

# Ajax- Search Products

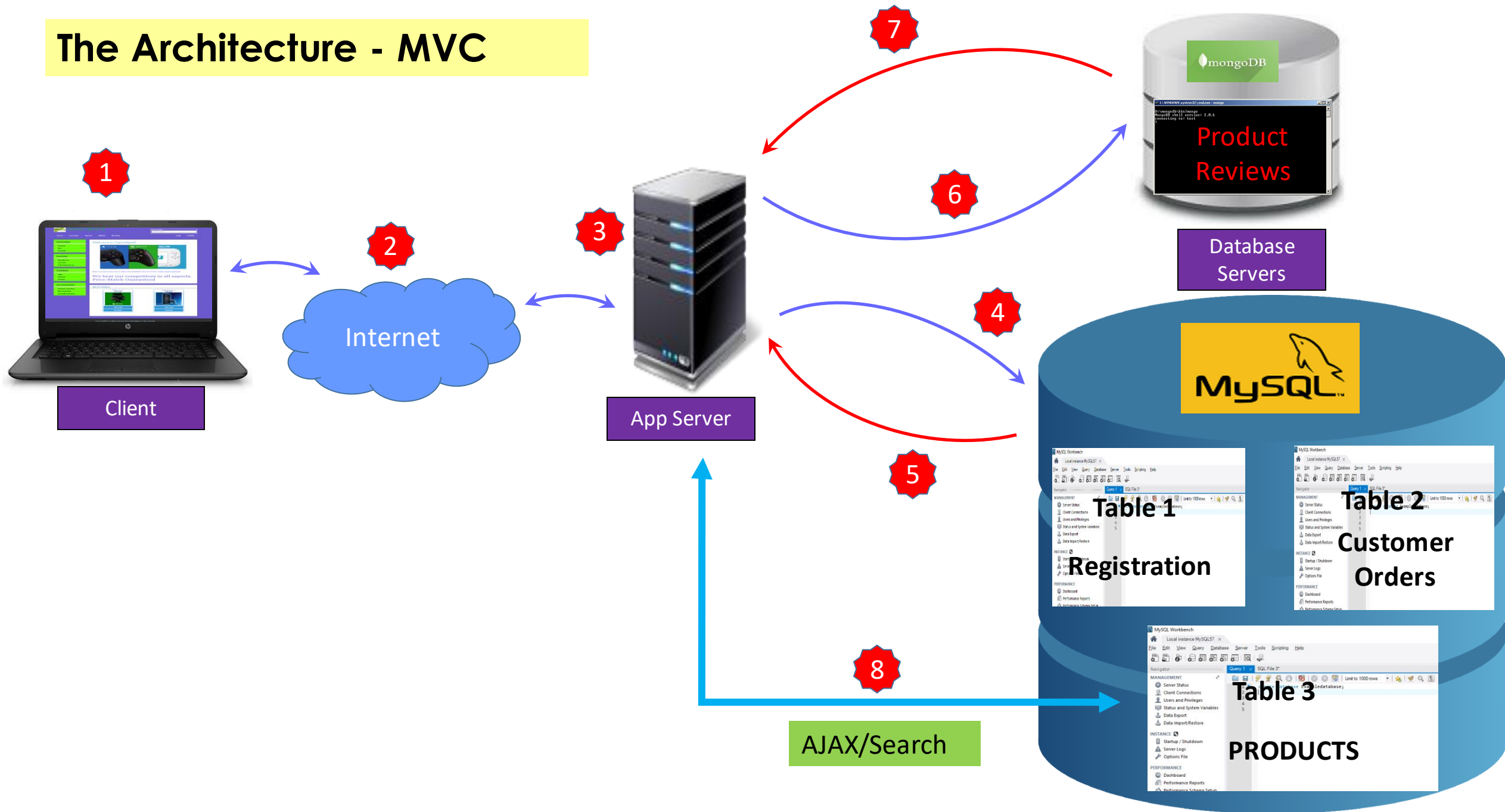
Tutorial – 5, Ajax Auto Complete Search feature

CSP 584 – Enterprise Web Application

Dr. Atef Bader

Illinois Institute of Technology

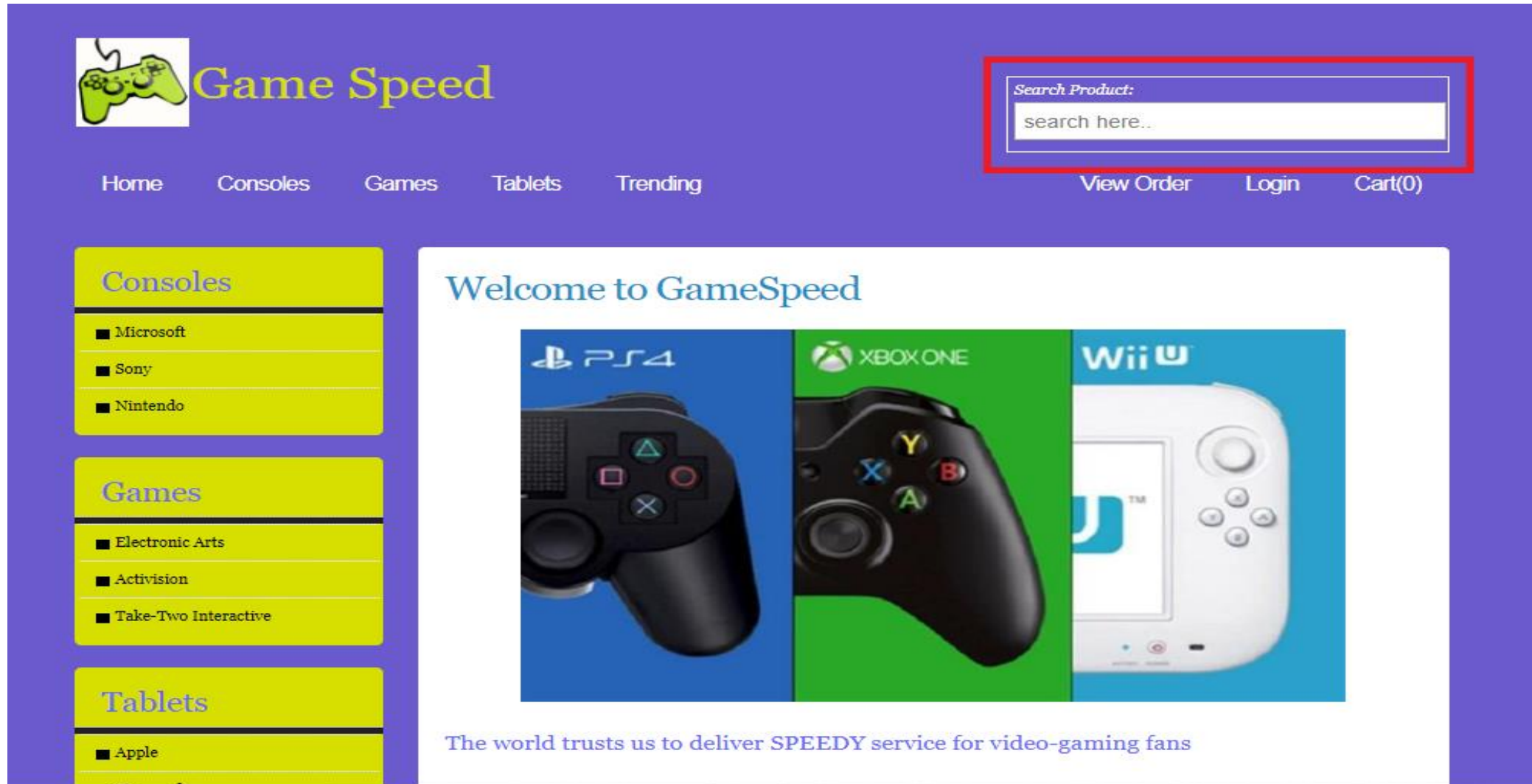
# The Architecture - MVC



# Introduction:

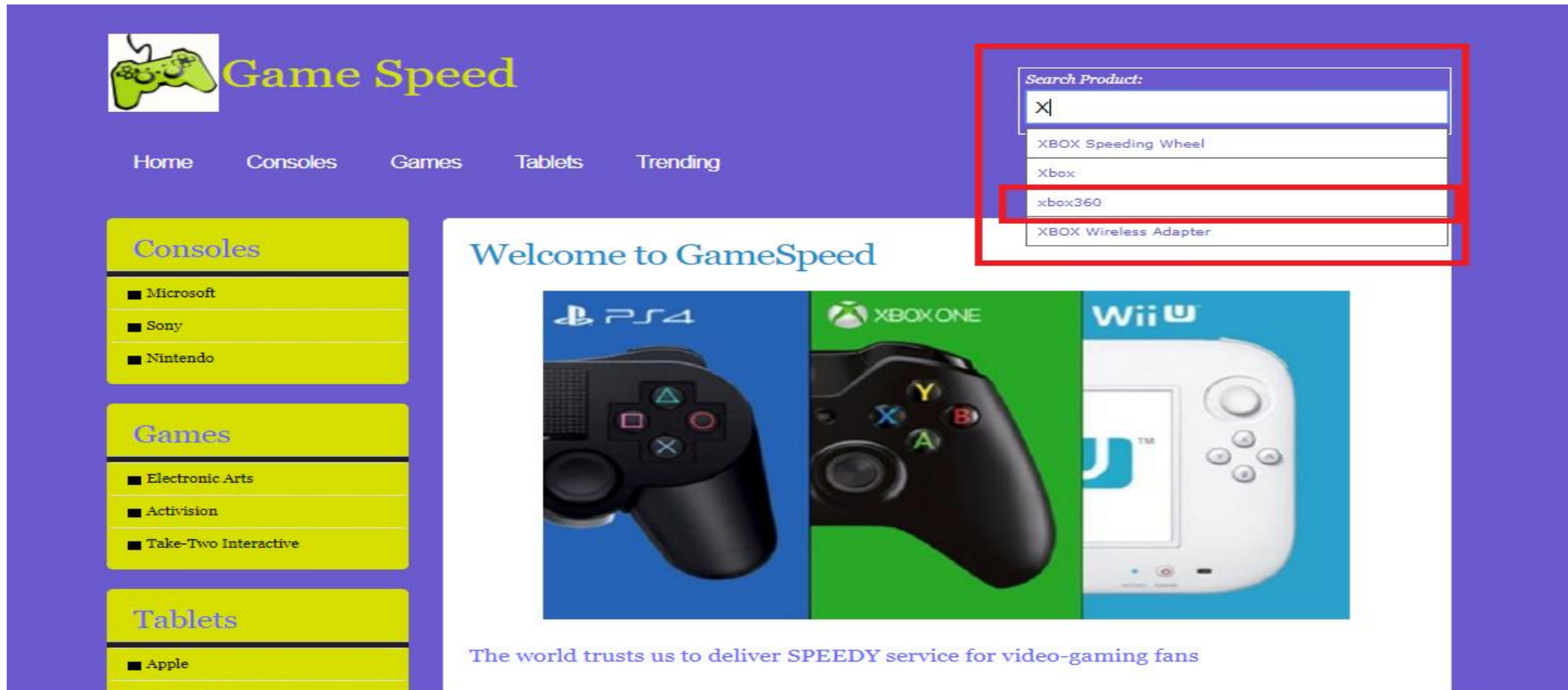
- ❖ **JavaScript** is the programming language of HTML and the Web.
- ❖ JavaScript is one of the 3 languages all web developers must learn:
  1. HTML to define the content of web pages
  2. CSS to specify the layout of web pages
  3. **JavaScript to program the behavior of web pages**
- ❖ To Learn More on JavaScript, visit: <http://www.w3schools.com/js/default.asp>
  
- **AJAX : Asynchronous JavaScript And XML.**
- AJAX allows **web pages to be updated asynchronously by exchanging data with a web server behind the scenes.**
- Advantages: Updates 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, Sends data to a server - in the background
- To Learn More on AJAX, visit: [http://www.w3schools.com/xml/ajax\\_intro.asp](http://www.w3schools.com/xml/ajax_intro.asp)

# Sample Example For Search Box



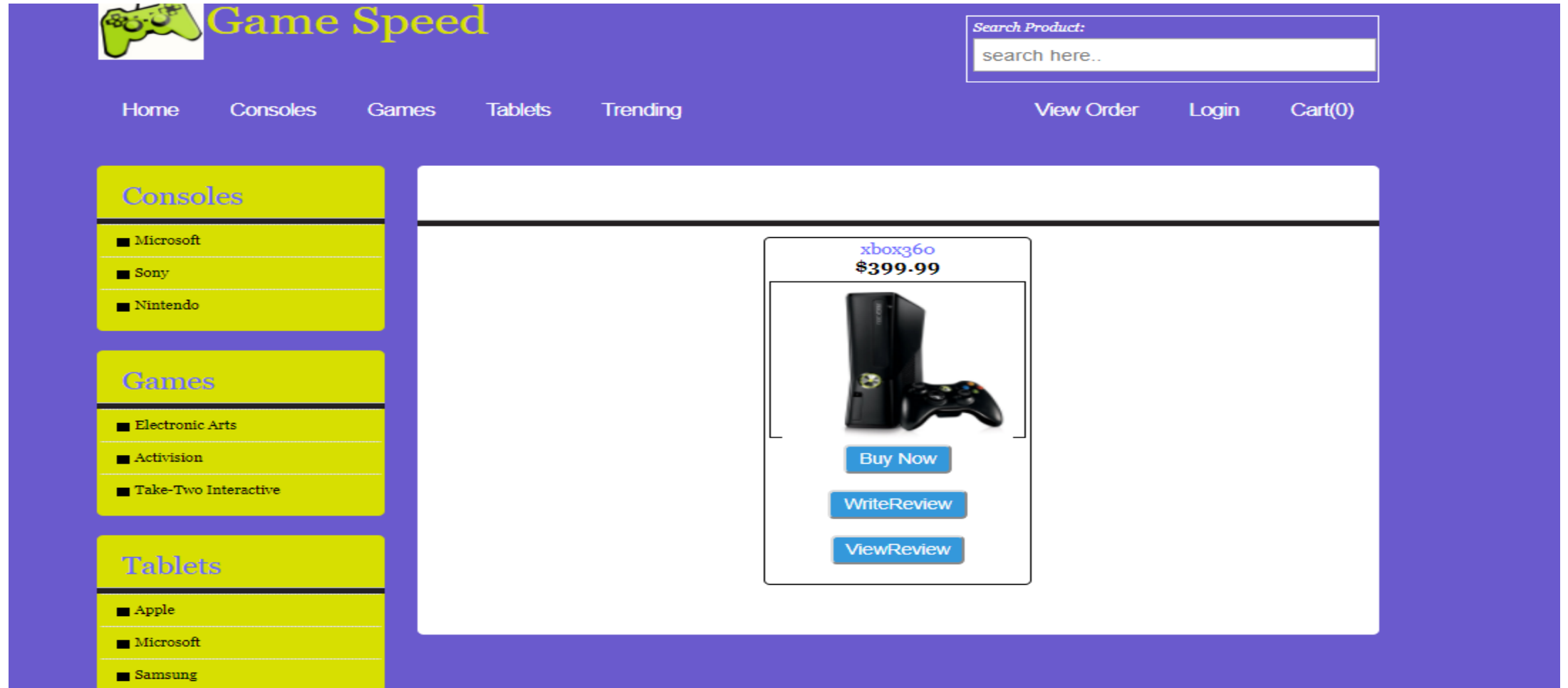
# Search Product Functionality

- After entering letter in the search box all the products starting with that letter should be displayed.
- In the Below Example , Customer enters the letter 'X' and the Suggestions from product are shown , From the suggestions list user clicks the Xbox360.



# View Product

On clicking a particular product displayed in search box, Page should be redirected to display that product detail



# Products Database

- All the products details are stored in product table in MySQL.
- Products Stored can be checked by executing select Query in database using MySQL workbench.

The screenshot displays the MySQL Workbench interface. The 'SQL File 1\*' tab is active, showing a query that selects all data from the 'productdetails' table. The 'Result Grid' is visible below the query, displaying a list of products. A red rectangle highlights the first three rows of the result set.

SQL File 1\* x

```
1 use exempladb;
2
3 select * from productdetails;
4
```

Limit to 1000 rows

Result Grid

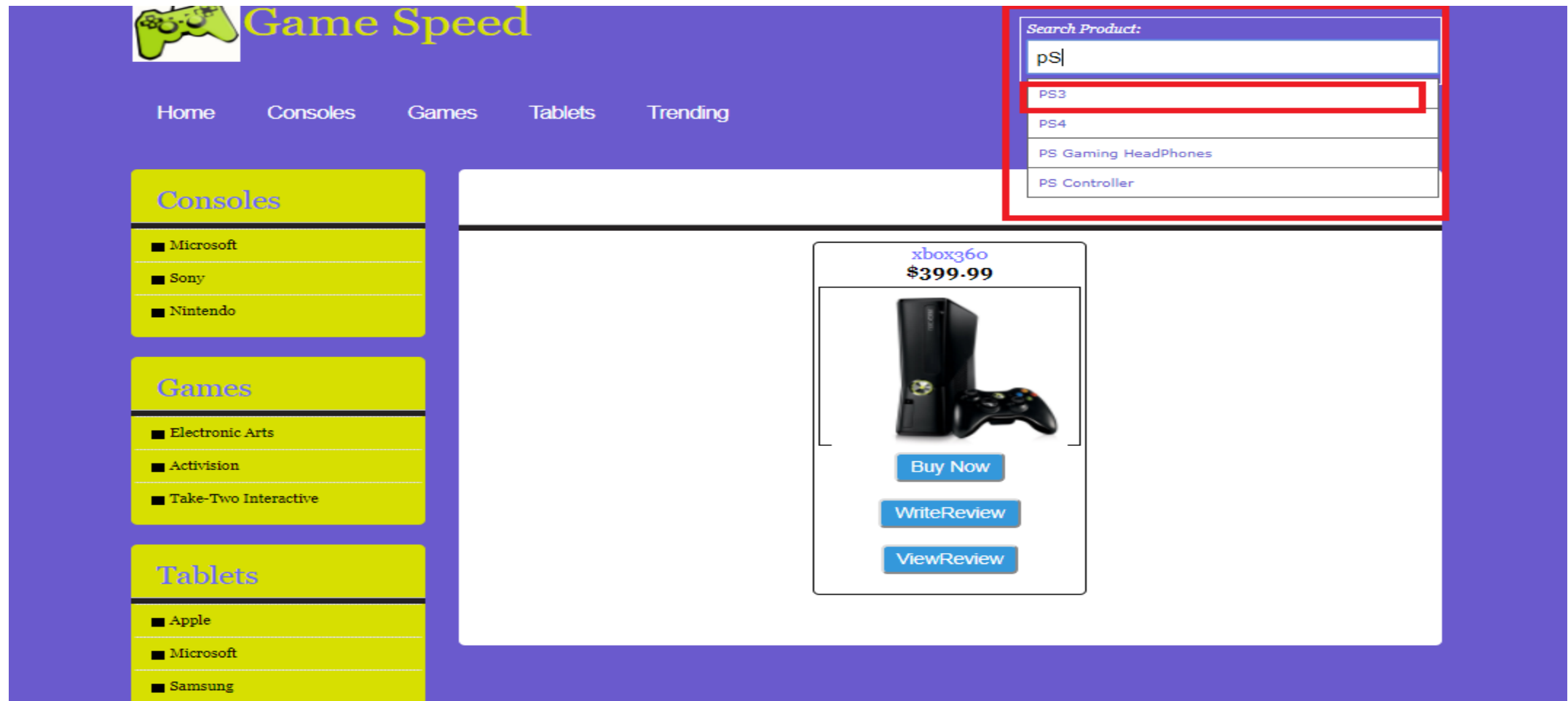
	ProductType	Id	productName	productPrice	productImage	productManufacture
	games	a4	Evolve	50.99	tti_evolve.jpg	TakeTwoInteractive
	tablets	ipad	Apple Ipad Pro	189.99	apple1.jpg	Apple
	tablets	ipad3	Apple Ipad 3	189.99	apple2.jpg	Apple
	accessories	nintendo sh	Nintendo Racing Controller	89.99	Wii2.jpg	Nintendo
	accessories	nintendo wc	Nintendo Controller	89.99	Wii1.jpg	Nintendo
	consoles	PS3	PS3	349.99	ps3.jpg	Sony
	accessories	ps3 wc	PS Gaming Headphones	89.99	Turtle Beach Headset.jpg	Sony
	consoles	PS4	PS4	349.99	PS4-console-bundle.jpg	Sony
	accessories	ps4 wc	PS4 Controller	89.99	XBOX_controller.jpg	Sony
	tablets	Samsung2	Samsung Tab 2	150.99	s1.jpg	Samsung
	tablets	Surface	Surface	299.99	m1.jpg	Microsoft
	tablets	Surface4	Surface Pro 4	299.99	m2.jpg	Microsoft
	consoles	WiiU	WiiU	240.00	WiiU_Console.jpg	Nintendo
	consoles	Xbox	Xbox	399.99	xbox1.jpg	Microsoft
	consoles	xbox360	xbox360	399.99	xbox360.jpg	Microsoft
	accessories	xbox360 ka	charging	89.99	charging.jpg	Microsoft
	accessories	xbox360 mp	Steering	89.99	steering.jpg	Microsoft

productdetails 1 x

Apply Revert Context

# One more search entry for search box

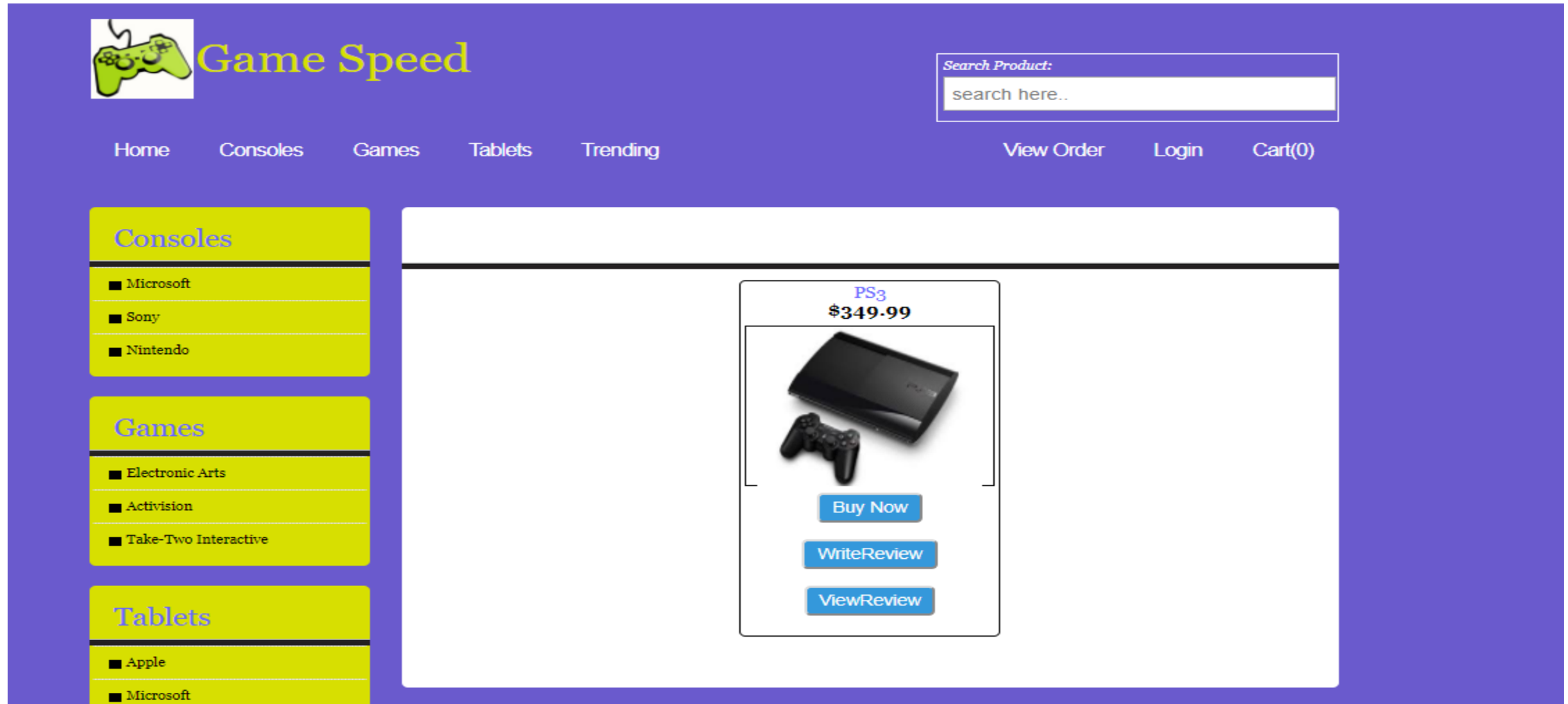
- After entering letter in the search box all the products starting with that letter will be displayed.
- In the Below Example , Customer enters the letter 'p' and the Suggestions from products are shown , From the suggestion list customer clicks the PlayStation 3.





# View Product

On clicking a particular product displayed in search box, Page should be redirected to display that product detail



# Adding Search Box in html

```
<body onload="init()">
  <script type="text/javascript" src="javascript.js"></script>
  <div name="autofillform">
    <input type="text" name="searchId" value="" class="input" id=" searchId"
      onkeyup="doCompletion()"
      placeholder="search here.." style="padding: 5px; font-size: 16px;" />
    <div id="auto-row">
      <table id="complete-table" class="gridtable" style="position: absolute; width:
        315px;"></table>
    </div>
  </div>
</body>
```

Include the  
javascript file in  
html

Call the Javascript  
function to process  
the search text

# JavaScript for Search

```
function init() {  
    completeField = document.getElementById("searchId ");  
    completeTable = document.getElementById("complete-table");  
    autoRow = document.getElementById("auto-row");  
}  
function doCompletion() {  
    var url = "autocomplete?action=complete&searchId =" + escape(searchId.value);  
    req = initRequest();  
    req.open("GET", url, true);  
    req.onreadystatechange = callback;  
}  
function initRequest() {  
    if (window.XMLHttpRequest) {  
        if (navigator.userAgent.indexOf('MSIE') != -1) {  
            isIE = true;  
        }  
        return new XMLHttpRequest();  
    } else if (window.ActiveXObject) {  
        isIE = true;  
        return new ActiveXObject("Microsoft.XMLHTTP");  
    }  
}
```

init() function  
Getting fields from  
html in javascript

We are sending searchId as  
parameter that is the text we  
entered in search box

Direct the webpage to  
autocomplete servlet

Creat Javascript Activex  
Object

```
function appendProduct(productName,productId){
```

```
    var row;  
    var cell;  
    var linkElement;
```

```
    if (isIE) {
```

```
        completeTable.style.display = 'block';
```

```
        row = completeTable.insertRow(completeTable.rows.length);
```

```
        cell = row.insertCell(0);
```

```
    } else {
```

```
        completeTable.style.display = 'table';
```

```
        row = document.createElement("tr");
```

```
        cell = document.createElement("td");
```

```
        row.appendChild(cell);
```

```
        completeTable.appendChild(row);
```

```
    }
```

```
    cell.className = "popupCell";
```

```
    linkElement = document.createElement("a");
```

```
    linkElement.className = "popupItem";
```

```
    linkElement.setAttribute("href", "autocomplete?action=lookup&searchId="+ productId);
```

```
    linkElement.appendChild(document.createTextNode(productName));
```

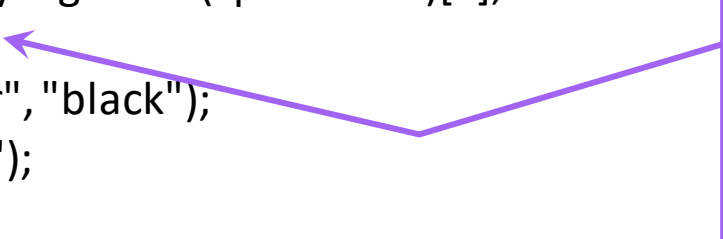
```
    cell.appendChild(linkElement);
```

```
}
```

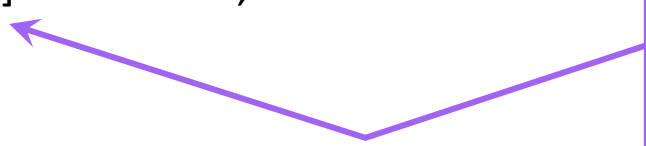
This function appends the products starting with searchId to web page

Append all products in table row and display them below the search box

```
function parseMessages(responseXML) {  
    // no matches returned  
    if (responseXML == null) {  
        return false;  
    } else {  
        var products = responseXML.getElementsByTagName("products")[0];  
        if (products.childNodes.length > 0) {  
            completeTable.setAttribute("bordercolor", "black");  
            completeTable.setAttribute("border", "1");  
  
            for (loop = 0; loop < products.childNodes.length; loop++) {  
                var product = products.childNodes[loop];  
                var productName = product.getElementsByTagName("productName")[0];  
                var productId = product.getElementsByTagName("id")[0];  
                appendProduct(productName.childNodes[0].nodeValue,  
                    productId.childNodes[0].nodeValue);  
            }  
        }  
    }  
}
```




Get the products obtained  
as response from auto  
complete servlet




Pass the product id and  
product name to  
appendProduct()  
function

```
function callback() {  
    clearTable();  
    if (req.readyState == 4) {  
        if (req.status == 200) {  
            parseMessages(req.responseXML);  
        }  
    }  
}
```



Parse the values given by  
auto complete servlet on  
response callback

```
function clearTable() {  
    if (completeTable.getElementsByTagName("tr").length > 0) {  
        completeTable.style.display = 'none';  
        for (loop = completeTable.childNodes.length - 1; loop >= 0 ; loop--) {  
            completeTable.removeChild(completeTable.childNodes[loop]);  
        }  
    }  
}
```



Clears the table below  
search box and removes  
all products from it

# Auto Complete Servlet Code

```
try
{
    StringBuffer sb = new StringBuffer();
    boolean namesAdded = false;
    if (action.equals("complete"))
    {
        if (!searchId.equals(""))
        {
            AjaxUtility a=new AjaxUtility();
            sb=a.readdata(searchId);
            if(sb!=null || !sb.equals(""))
            {
                namesAdded=true;
            }
            if (namesAdded)
            {
                response.setContentType("text/xml");
                response.getWriter().write("<products>" + sb.toString() + "</products >");
            }
        }
    }
}
```

Calling AjaxUtility class  
readData() Function to  
get products starting  
with searchId

Sending the string  
buffer as response  
in xml format

# Ajax Utility Function- getData()

getData() function used to get the products from database and store in hashmap

```
public static HashMap<String,Product> getData()
```

```
{    HashMap<String,Product> hm=new HashMap<String,Product>();
```

```
    try
```

```
    {        getConnection();
```

```
        Statement stmt=conn.createStatement();
```

```
        String selectCustomerQuery="select * from product";
```

```
        ResultSet rs = stmt.executeQuery(selectCustomerQuery);
```

```
        while(rs.next())
```

```
        {
```

```
            Product p = new Product(rs.getString("productId"), rs.getString("productName"));
```

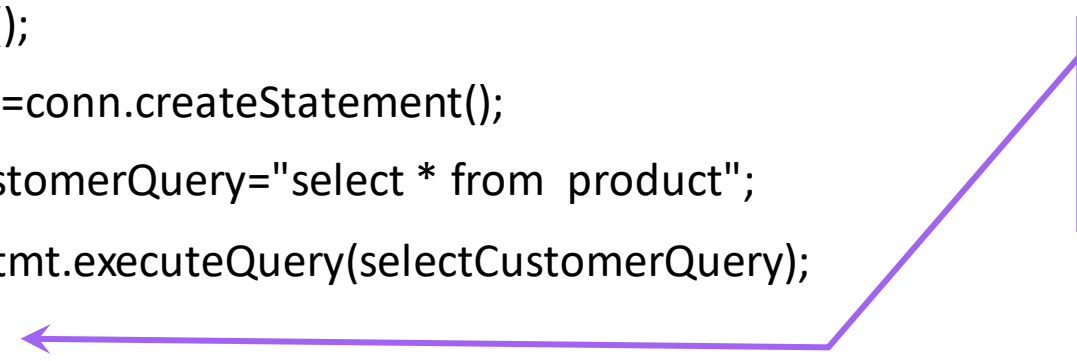
```
            hm.put(rs.getString("productId"), p);
```

```
        }
```

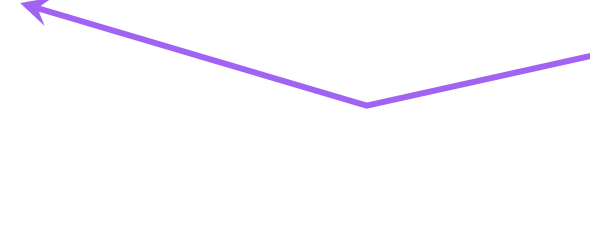
```
    }
```

```
    return hm;
```

```
}
```



Iterate through result set to get each product record



Get the data from table and store in hashmap



# Ajax Utility Function- readData()

readData() function used to get the products starting with letter typed from hashmap into string buffer

```
public StringBuffer readdata(String searchId)
{
    HashMap<String,Product> data;
        data=getData();
    Iterator it = data.entrySet().iterator();
    while (it.hasNext())
    {
        Map.Entry pi = (Map.Entry)it.next();
        Product p=(Product)pi.getValue();
        if (p.getName().toLowerCase().startsWith(searchId))
        {
            sb.append("<product>");
            sb.append("<id>" + p.getId() + "</id>");
            sb.append("<productName>" + p.getName() + "</ productName>");
            sb.append("</ product >");
        }
    }
    return sb;
}
```

Calling get data function to get the product data into hashmap

Check if hashmap contains any product starting with letter stored in searchId

Append the product details in xml tag format

Questions ??