

AJAX for Java Web Applications



Session: 2

JSON and DWR

Objectives

- ◆ Explain JSON
- ◆ Describe the JSON data types and arrays
- ◆ Explain how to use JSON with AJAX
- ◆ Explain DWR and steps to use DWR
- ◆ Explain Reverse Ajax

For Aptech Centre Use Only

Introduction

- ◆ AJAX is a group of interrelated technologies.
- ◆ Thus, JSON, DWR, and many other frameworks can be used with AJAX to simplify the work of a developer.
- ◆ This also helps to improve the performance and efficiency of an application.

JSON 1-2

JavaScript Object Notation (JSON), a subset of JavaScript language, is a text-based data format that supports data exchange between a Web browser and a server.

It is available as libraries in many languages, such as C#, Java, Python, and so on.

JSON 3 is a modern JSON implementation compatible with a variety of JavaScript platforms, such as IE 6, Safari 2, Opera 7, and Netscape 6. The current version is 3.3.2.

It is used for transforming Java arrays, beans, collections, maps, and XML to JSON.

The native data types such as Date, Error, Regular Expression, and Function were not supported in the previous versions of JSON lib. However, in this version, converting from Date to String is possible.

- ◆ JSON is preferred over XML due to the following reasons:
 - ◆ JSON is lighter and faster as compared to XML.
 - ◆ JSON objects are typed whereas XML objects are untyped.
 - ◆ JSON supports data types such as string, number, array and boolean, whereas XML supports only the string data type.
 - ◆ JSON code is native to JavaScript code. It is readily accessible to JavaScript code. However, XML code needs to be parsed and assigned to variables using tedious DOM operations.

JSON Data Types 1-2

- JSON's text-based data format is based on JavaScript's object notation. The elements of the object notation supported by JSON are listed in the following table:

Type	Description
Object	An object is a collection of unordered name/value pairs. A JSON object is represented by {}.
Object Member	A JSON object member consists of a name/value pair, which is a combination of string and value. Members are separated by using commas. Object name and its value in a name/value pair are separated by a colon.
Array	A JSON array consists of elements or values that are separated by commas. Array indexes are zero(0)-based.
Value	A JSON value can be a string, a number, a boolean, an object, null, or an array.
String	A JSON string consists of Unicode characters except double quotes ('), backslash (\), or control characters. A JSON string is enclosed within double quotes. For example, {"myString": "abcdef"}
Number	Double-precision floating-point format in JavaScript. For example, {"myNum": 123.456}
Boolean	True or False. For example, {"myBool": true}
Whitespace	Can be used between any pair of tokens
null	Variable with null (empty) value. For example, {"myNull": null}

JSON Data Types 2-2

- ◆ Following Code Snippet presents the personal details of Jack Daniels in JSON notation, using JSON elements:

```
{
  "fullname": "Jack Daniels",
  "company": "JSON Consulting",
  "age" : 20,
  "email": [
    {"type": "work", "value": "jack.daniels@jsonconsult.com"},
    {"type": "home", "value": "jack@hotmail.com"}
  ],
  "contactno": [
    {"type": "work", "value": "123456"},
    {"type": "fax", "value": "34567"},
    {"type": "mobile", "value": "9987651111"}
  ],
  "addresses": [
    {"type": "work", "format": "us",
     "value": "1234 Manhutton"},
    {"type": "home", "format": "us",
     "value": "5678 Springfield"}
  ]
}
```

JSON Objects

- ◆ JSON objects can be used in JavaScript without any additional effort because JSON is a subset of JavaScript.
- ◆ A JavaScript variable can be declared and assigned a JSON-formatted data structure to the variable.
- ◆ A particular element of a JSON-formatted object can be accessed and modified by using the 'dot' notation from JavaScript code.
- ◆ The different types of JSON objects that can be accessed using 'dot' notations are as follows:

JSON object elements

- JSON object elements can be accessed and modified using 'dot' notation. For example, Student.name

Array elements

- An array element can be accessed and modified by using an index.
- Attribute of an array element can be accessed as array_name[index].attribute.

Nested JSON objects

- A nested object can be accessed and modified by using 'dot' notation as first_object.second_object.attribute.

- ◆ Following Code Snippet demonstrates use of 'dot' notation to access different types of JSON objects:

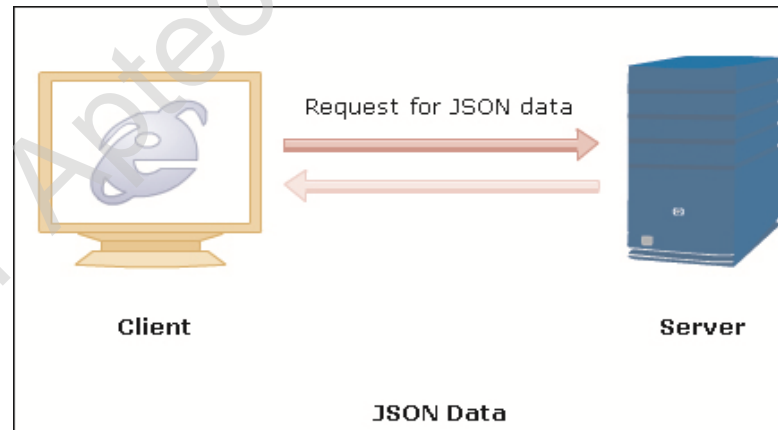
```
email[0].value;  
email[1].value;
```


Receiving JSON Data from the Server

- ◆ In an AJAX application, both client and server can receive and process JSON text.
- ◆ When a client requests a server for some data, the server can send the requested data in JSON format, XML format, or as plain text.
- ◆ If the server returns the data in JSON format, the client receives the data in string format.
- ◆ The client uses the following steps to process the data:

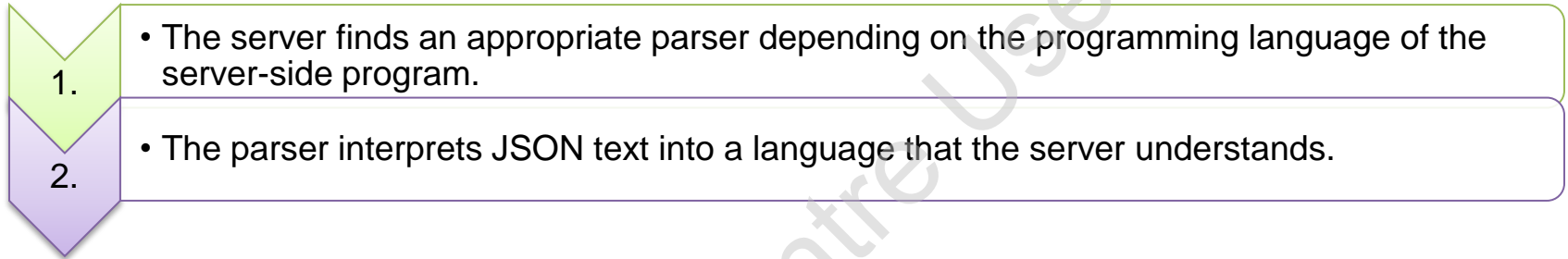
1. The client converts the string data into a JavaScript object by using the statement, `eval('(' + request.responseText + ')')`.
2. The client can access and modify properties of the converted JavaScript object.

- ◆ Following figure shows the client requesting for JSON data:

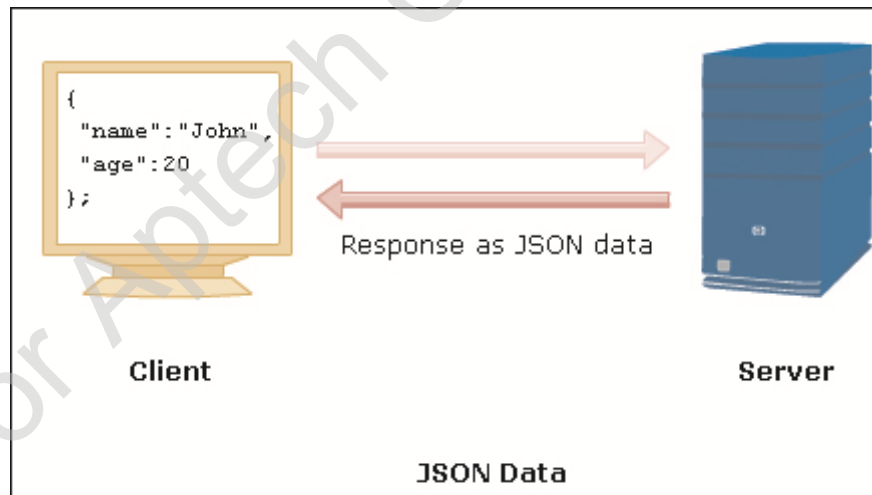


Receiving JSON Data from the Client

- ◆ When a client sends the processed data to the server, the client converts the JSON object to JSON text.
- ◆ The server uses the following steps to process the JSON text received from the client:

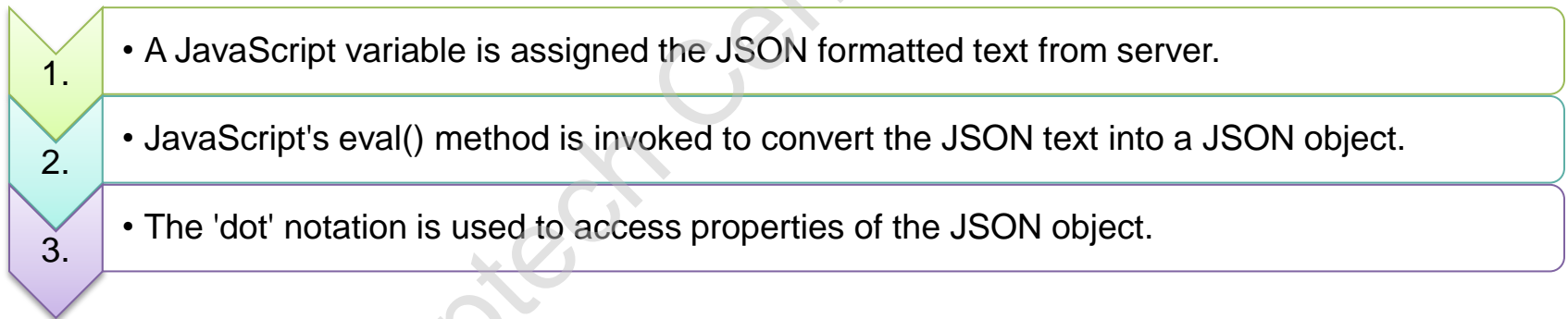


- ◆ Following figure shows receiving of JSON data from server:



'Assignment' Technique

- ◆ The three techniques of receiving JSON formatted text from server are as follows:
 - ◆ **Assignment**
 - ◆ **Callback**
 - ◆ **Parse**
- ◆ The steps followed by the 'Assignment' technique to receive JSON data are as follows:



- ◆ Following Code Snippet demonstrates the 'Assignment' technique to receive JSON data:

```
var JSONres= "week = { 'weekday': 'Monday' }";  
eval (JSONres) ;  
document.writeln (week.weekday) ;
```

'Callback' Technique

- ◆ In the 'Callback technique, a pre-defined function is called and the server response is passed in JSON format as the first argument to the function.
- ◆ The Callback technique is used to receive JSON data from Web sites of external domains.
- ◆ Following Code Snippet demonstrates the 'Callback' technique to receive JSON data:

```
function processData(inputJSON) {  
    document.writeln(inputJSON.weekday); //output Monday  
}  
var week= "processData( { 'weekday' : 'Monday' } )";  
eval(week);
```

- ◆ The code defines a callback function with JSON object as argument that Outputs 'Monday'.
- ◆ Next, it passes JSON object as argument to processData() function and assigns it to JavaScript variable.
- ◆ Finally, it executes the JavaScript code.

'Parse' Technique

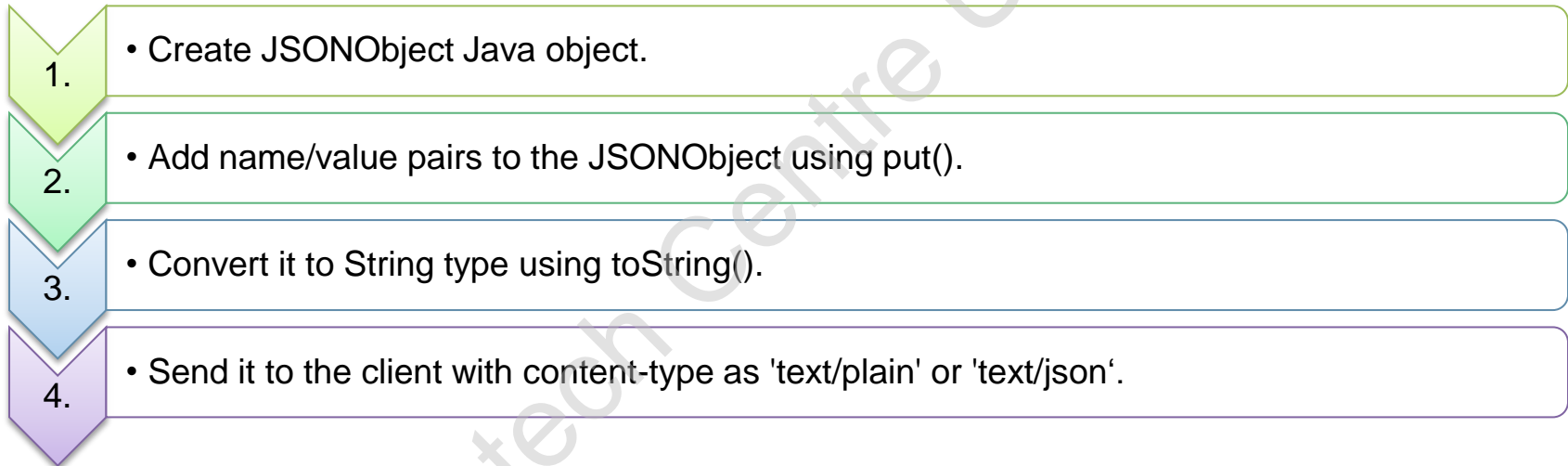
- ◆ The 'Parse' technique parses and executes only JSON text that comes from the server as part of response text.
- ◆ Therefore, this technique of receiving JSON data is the safest.
- ◆ The JSON method `parseJSON()` is used to parse and execute JSON text that is received as a response from the server.
- ◆ Following Code Snippet demonstrates the 'Parse' technique to receive JSON data:

```
JSONresponse = '{"weekday": "Monday"}';  
object=JSONresponse.parseJSON();  
document.writeln(object.weekday);
```

- ◆ The code assigns JSON text to `JSONresponse`.
- ◆ Next, it parses JSON text and displays the result.

Sending JSON Data

- ◆ To send a JSON object as part of a client request, convert the JSON object into a string data by using `toJSONString()` and then, use GET or POST method with `XMLHttpRequest` object.
- ◆ The steps for sending JSON data are as follows:



- ◆ Following Code Snippet demonstrates the steps for sending JSON data:

```
JSONObject objJSON=new JSONObject();  
objJSON.put("weekday","Monday");  
StringEntity entity = new StringEntity(objJSON.toString());  
entity.setContentType("text/json;charset=UTF-8");
```

JSON with AJAX 1-7

- ◆ JSON can be used to pass data asynchronously using AJAX between the client and server.
- ◆ JSON files, when necessary, can be accessed and parsed by JavaScript using AJAX and then, the two tasks to be performed are as follows:

The parsed values are stored in variables for further processing before displaying them on the Web page.

The value gets displayed on the Web page as the data is directly assigned to the DOM elements of JavaScript.

- ◆ Consider a scenario where the Web page is updated asynchronously using JSON and JavaScript.
- ◆ Following are the steps to be followed for using JSON:

1.

- Download the JSON library from the <http://mvnrepository.com/artifact/net.sf.json-lib/json-lib/2.4> or any other freely available site.

2.

- Open Netbeans IDE and create a Web application.

JSON with AJAX 2-7

3.

- The JSON jar file is to be added in the Libraries folder of the project. Right-click the Libraries folder and select Add JAR/Folder and select the JSON jar file from the file dialog box.

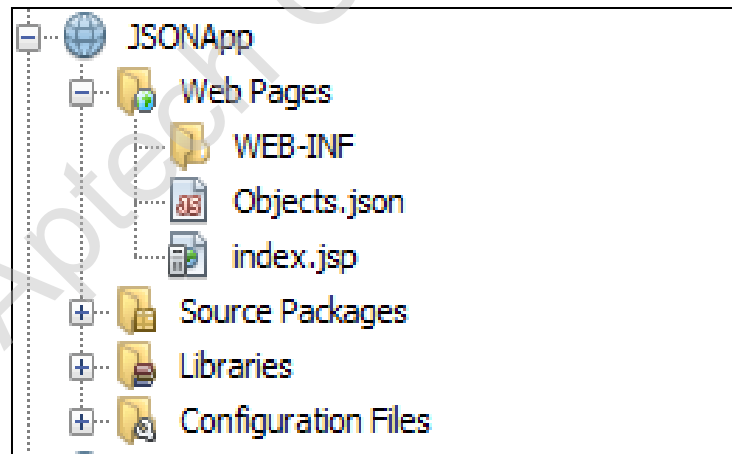
4.

- Right-click the Web Pages folder and then, select New → JSP. Name it as index.jsp.

5.

- Right-click the Web Pages folder and then, select New → JSON file. Name it as Objects.json.

- ◆ The directory structure of the Web application is shown in the following figure:



JSON with AJAX 3-7

- ◆ Following Code Snippet demonstrates the JSON file named Objects.json which consists of nested objects:

```
{
  "Objects": [
    {
      "Students": [
        {
          "StudentID": "1",
          "Name": "Andrews"
        },
        {
          "StudentID": "2",
          "Name": "Ashley"
        },
        {
          "StudentID": "3",
          "Name": "Jeff"
        }
      ]
    }
  ],
}
```



```
{
  "Subjects": [
    {
      "ID": "1",
      "Name": "Networking"
    },
    {
      "ID": "2",
      "Name": "C-Programming"
    },
    {
      "ID": "3",
      "Name": "Java"
    }
  ]
}
```

JSON with AJAX 4-7

- ◆ The index.jsp file accesses the data from the Objects.json file and displays it accordingly.
- ◆ Following Code Snippet demonstrates the index.jsp file:

```
<script type="text/javascript">
function loadJSON(object){
    var data_file
    data_file = "Objects.json";
    var ajax_request = new XMLHttpRequest();
    ...
    ajax_request.onreadystatechange = function(){
        if (ajax_request.readyState == 4 ){

            var jsonObject = JSON.parse(ajax_request.responseText);
            var str="<table border=1 width='40%'>";

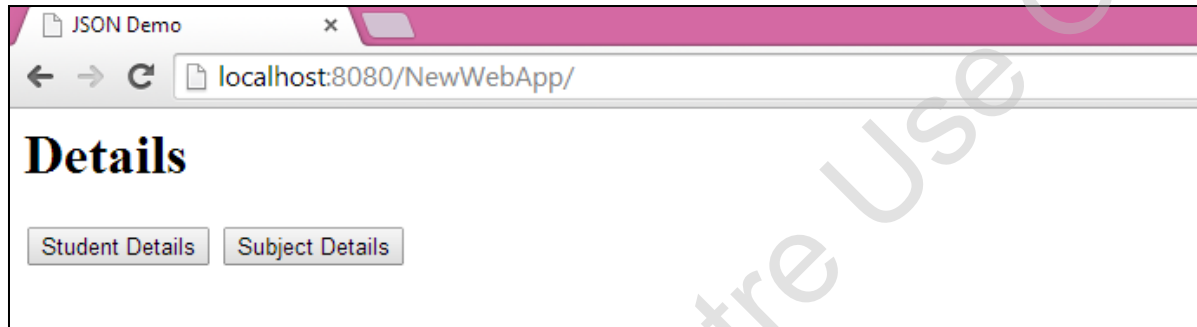
            if(object=="Student"){
                for(var i=0; i< jsonObject.Objects[0].Students.length; i++){
                    str = str + " <tr><td>" +
                    jsonObject.Objects[0].Students[i].StudentID + "</td>";
                    str = str + "<td>" + jsonObject.Objects[0].Students[i].Name +
                    "</td> </tr>";
                }
            }
        }
    }
}
```

JSON with AJAX 5-7

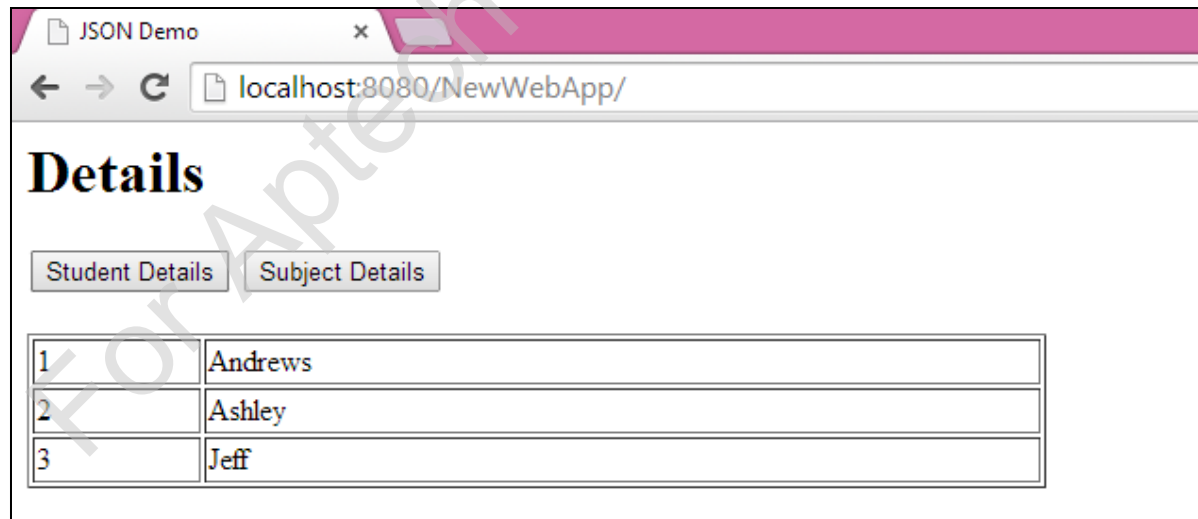
```
else {
    for(var i=0; i< jsonObject.Objects[1].Subjects.length; i++){
        str = str + " <tr><td>" + jsonObject.Objects[1].Subjects[i].ID + "</td>";
        str = str + "<td>" + jsonObject.Objects[1].Subjects[i].Name +
"</td> </tr>";
    }
    str=str + "</table>";
    document.getElementById("container").innerHTML=str;
}
};
ajax_request.open("GET", data_file, true);
ajax_request.send();
}
</script>
...
<body>
<h1>Details </h1>
<button type="button" id="btnStudent"
onclick="loadJSON('Student')">Student Details </button>
<button type="button" id="btnSubject"
onclick="loadJSON('Subject')">Subject Details </button>
<br/> <div id="container">
</div> </body>
...
```

JSON with AJAX 6-7

- ◆ Following figure shows the output of index.jsp Web page:

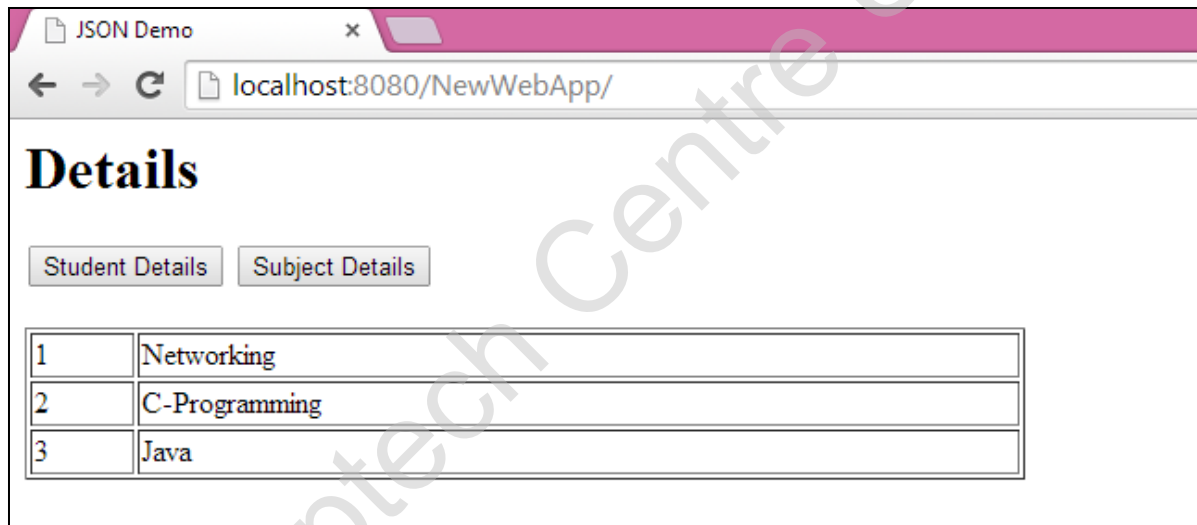


- ◆ On clicking the Student Details button, the loadJSON() method is called in which details about students in the Objects.json file will be displayed in a tabular format as shown in the following figure:



JSON with AJAX 7-7

- ◆ The div container is updated asynchronously with the Student details retrieved from the Objects.json file.
- ◆ Following figure shows the subject details displayed on click of the Subject Details button:

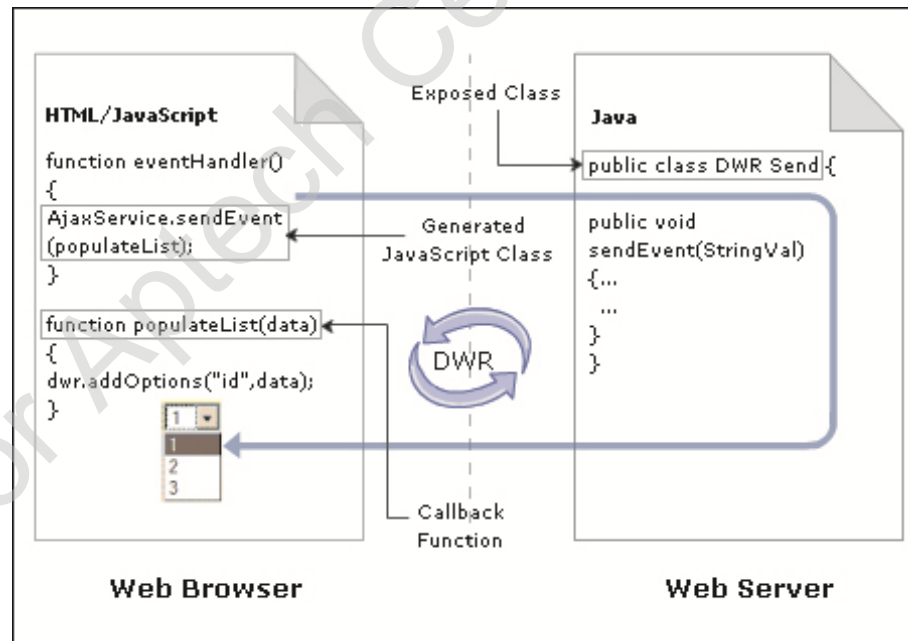


- ◆ Thus, in the output, the whole Web page is not updated but only the div container is getting updated on click of a button.

Direct Web Remoting (DWR)

Direct Web Remoting (DWR) is an AJAX framework that allows clients to remotely access server functionalities. It is based on Java technology. It can generate JavaScript code from Java classes.

- ◆ Client applications cannot access the server-side Java classes directly.
- ◆ To allow the client applications to access the server-side Java classes, DWR converts the Java classes into JavaScript code.
- ◆ The developer can decide which Java classes would be accessible to the client browser by configuring DWR. Following figure shows the DWR framework:



Features of Direct Web Remoting

Hides the complexities of AJAX technology: DWR automatically manages the XMLHttpRequest object, object serialization, and remote invocation of Java classes in the Web server.

Support for marshalling/unmarshalling of objects: DWR library can marshal/unmarshal any JavaScript object that is exchanged between a client and a server application. DWR converts data types of Java to JavaScript and JavaScript to Java.

Integration with popular frameworks: DWR library supports Spring, Hibernate, Struts, and JSF frameworks.

Documentation: DWR provides comprehensive documentation for its libraries.

Reverse AJAX: DWR allows the server to connect with clients and send updated data to clients asynchronously.

Components of DWR

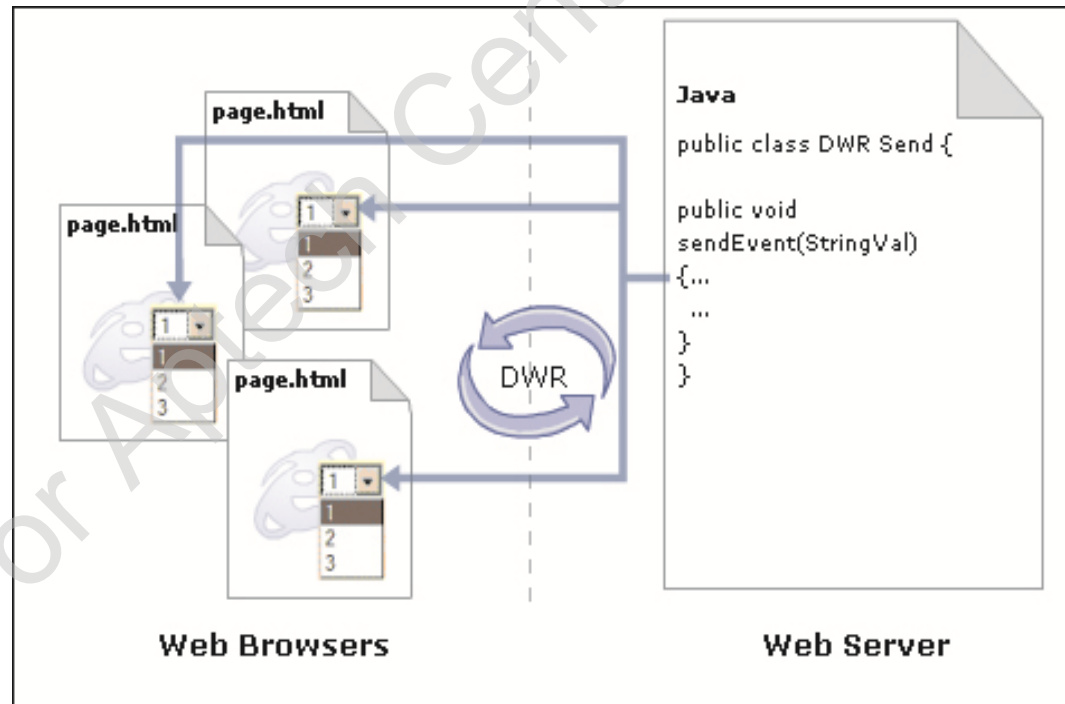
Java Servlet

- Processes client requests and sends responses back to the client.
- DWR contains a runtime library that helps the servlet to process requests and responses.

JavaScript Code

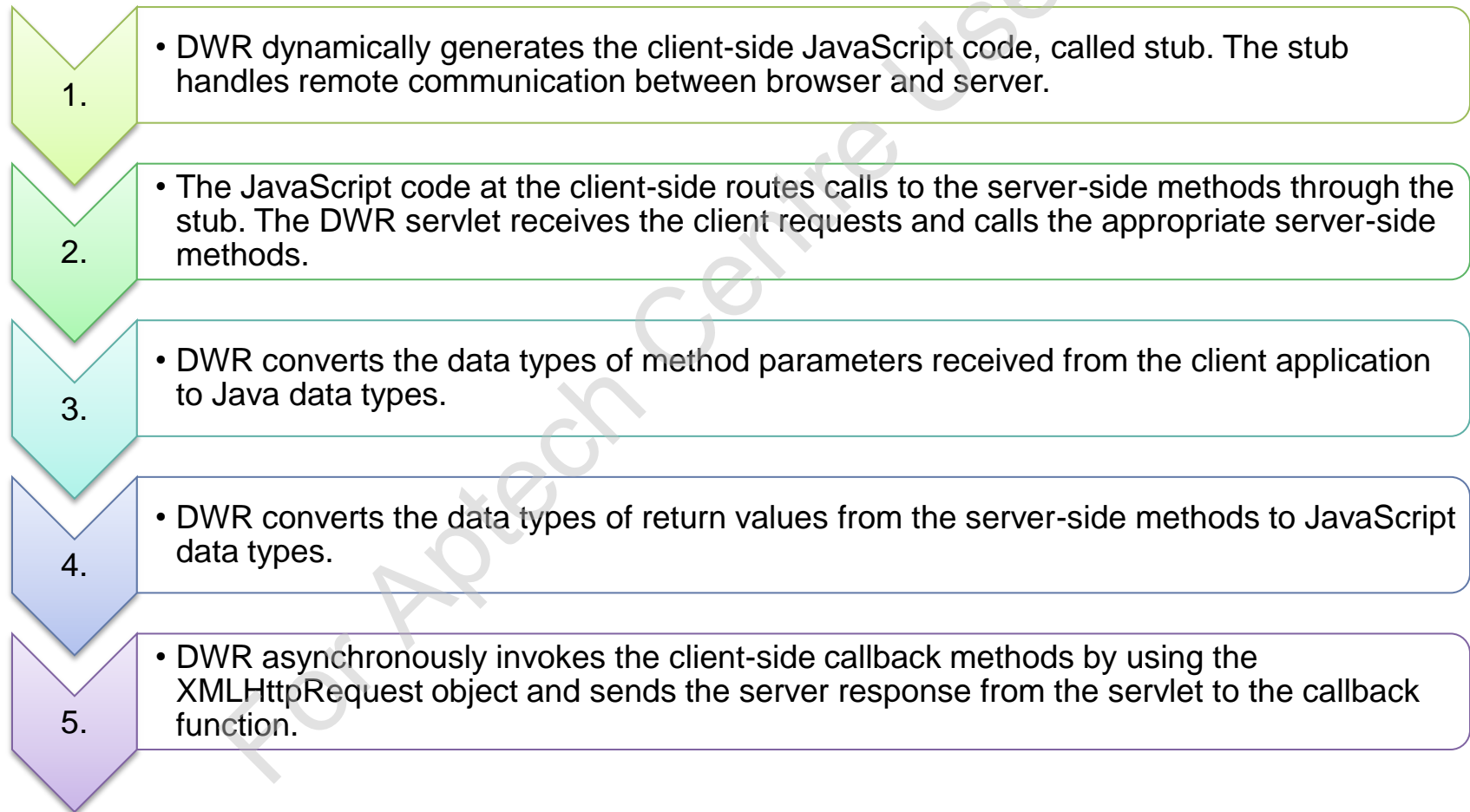
- Runs in the client application.
- It can dynamically update the Web page with the help of a JavaScript library that is part of the DWR architecture.

- ◆ Following figure shows the components of DWR:



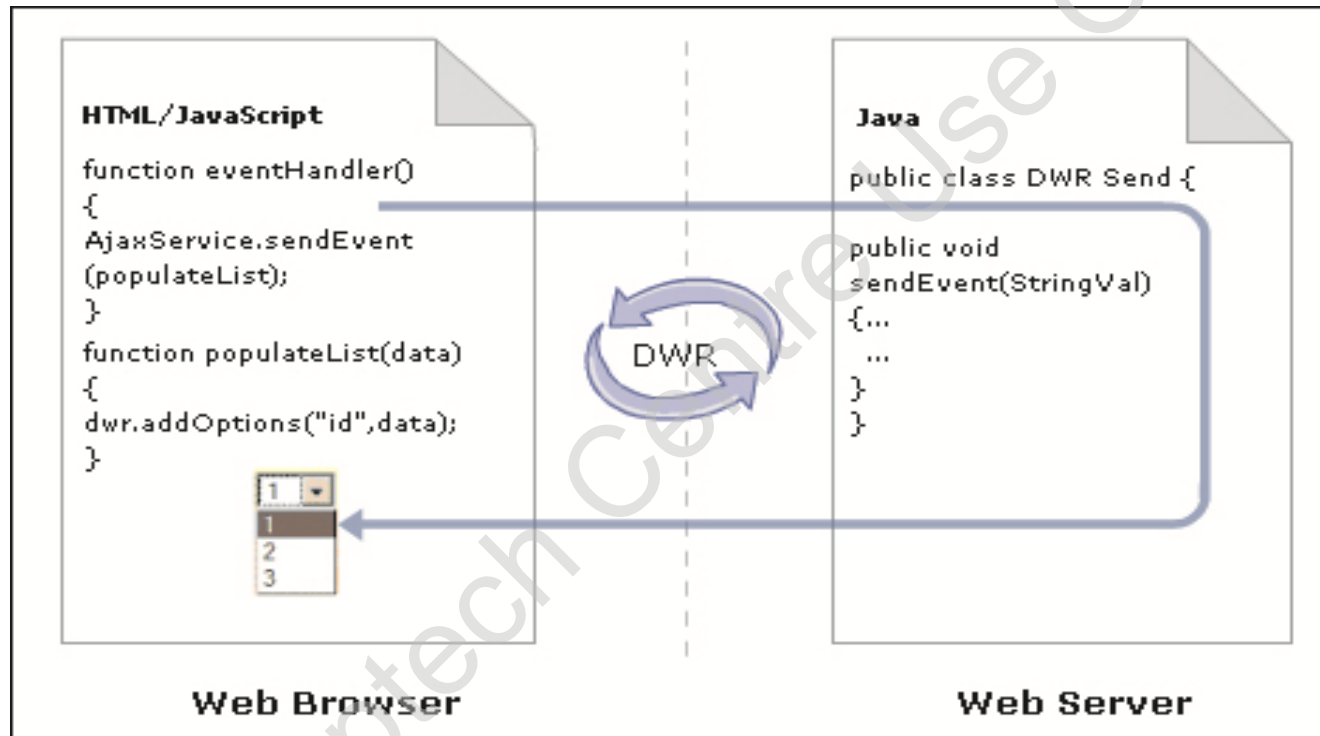
Working of DWR 1-2

- ◆ In DWR-based AJAX applications, the DWR servlet dynamically generates JavaScript classes for each exposed server-side Java class.
- ◆ The workflow of DWR can be summarized as follows:



Working of DWR 2-2

- ◆ Following figure shows the working of DWR:



Handling Asynchronous AJAX Calls

- ◆ In an AJAX application, a client sends a request asynchronously by using JavaScript code.
- ◆ The server that runs Java servlets processes the request synchronously and sends back the response to the client.
- ◆ The processing at server is synchronous because Java technology is synchronous in nature.
- ◆ To enable asynchronous communication between a client and a server, perform the following steps:

1. • Declare a callback function in the client-side code. The callback function handles responses from the server.
4. • Register the callback function with the server.

- ◆ Following Code Snippet demonstrates how to declare a callback function in JavaScript-based client and register it with server:

```
//Declare callback function on
//client-side
function handleGetName(str){
    alert(str);
}

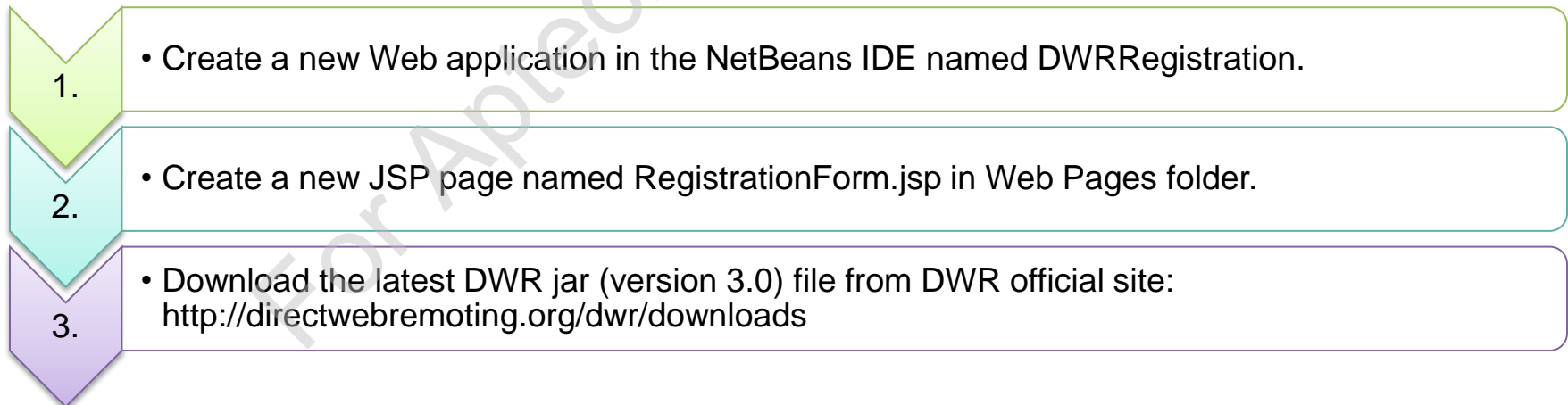
//Server-side Java Class that will be
// remotely accessed by browser
public class Student{
```

```
public String getName(String name) {
    ...
} }

// Invoke remotely getName() from
//client-side and register callback
//function getName()
StudentJavaScript.getName(42,
    handleGetName);
```

Steps of Using DWR in AJAX 1-7

- ◆ Consider a DWR Web application in which user is required to register.
- ◆ The Web application displays the account registration page.
- ◆ The registration page contains textboxes to accept username, password, e-mail address, and a Submit button.
- ◆ When a user enters his/her details and clicks the Submit button, the page waits for a response from the Web server.
- ◆ If the username already exists, the user will be notified about the same on leaving the text box focus.
- ◆ Following steps demonstrate how an AJAX based application can be developed using DWR:



Steps of Using DWR in AJAX 2-7

4.

- Add the dwr-jar file to the Libraries folder of the Web application.

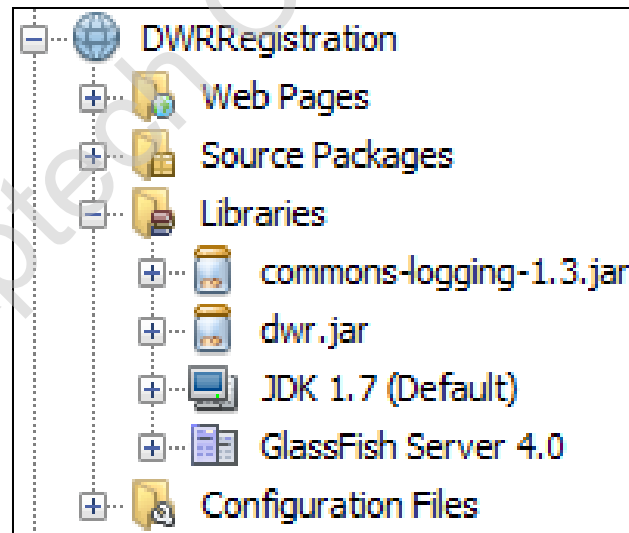
5.

- Download the commons-logging-1.3.jar from <http://www.java2s.com/Code/Jar/c/Downloadcommonslogging13jar.htm>

6.

- Add the commons-logging-1.3.jar to the Libraries folder of the Web application.

- ◆ The directory structure of the Web application is shown in the following figure:



Steps of Using DWR in AJAX 3-7

◆ Server-side Java Class

- ◆ Following Code Snippet demonstrates a server-side Java class that will be exposed for generation of a JavaScript class:

```
package com;

public class Registration {
    private String uname;

    public String getUname() {
        return uname;
    }

    public void setUname(String uname) {
        this.uname = uname;
    }

    public String getMessage(String
username){
        String str = null;
        String[] unames= new
String[4];
        unames[0]="jsmith";
        unames[1]="ashandrews";

        unames[2]="johnsmith";
        unames[3]="maryjoe";

        for (String name : unames) {
            if(name.equalsIgnoreCase(username))
            {
                str= "Username already
exists";
                break;
            }
            else
                str= "";
        }
        return str;
    }
}
```

- ◆ The getMessage() method will be used in the RegistrationForm.jsp page.

Steps of Using DWR in AJAX 4-7

◆ Modify web.xml to Add the DwrServlet Mapping

- ◆ Following Code snippet shows the configuration values for web.xml:

```
<servlet>
<servlet-name>dwr-invoker</servlet-
name>
<servlet-
class>org.directWebremoting.servlet.DW
RServlet
</servlet-class>
<init-param>
<param-name>debug</param-name>
<param-value>true</param-value>
</init-param>
```

```
</servlet>
<servlet-mapping>
<servlet-name>dwr-invoker</servlet-
name>
<url-pattern>/dwr/*</url-pattern>
</servlet-mapping>
```

◆ Configure DWR

- ◆ Following Code Snippet demonstrates the dwr.xml file which determines which Java classes DWR can convert to JavaScript classes:

```
...
<dwr>
<allow>
<convert converter="bean"
match="com.Registration"/>
<create creator="new"
javascript="RegistrationJS"
class="com.Registration">
```

```
<!-- here RegistrationJS is dynamic
javascript class generated by dwr-->
<param name="class"
value="com.Registration"/>
<include method="getMessage"/>
</create>
</allow>
</dwr>
```

Steps of Using DWR in AJAX 5-7

◆ Client-Side JavaScript

- ◆ The JavaScript files engine.js and util.js should be included in the client-side application. The file RegistrationJS.js is the generated JavaScript file.
- ◆ Following Code Snippet demonstrates the RegistrationForm.jsp page where the data from the server is retrieved through DWR:


```
<script type='text/javascript' src='/DWRRegistration/dwr/engine.js'></script>
<script type='text/javascript' src='/DWRRegistration/dwr/util.js'></script>
<script type='text/javascript'
src='/DWRRegistration/dwr/interface/RegistrationJS.js'></script>
<script type='text/javascript'>
function update(){
    var uname;
    //debugger;
    uname=document.getElementById("txtUname").value;
    Registration.getMessage(uname,function(data) {
        dwr.util.setValue("message", data);
    });
}
function register(){
    alert("Registration Completed");
}

</script>
```


Steps of Using DWR in AJAX 6-7

[illegible]

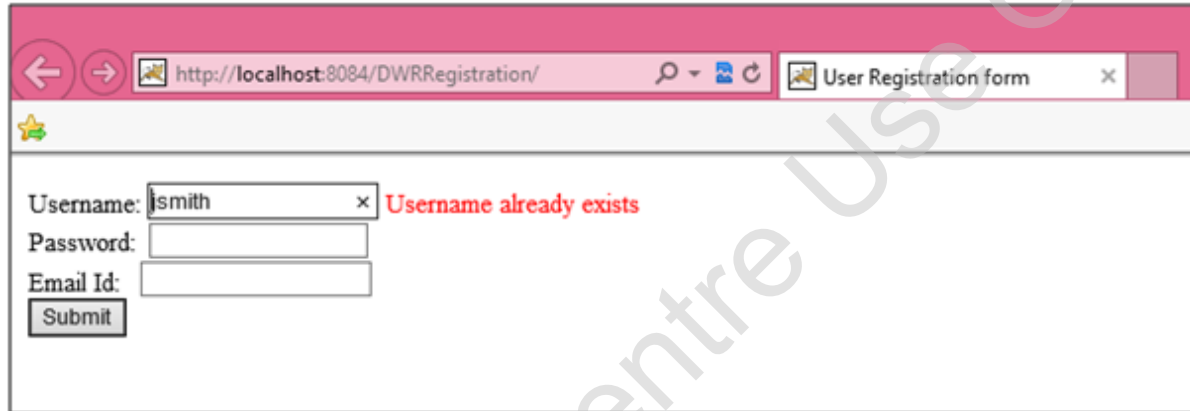
- ◆ The update method is called on blur event of the text box. The update function calls the JavaScript class RegistrationJS.js which is generated automatically by DWR.
- ◆ Following figure shows the output of RegistrationForm.jsp:



The screenshot shows a web browser window with the address bar displaying `http://localhost:8084/DWRRegistration/`. The browser has a single tab titled "User Registration form". The page content includes three text input fields labeled "Username:", "Password:", and "Email Id:", followed by a "Submit" button. A large, semi-transparent watermark "For Apple" is visible across the center of the browser window.

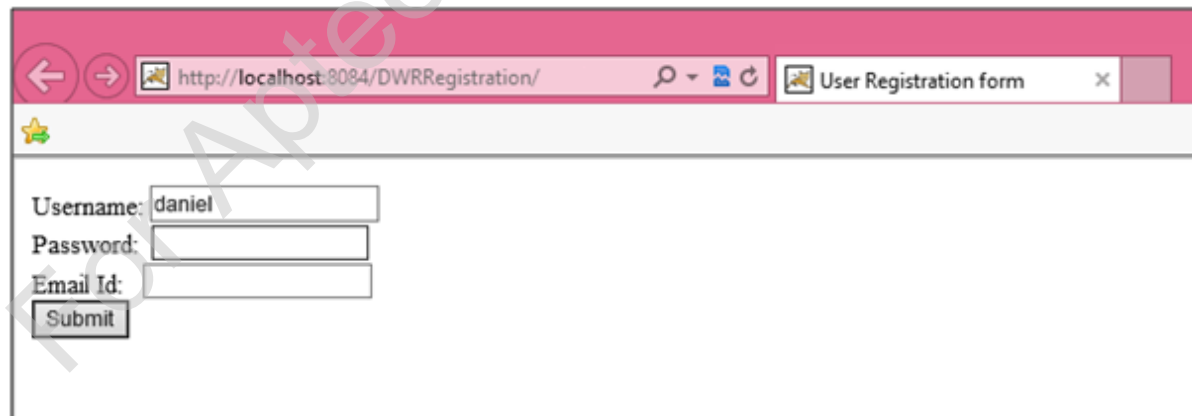
Steps of Using DWR in AJAX 7-7

- ◆ When an existing username is entered for registration, the appropriate message is displayed as shown in the following figure:



A screenshot of a web browser window titled 'User Registration form'. The address bar shows 'http://localhost:8084/DWRRegistration/'. The form contains three input fields: 'Username:' with the value 'jsmith', 'Password:', and 'Email Id:'. A red error message 'Username already exists' is displayed next to the Username field. A 'Submit' button is located below the input fields.

- ◆ The message is retrieved from the server after checking the existing names in the array. If the name does not exist, no such message is displayed as shown in the following figure:



A screenshot of a web browser window titled 'User Registration form'. The address bar shows 'http://localhost:8084/DWRRegistration/'. The form contains three input fields: 'Username:' with the value 'daniel', 'Password:', and 'Email Id:'. No error message is displayed. A 'Submit' button is located below the input fields.

Reverse Ajax 1-2

- ◆ To describe the asynchronous transfer of messages from browser to server and vice-versa, DWR introduced a new term, 'Reverse Ajax'.
- ◆ With Reverse Ajax, DWR permits Java code to run on a server to use client side APIs.
- ◆ It permits a two way interaction, that is, a browser calling a server and a server calling a browser.
- ◆ The Reverse Ajax feature neatly wraps all the existing techniques and selects the best method for the user.
- ◆ DWR supports three techniques which are as follows:

Polling

- At frequent and regular intervals, request is sent by browser to the server to check if the page has been updated.

Piggyback

- Server waits for the browser request if it has any update to send and in addition to the requested information, also sends the update.

Comet

- The server keeps the reply open after a request has been received from the browser, so that it can pass the information when the request arrives.
- This technique is also called as a long-lived HTTP connection.

Reverse Ajax 2-2

- ◆ In the Web page, for allowing active Reverse Ajax (ARA) to occur, the following line should be included:
 - ◆ `<body onload="dwr.engine.setActiveReverseAjax(true);">`
- ◆ Following Code Snippet should be included in the web.xml file:

```
<init-param>
    <param-name>pollAndCometEnabled</param-name>
    <param-value>true</param-value>
</init-param>
```

- ◆ For Comet/Polling DWR can be configured when extra load is acceptable and for having faster response time.
- ◆ This mode is known as active Reverse Ajax.
- ◆ By default Reverse Ajax is turned off in the DWR and thus, allows only piggyback mechanism by default.
- ◆ It is easy to use Reverse Ajax and DWR from a thread outside of DWR.

Alternate Technologies

- ◆ DWR is a remoting technology that is based on Java. It uses a custom protocol for remoting.
- ◆ Alternative remoting technologies available are as follows:

XML-RPC

- XML-RPC is a lightweight protocol that can exchange structured information in XML format between nodes of a distributed environment.
- It is independent of any programming language.

SOAP

- Simple Object Access Protocol (SOAP) is a protocol that exchanges XML-based messages in distributed environment using HTTP/HTTPS.
- SOAP provides a basic messaging framework for Web services.

JSON-RPC

- JSON-RPC is a protocol that allows bidirectional communication between server and client, unlike XML- RPC or SOAP.
- JSON-RPC allows peer-to-peer communication between server and client. It contains only a few commands and data types.

Summary

- ◆ JSON is a subset of JavaScript language that is used for interchanging of data between a browser and the server. It is easier to manipulate as compared to XML.
- ◆ JSON supports data types such as string, number, array, and boolean.
- ◆ JSON can be used to pass data asynchronously using AJAX between the client and server.
- ◆ DWR is an AJAX framework that can generate JavaScript code from Java classes. It is based on Java technology.
- ◆ A client browser can access remote Java classes that run in the Web server by invoking the generated JavaScript functions.
- ◆ The Java classes that would be accessible to the client browser can be decided by configuring DWR.
- ◆ The Reverse Ajax feature introduced in DWR 2.0 neatly wraps all the existing techniques and selects best method for the user.