlhttprequest_create.asp)[Next ❯](https://www.w3schools.com/xml/ajax_xmlhttprequest_response.asp)

The XMLHttpRequest object is used to exchange data with a server.

## Send a Request To a Server

To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

xhttp.open("GET", "ajax\_info.txt", true);  
xhttp.send();

|  |  |
| --- | --- |
| **Method** | **Description** |
| open(*method, url, async*) | Specifies the type of request  *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous) |
| send() | Sends the request to the server (used for GET) |
| send(*string*) | Sends the request to the server (used for POST) |

## GET or POST?

GET is simpler and faster than POST, and can be used in most cases.

However, always use POST requests when:

* A cached file is not an option (update a file or database on the server).
* Sending a large amount of data to the server (POST has no size limitations).
* Sending user input (which can contain unknown characters), POST is more robust and secure than GET.

## GET Requests

A simple GET request:

### Example

xhttp.open("GET", "demo\_get.asp", true);  
xhttp.send();

### Example

xhttp.open("GET", "demo\_get.asp?t=" + Math.random(), true);  
xhttp.send();

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_get_unique)

If you want to send information with the GET method, add the information to the URL:

### Example

xhttp.open("GET", "demo\_get2.asp?fname=Henry&lname=Ford", true);  
xhttp.send();

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_get2)

## POST Requests

A simple POST request:

### Example

xhttp.open("POST", "demo\_post.asp", true);  
xhttp.send();

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_post)

To POST data like an HTML form, add an HTTP header with setRequestHeader(). Specify the data you want to send in the send() method:

### Example

xhttp.open("POST", "demo\_post2.asp", true);  
xhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  
xhttp.send("fname=Henry&lname=Ford");

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_post2)

|  |  |
| --- | --- |
| **Method** | **Description** |
| setRequestHeader(*header, value*) | Adds HTTP headers to the request  *header*: specifies the header name *value*: specifies the header value |

## The url - A File On a Server

The url parameter of the open() method, is an address to a file on a server:

xhttp.open("GET", "ajax\_test.asp", true);

The file can be any kind of file, like .txt and .xml, or server scripting files like .asp and .php (which can perform actions on the server before sending the response back).

## Asynchronous - True or False?

Server requests should be sent asynchronously.

The async parameter of the open() method should be set to true:

xhttp.open("GET", "ajax\_test.asp", true);

By sending asynchronously, the JavaScript does not have to wait for the server response, but can instead:

* execute other scripts while waiting for server response
* deal with the response after the response is ready

## The onreadystatechange Property

With the XMLHttpRequest object you can define a function to be executed when the request receives an answer.

The function is defined in the **onreadystatechange** property of the XMLHttpResponse object:

### Example

xhttp.onreadystatechange = function() {  
  if (this.readyState == 4 && this.status == 200) {  
    document.getElementById("demo").innerHTML = this.responseText;  
  }  
};  
xhttp.open("GET", "ajax\_info.txt", true);  
xhttp.send();

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_first)

You will learn more about onreadystatechange in a later chapter.

## Synchronous Request

To execute a synchronous request, change the third parameter in the open() method to false:

xhttp.open("GET", "ajax\_info.txt", false);

Sometimes async = false are used for quick testing. You will also find synchronous requests in older JavaScript code.

Since the code will wait for server completion, there is no need for an onreadystatechange function:

### Example

xhttp.open("GET", "ajax\_info.txt", false);  
xhttp.send();  
document.getElementById("demo").innerHTML = xhttp.responseText;

Synchronous XMLHttpRequest (async = false) is not recommended because the JavaScript will stop executing until the server response is ready. If the server is busy or slow, the application will hang or stop.

Synchronous XMLHttpRequest is in the process of being removed from the web standard, but this process can take many years.

Modern developer tools are encouraged to warn about using synchronous requests and may throw an InvalidAccessError exception when it occurs.

# AJAX - Server Response

[❮ Previous](https://www.w3schools.com/xml/ajax_xmlhttprequest_send.asp)[Next ❯](https://www.w3schools.com/xml/ajax_xmlfile.asp)

## The onreadystatechange Property

The **readyState** property holds the status of the XMLHttpRequest.

The **onreadystatechange** property defines a function to be executed when the readyState changes.

The **status** property and the **statusText** property holds the status of the XMLHttpRequest object.

|  |  |
| --- | --- |
| **Property** | **Description** |
| onreadystatechange | Defines a function to be called when the readyState property changes |
| readyState | Holds the status of the XMLHttpRequest.  0: request not initialized  1: server connection established 2: request received  3: processing request  4: request finished and response is ready |
| status | 200: "OK" 403: "Forbidden" 404: "Page not found" For a complete list go to the [Http Messages Reference](https://www.w3schools.com/tags/ref_httpmessages.asp) |
| statusText | Returns the status-text (e.g. "OK" or "Not Found") |

The onreadystatechange function is called every time the readyState changes.

When readyState is 4 and status is 200, the response is ready:

### Example

function loadDoc() {  
    var xhttp = new XMLHttpRequest();  
    xhttp.onreadystatechange = function() {  
        if (this.readyState == 4 && this.status == 200) {  
            document.getElementById("demo").innerHTML =  
            this.responseText;  
       }  
    };  
    xhttp.open("GET", "ajax\_info.txt", true);  
    xhttp.send();   
}

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_first)

The onreadystatechange event is triggered four times (1-4), one time for each change in the readyState.

## Using a Callback Function

A callback function is a function passed as a parameter to another function.

If you have more than one AJAX task in a website, you should create one function for executing the XMLHttpRequest object, and one callback function for each AJAX task.

The function call should contain the URL and what function to call when the response is ready.

### Example

loadDoc("url-1", myFunction1);  
  
loadDoc("url-2", myFunction2);  
  
function loadDoc(url, cFunction) {  
  var xhttp;  
  xhttp=new XMLHttpRequest();  
  xhttp.onreadystatechange = function() {  
    if (this.readyState == 4 && this.status == 200) {  
      cFunction(this);  
    }  
 };  
  xhttp.open("GET", url, true);  
  xhttp.send();  
}  
  
function myFunction1(xhttp) {  
  // action goes here  
}   
function myFunction2(xhttp) {  
  // action goes here  
}

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_callback)

## Server Response Properties

|  |  |
| --- | --- |
| **Property** | **Description** |
| responseText | get the response data as a string |
| responseXML | get the response data as XML data |

## Server Response Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| getResponseHeader() | Returns specific header information from the server resource |
| getAllResponseHeaders() | Returns all the header information from the server resource |

## The responseText Property

The **responseText** property returns the server response as a JavaScript string, and you can use it accordingly:

### Example

document.getElementById("demo").innerHTML = xhttp.responseText;

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_first)

## The responseXML Property

The XML HttpRequest object has an in-built XML parser.

The **responseXML** property returns the server response as an XML DOM object.

Using this property you can parse the response as an XML DOM object:

### Example

Request the file [cd\_catalog.xml](https://www.w3schools.com/xml/cd_catalog.xml) and parse the response:

xmlDoc = xhttp.responseXML;  
txt = "";  
x = xmlDoc.getElementsByTagName("ARTIST");  
for (i = 0; i < x.length; i++) {  
  txt += x[i].childNodes[0].nodeValue + "<br>";  
  }  
document.getElementById("demo").innerHTML = txt;  
xhttp.open("GET", "cd\_catalog.xml", true);  
xhttp.send();

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_responsexml)

You will learn a lot more about XML DOM in the DOM chapters of this tutorial.

## The getAllResponseHeaders() Method

The **getAllResponseHeaders()** method returns all header information from the server response.

### Example

var xhttp = new XMLHttpRequest();  
xhttp.onreadystatechange = function() {  
  if (this.readyState == 4 && this.status == 200) {  
    document.getElementById("demo").innerHTML =  
    this.getAllResponseHeaders();  
  }  
};

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_header)

## The getResponseHeader() Method

The **getResponseHeader()** method returns specific header information from the server response.

### Example

var xhttp = new XMLHttpRequest();  
xhttp.onreadystatechange = function() {  
  if (this.readyState == 4 && this.status == 200) {  
    document.getElementById("demo").innerHTML =  
    this.getResponseHeader("Last-Modified");  
  }  
};  
xhttp.open("GET", "ajax\_info.txt", true);  
xhttp.send();

[Try it Yourself »](https://www.w3schools.com/xml/tryit.asp?filename=tryajax_lastmodified)

AJAX Technologies

As describe earlier, ajax is not a technology but group of inter-related technologies. [AJAX](https://www.javatpoint.com/ajax-tutorial) technologies includes:

* [HTML](https://www.javatpoint.com/html-tutorial)/[XHTML](https://www.javatpoint.com/xhtml-tutorial) and [CSS](https://www.javatpoint.com/css-tutorial)
* DOM
* [XML](https://www.javatpoint.com/xml-tutorial) or [JSON](https://www.javatpoint.com/json-tutorial)
* [XMLHttpRequest](https://www.javatpoint.com/understanding-xmlhttprequest)
* [JavaScript](https://www.javatpoint.com/javascript-tutorial)

HTML/XHTML and CSS

These technologies are used for displaying content and style. It is mainly used for presentation.

DOM

It is used for dynamic display and interaction with data.

XML or JSON

For carrying data to and from server. JSON (Javascript Object Notation) is like XML but short and faster than XML.

XMLHttpRequest

For [asynchronous communication](https://www.javatpoint.com/understanding-synchronous-vs-asynchronous) between client and server. For more visit next page.

JavaScript

It is used to bring above technologies together.

Independently, it is used mainly for client-side validation.

Understanding XMLHttpRequest

1. [XMLHttpRequest](https://www.javatpoint.com/understanding-xmlhttprequest)
2. [Properties of XMLHttpRequest](https://www.javatpoint.com/understanding-xmlhttprequest)
3. [Methods of XMLHttpRequest](https://www.javatpoint.com/understanding-xmlhttprequest)

An object of XMLHttpRequest is used for asynchronous communication between client and server.

It performs following operations:

1. Sends data from the client in the background
2. Receives the data from the server
3. Updates the webpage without reloading it.

Properties of XMLHttpRequest object

The common properties of XMLHttpRequest object are as follows:

|  |  |
| --- | --- |
| **Property** | **Description** |
| onReadyStateChange | It is called whenever readystate attribute changes. It must not be used with synchronous requests. |
| readyState | represents the state of the request. It ranges from 0 to 4.  **0** UNOPENED open() is not called.  **1** OPENED open is called but send() is not called.  **2** HEADERS\_RECEIVED send() is called, and headers and status are available.  **3** LOADING Downloading data; responseText holds the data.  **4** DONE The operation is completed fully. |
| reponseText | returns response as text. |
| responseXML | returns response as XML |

Methods of XMLHttpRequest object

The important methods of XMLHttpRequest object are as follows:

|  |  |
| --- | --- |
| **Method** | **Description** |
| void open(method, URL) | opens the request specifying get or post method and url. |
| void open(method, URL, async) | same as above but specifies asynchronous or not. |
| void open(method, URL, async, username, password) | same as above but specifies username and password. |
| void send() | sends get request. |
| void send(string) | send post request. |
| setRequestHeader(header,value) | it adds request headers. |

How AJAX works?

AJAX communicates with the server using XMLHttpRequest object. Let's try to understand the flow of ajax or how ajax works by the image displayed below.

As you can see in the above example, XMLHttpRequest object plays a important role.

1. User sends a request from the UI and a javascript call goes to XMLHttpRequest object.
2. HTTP Request is sent to the server by XMLHttpRequest object.
3. Server interacts with the database using JSP, PHP, Servlet, ASP.net etc.
4. Data is retrieved.
5. Server sends XML data or JSON data to the XMLHttpRequest callback function.
6. HTML and CSS data is displayed on the browser.

# Ajax Java Example

To create [ajax](https://www.javatpoint.com/ajax-tutorial) example, you need to use any server-side language e.g. [Servlet](https://www.javatpoint.com/servlet-tutorial), [JSP](https://www.javatpoint.com/jsp-tutorial), [PHP](https://www.javatpoint.com/php-tutorial), [ASP.Net](https://www.javatpoint.com/asp-net-tutorial) etc. Here we are using JSP for generating the server-side code.

In this example, we are simply printing the table of the given number.

#### Steps to create ajax example with jsp

You need to follow following steps:

1. load the org.json.jar file
2. create input page to receive any text or number
3. create server side page to process the request
4. provide entry in web.xml file

#### Load the org.json.jar file

download this example, we have included the org.json.jar file inside the WEB-INF/lib directory.

#### create input page to receive any text or number

In this page, we have created a form that gets input from the user. When user clicks on the showTable button, **sendInfo()** function is called. We have written all the ajax code inside this function.

We have called the **getInfo()** function whenever ready state changes. It writes the returned data in the web page dynamically by the help of **innerHTML** property.

**table1.html**

1. **<html>**
2. **<head>**
3. **<script>**
4. var request;
5. function sendInfo()
6. {
7. var v=document.vinform.t1.value;
8. var url="index.jsp?val="+v;
10. if(window.XMLHttpRequest){
11. request=new XMLHttpRequest();
12. }
13. else if(window.ActiveXObject){
14. request=new ActiveXObject("Microsoft.XMLHTTP");
15. }
17. try
18. {
19. request.onreadystatechange=getInfo;
20. request.open("GET",url,true);
21. request.send();
22. }
23. catch(e)
24. {
25. alert("Unable to connect to server");
26. }
27. }
29. function getInfo(){
30. if(request.readyState==4){
31. var val=request.responseText;
32. document.getElementById('amit').innerHTML=val;
33. }
34. }
36. **</script>**
37. **</head>**
38. **<body>**
39. **<marquee><h1>**This is an example of ajax**</h1></marquee>**
40. **<form** name="vinform"**>**
41. **<input** type="text" name="t1"**>**
42. **<input** type="button" value="ShowTable" onClick="sendInfo()"**>**
43. **</form>**
45. **<span** id="amit"**>** **</span>**
47. **</body>**
48. **</html>**

#### create server side page to process the request

In this jsp page, we printing the table of given number.

**index.jsp**

1. **<**%
2. int n=Integer.parseInt(request.getParameter("val"));
4. for(int i=1;i**<**=10;i++)
5. out.print(i\*n+"**<br>**");
7. %**>**

#### web.xml

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5. http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"**>**
7. **<session-config>**
8. **<session-timeout>**
9. 30
10. **</session-timeout>**
11. **</session-config>**
12. **<welcome-file-list>**
13. **<welcome-file>**table1.html**</welcome-file>**
14. **</welcome-file-list>**
15. **</web-app>**

# Ajax Java Example with Database

In this example, we are interacting with the database. You don't have to make any extra effort. Only write the database logic in you server side page.

In this example, we have written the server side code inside the index.jsp file.

#### Steps to create ajax example with database through jsp

You need to follow following steps:

1. load the org.json.jar file
2. create input page to receive any text or number
3. create server side page to process the request

#### Load the org.json.jar file

download this example, we have included the org.json.jar file inside the WEB-INF/lib directory.

#### create input page to receive any text or number

In this page, we have created a form that gets input from the user. When user press any key **sendInfo()** function is called. We have written all the [ajax](https://www.javatpoint.com/ajax-tutorial) code inside this function.

We have called the **getInfo()** function whenever ready state changes. It writes the returned data in the web page dynamically by the help of **[innerHTML](https://www.javatpoint.com/javascript-innerHTML)** property.

**table1.html**

1. **<html>**
2. **<head>**
3. **<script>**
4. var request;
5. function sendInfo()
6. {
7. var v=document.vinform.t1.value;
8. var url="index.jsp?val="+v;
10. if(window.XMLHttpRequest){
11. request=new XMLHttpRequest();
12. }
13. else if(window.ActiveXObject){
14. request=new ActiveXObject("Microsoft.XMLHTTP");
15. }
17. try{
18. request.onreadystatechange=getInfo;
19. request.open("GET",url,true);
20. request.send();
21. }catch(e){alert("Unable to connect to server");}
22. }
24. function getInfo(){
25. if(request.readyState==4){
26. var val=request.responseText;
27. document.getElementById('amit').innerHTML=val;
28. }
29. }
31. **</script>**
32. **</head>**
33. **<body>**
34. **<marquee><h1>**This is an example of ajax**</h1></marquee>**
35. **<form** name="vinform"**>**
36. Enter id:**<input** type="text" name="t1" onkeyup="sendInfo()"**>**
37. **</form>**
39. **<span** id="amit"**>** **</span>**
41. **</body>**
42. **</html>**

#### create server side page to process the request

In this jsp page, we printing the id and name of the employee for the given id.

**index.jsp**

1. **<**%@ page import="java.sql.\*"%**>**
3. **<**%
4. String s=request.getParameter("val");
5. if(s==null || s.trim().equals("")){
6. out.print("Please enter id");
7. }else{
8. int id=Integer.parseInt(s);
9. out.print(id);
10. try{
11. Class.forName("com.mysql.jdbc.Driver");
12. Connection con=DriverManager.getConnection("jdbc:mysql://localhost:3306/mdb","root","root");
13. PreparedStatement ps=con.prepareStatement("select \* from emp where id=?");
14. ps.setInt(1,id);
15. ResultSet rs=ps.executeQuery();
16. while(rs.next()){
17. out.print(rs.getInt(1)+" "+rs.getString(2));
18. }
19. con.close();
20. }catch(Exception e){e.printStackTrace();}
21. }
22. %**>**

# Java AJAX Email Finder Example

We can create an AJAX example in Java which checks whether the given email id exists in the database or not.

#### Steps to create email finder example using AJAX in Java

You need to follow following steps:

1. Create table in database
2. load the org.json.jar file
3. Create input form
4. Create server side page to search employee using name

#### Create table in database

In this example, we are using oracle 10g database. Here, we have created a table "user100" which has following data.

#### Load the org.json.jar file

download this example, we have included the org.json.jar file inside the WEB-INF/lib directory.

#### Create input form

In this page, we have created a form that gets input from the user to find email. When you click on the check availability button, it tells whether email id is available or not.

**index.html**

1. <!DOCTYPE html**>**
2. **<html>**
3. **<head>**
4. **<title>**Email Finder Example**</title>**
5. **<script>**
6. var request;
7. function sendInfo(){
8. var email=document.vinform.email.value;
9. var url="emailfinder.jsp?email="+email;
11. if(window.XMLHttpRequest){
12. request=new XMLHttpRequest();
13. }
14. else if(window.ActiveXObject){
15. request=new ActiveXObject("Microsoft.XMLHTTP");
16. }
17. try{
18. request.onreadystatechange=getInfo;
19. request.open("GET",url,true);
20. request.send();
21. }catch(e){alert("Unable to connect to server");}
22. }
24. function getInfo(){
25. if(request.readyState==4){
26. var val=request.responseText;
27. document.getElementById('mylocation').innerHTML=val;
28. }
29. }
31. **</script>**
32. **</head>**
33. **<body>**
34. **<marquee><h1>**AJAX Email Checker Example**</h1></marquee>**
35. **<form** name="vinform"**>**
36. **<input** type="email" name="email" placeholder="enter email"**/>**
37. **<input** type="button" onclick="sendInfo()" value="Check Availability"**/>**
38. **<span** id="mylocation"**></span>**
39. **</form>**
41. **</body></html>**

#### Create server side page to process the request

In this jsp page, we are writing the database code to search email id.

**emailfinder.jsp**

1. **<**%@ page import="java.sql.\*" %**>**
2. **<**%
3. String email=request.getParameter("email");
4. if(email.contains("@")&&email.contains(".")){
5. try{
6. Class.forName("oracle.jdbc.driver.OracleDriver");
7. Connection con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe","system","oracle");
8. PreparedStatement ps=con.prepareStatement("select \* from user100 where email=?");
9. ps.setString(1,email);
10. ResultSet rs=ps.executeQuery();
11. if(rs.next()){
12. out.print("Unavailable! **<img** src='unchecked.gif'**/>**");
13. }else{
14. out.print("Available! **<img** src='checked.gif'**/>**");
15. }
16. }catch(Exception e){out.print(e);}
17. }else{
18. out.print("Invalid email!");
19. }
20. %**>**

# Comment Form Example using AJAX in Java