Ajax Client Runtime in IBM WebSphere Application Server Web 2.0 and Mobile Toolkit Version 1.1.0

Contents

Ajax Client Runtime

Welcome to the Ajax Client Runtime. This section provides an overview of the Ajax Client Runtime and information about the Dojo Toolkit, a core component of the runtime.

This `85528_WAS_web2mobile-ajaxruntime-help.pdf` file derives from the 8.5.5 `com.ibm.websphere.web2mobile.ajaxruntime.help` plug-in that was used in WebSphere Application Server 8.5.5.28 and previous versions. The `com.ibm.websphere.web2mobile.ajaxruntime.help` plug-in is available in the `855_WAS_all_product_doc.zip` file at `https://public.dhe.ibm.com/software/webserver/appserv/library/v85/`.

Avoid trouble: The Ajax Client Runtime is deprecated. Do not use Ajax Client Runtime for new applications. Dojo Toolkit, including Dojo Diagrammer - Dojo desktop and mobile applications continue to be supported through Dojo 1.10. Dojo 1.11 and later are not supported. You can continue to develop Dojo applications by using the open source Dojo Toolkit. For more information, see statements about the Web 2.0 and Mobile Toolkit deprecated feature in Deprecated features.

This section covers two topics:

Overview
A brief overview of the Ajax Client Runtime and what it provides to developers.

The Dojo Toolkit
Information on a core component of the Ajax Client Runtime, the Dojo Toolkit.

Getting started with the Ajax Client Runtime

The Ajax Client Runtime is a powerful JavaScriptTM development comprised of Dojo Toolkit and a set of extensions provided by IBM. This document is the starting point for learning more about the Ajax Client Runtime.

The following topics are covered:

Overview
The overview of the Ajax Client Runtime.

Toolkits
The Dojo Toolkit

Overview of the Ajax client run time

The Ajax client run time is an Ajax development toolkit that is based on an IBM-supported version of the Dojo Toolkit.

Ajax client runtime features

The Ajax client runtime is based on an IBM supported version of the Dojo Toolkit. In addition to an extensive pedigree review, the Ajax client run time includes extensions to the Dojo Toolkit that supports the following areas:

- IBM ILOG Diagrammer for working with advanced diagramming and layout capabilities.
- SOAP client for working with SOAP envelopes and RPC.

Package structure

Three package structures distribute this library.

IBM WebSphere Application Server Web 2.0 and Mobile Toolkit
The runtime is distributed as a set of JavaScript, HTML, and documentation installed by the toolkit into your Web 2.0 and Mobile Toolkit home directory. From this directory, users can copy the files into their web archive (WAR) files as static resources, or they can be copied to a web server as static content, whichever best fits your usage. This copy is a nonoptimized (nonprofile) build of Dojo. For the best performance in a particular application, building a profile of the Dojo Toolkit from this set of files is necessary . The Dojo Toolkit community provides documentation and a tool for constructing profiles.

The Ajax client run time is located in the following directory:

Directory	Description
`<toolkit_root>/ajax-rt_1.X/`	The non optimized Ajax client run time (The Dojo Toolkit plus extensions).

The Dojo Toolkit

The Dojo Toolkit is a powerful and flexible modular Ajax software development kit. It is broken down into three major layers: Dojo Core, Dijit, and DojoX.

Dojo Toolkit layers

Dojo Core
All the major functions needed to do Ajax development, plus many features that are not found in other toolkits. Refer to Dojo Core reference documentation.

Dijit
A high quality set of interaction rich widgets and themes for use when developing Ajax applications. Refer to Dijit reference documentation.

DojoX (Dojo eXtensions)
A module to contain widgets and APIs that are useful for developing Ajax applications, but are not needed in all applications. Refer to DojoX reference documentation.

The Dojo Toolkit website serves as a guide to these layers, introducing concepts as you need them and working downward from high-level usage to build your own widgets, custom namespaces, and unit tests.

Dojo Toolkit versions

This version of the Toolkit provides you with Dojo Toolkit 1.7.X and Dojo 1.8.X

Dojo 1.7.X includes some new modules from the Dojo Toolkit 1.8. These modules are the following:

DojoX Calendar
A Calendar widget that allows you to visualize and edit events. Refer to DojoX Calendar documentation.

DojoX Gauges
A Gauges framework that provides some predefined gauges and allows you to easily create your own. Refer to the DojoX Gauges documentation.

DojoX Mobile
A full featured mobile widget library with various mobile themes. Refer to DojoX Mobile documentation.

DojoX TreeMap
A Treemap widget that allows you to visualize and analyze complex data sets. Refer to DojoX TreeMap documentation.

Additional information

The Dojo version that is provided with the Web 2.0 and Mobile Toolkit is a selected version containing only Dojo packages that are considered stable and fully tested. More packages available for download are contained to DojoX. Refer to the installed readme file under the `ajax-rt_1.X and ajax_rt_1.8` directory for detailed information on more DojoX modules.

Overview of the IBM SOAP library

The `ibm_soap` package contains two parts: the SOAP service and the SOAP service widget, which provide functions for creating SOAP envelopes and connecting to external SOAP services.

SOAP library components

The `ibm_soap` package provides the following functions:

SOAP service
: This library extends the dojo.rpc.RpcService class and provides an easier way to create the SOAP envelope around a request.

SOAP widget
: This widget uses the SOAP service and enables a convenient way to connect to external SOAP services and start their methods in a simple way.

These components are examined further in the Reference and Usage examples topics. More information is available in the following topics:

- Usage Examples: Usage examples for the IBM SOAP library
- Tests: Tests for the IBM SOAP library
- Reference: IBM SOAP library API reference

Package structure

The SOAP library is packaged as a part of the Ajax client runtime, which is distributed under three different variations. Refer to the corresponding documentation for further details about the package structure. Internally, the library is organized in the following way:

`/io`

- ws.js: Helper classes for building the SOAP envelope

`/rpc`

- SoapService.js: The SOAP service library

`/tests`
: Unit tests

`/util`

- WsdlParser.js: Parses `.wsdl` files and can convert the service description to JavaScript Object Notation (JSON) format

`/widget`
: SOAP library widget

Usage examples for the IBM SOAP library

The SOAP service can be instantiated using the service description from either a local or a remote file. The service description can be either in a `.smd` or a `.wsdl` file format.

Instantiating the SOAP service

See the following usage example:

```
<div dojoType="ibm_soap.widget.SoapService" id="bnpriceService"
 url="./bnpriceGenerated.smd">
```

```
<div dojoType="ibm_soap.widget.SoapService" id="amazonCommerceService"
 url="./AWSECommerceService.wsdl">
```

Provide the location of the service description as the URL. After the widgets are parsed and instantiated, the service is ready for use. Typically, you must begin with a service that provides a service description as a `.wsdl` file. You can use the provided WsdlParser to convert to `.smd` format. See the following example usage:

```
var parser=new ibm_soap.util.WsdlParser();
parser.parse("./serviceDescription.wsdl");
var smdString = parser.smdString;
```

When you parse a WSDL format description, you can access the results by using the `smdString` or the `smdObject` member of the parser. Consider the following items when converting the service description:

- Any complex type information that is contained in the service description is not converted to the JavaScriptTM Object Notation (JSON) format. The type is considered as object. Refer to the original Web Service Definition Language (WSDL) file to correctly form the parameters.
- The WsdlParser accepts a WSDL description in one of the following ways:
  - URL: If the argument passed in to the parse method is a string, then that argument is assumed to be a URL to the `.wsdl` file, for example:

    ```
    parser.parse("./serviceDescription.wsdl")
    ```

  - String: If the argument is not a string, then the parser searches for a member that is named `wsdlStr` in the object that is passed in, and evaluates the value to retrieve the service description, for example:

    ```
    // Assume that the variable wsdlStr holds the WSDL description as a string
    var args = new Object();
    args.wsdlStr = wsdlStr
    parser.parse(args);
    ```

Calling a service method

To call a method described by the service, you need to know the method name and the structure of the parameters that are expected by the method. For instance, see the example of how to call the ItemLookUp method for the Amazon service. First, build the parameter list:

```
// Assume accessKeyId holds the AccessKeyId required by Amazon.
var amazonServiceParms = new dojox.wire.ml.XmlElement("ItemLookUpRequest");
amazonServiceParms.setPropertyValue("AWSAccessKeyId",accessKeyId);
amazonServiceParms.setPropertyValue("ItemId",isbn);
```

Set the required properties by calling the setPropertyValue method. The variable, *isbn*, stores the ISBN that is to be searched. Next, invoke the service method using the *service* member of the widget as the following example:

```
var deferred = amazonCommerceService.service.ItemLookup(amazonServiceParms)
```

The call returns an object of type *dojo.Deferred* that contains the BODY node of the returned content. You can use simple Document Object Model (DOM) manipulation to access the result. See the following example of accessing the results from the returned data:

```
deferred.addCallback(function(results){
 var title = results.getElementsByTagName("Title")[0].firstChild.nodeValue;
```

```
  var author = results.getElementsByTagName("Author")[0].firstChild.nodeValue;
  var publisher = results.getElementsByTagName("Manufacturer")[0].firstChild.nodeValue;
  var resultNode = dojo.byId("details");
  resultNode.innerHTML = "<br><b>Title:</b>"  + title
    + "<br><b>Author:</b>" + author
    + "<br><b>Publisher:</b>" + publisher;
});
```

Proxy requirement

Because of cross-domain XMLHttpRequest (XHR) restrictions, you might need to use a proxy to reach the service endpoints that are defined in the service description. You must modify the service endpoints to incorporate the proxy. The modified version of the Barnes and Noble price service might resemble the following example:

```
<app name>/ajaxProxy/www.abundanttech.com/WebServices/bnprice/bnprice.asmx.
```

After you change the service endpoint URLs in the service description file, you might also need to perform additional configuration for the proxy to forward requests to the external server. Refer to the documentation provided with the proxy that you are using for more information.

The IBM SOAP library tests

The IBM SOAP library tests are a series of functional tests that verify that the JavaScriptTM SOAP API behaves as expected with a series of inputs.

The tests are based on the Dojo Objective Harness (D.O.H.) framework for unit testing JavaScript code. For more information about D.O.H., see the Dojo Toolkit documentation on D.O.H.

Test areas

The following areas are tested:

- Loading the SMD for the SOAP service from different locations:
  - A URL
  - A string
  - An SMD object

Running the tests

You can run the tests by loading the `ibm_soap/tests/runTests.html` file into a supported web browser. You can view the implementation of the tests by loading the `ibm_soap/tests/rpc.js` file into your favorite text editor.

IBM SOAP library reference

The SOAP library primarily consists of the SOAP library, itself that is used to instantiate the service and the utility functions that you use to convert Web Service Definition Language (WSDL) files into an equivalent `.smd` description file. In addition, a widget is provided that uses the SOAP library.

SOAP service

The SOAP service provides functions for performing asynchronous communication with the server, parsing results, generating methods, and processing service descriptions.

SOAP service functions

Function	Description
bind (String method, Array parameters, Object deferredRequestHandler, String URL, Object soapParms)	Performs the asynchronous communication with the server
parseResults (Object data)	Parses the results that are received from the server
generateMethod (String method, Array parameters, String URL, Object soapParms)	Generates methods that can be called to invoke the service methods
processSmd (Object object)	Processes the service description

SOAP service function details

bind

bind: void bind(String method, Array parameters, Object deferredRequestHandler, String URL, Object soapParms)

Method: String - Name of the service method to invoke

Parameters: Array - Array of parameters to pass to the service method to be invoked

deferredRequestHandler : Object - The method that is used to handle the results of the asynchronous call to the server

URL : String - Service endpoint URL

soapParms : Object - Contains SOAP-specific information that is useful for creating the SOAP envelope

parseResults

parseResults (Object data)

data : Object - Data received from the server

generateMethod

generateMethod (String method, Array parameters, String URL, Object soapParms)

method : String - Name of the method to generate

Parameters: Array - Array of parameters to be passed to the service method

URL : String - Service endpoint URL

soapParms : Object - Contains SOAP-specific information that is useful for creating the SOAP envelope

processSmd

processSmd (Object object)

object Object - Service description object

SOAP service widget variables

Variable	Type	Description
url	String	Service description URL
serviceUrl	String	Service endpoint URL
service	Object	SOAP service object that has the various methods to be invoked

SOAP service widget functions

Function	Description
setUrl	Sets the URL to the service description document
setServiceUrl	Sets the URL to the service endpoint
callMethod	Invokes a service method with the given parameters

SOAP service widget function details

setUrl

setUrl(String url)

url: String - Service description URL

Parameters:

setServiceUrl

setServiceUrl(String serviceUrl)

serviceUrl : String - Service endpoint URL

callMethod

callMethod(String method, Array parameters)

method : String - Name of the service method to invoke

parameters: Array - Array of parameters to use for invoking the specified method

WSDL parser variables

Variable	Type	Description
wsdlObj	Object	Service description as a WSDL object
wsdlString	String	Service description as a WSDL string
smdObj	Object	Service description as a JSON Object
smdString	String	Service description as a JavaScript Object Notation (JSON) string

WSDL parser functions

Function	Description
parse	Parses the service description from WSDL format to JSON format

WSDL parser function details

parse

parse(Object obj)

obj: Object - Parameter that contains the WSDL description

Index