Elevate Web Builder 2 Modules Manual

Table Of Contents

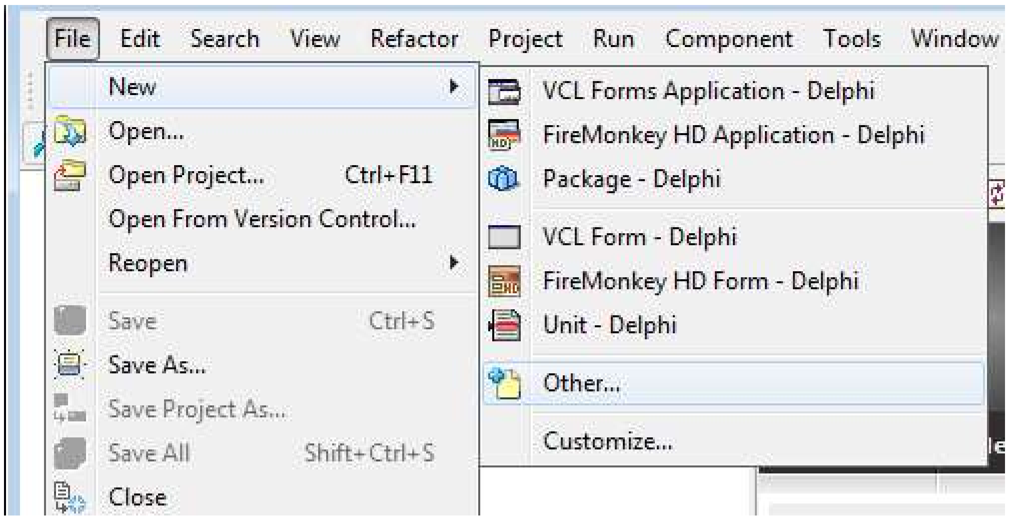
This page intentionally left blank

Chapter 1
Getting Started

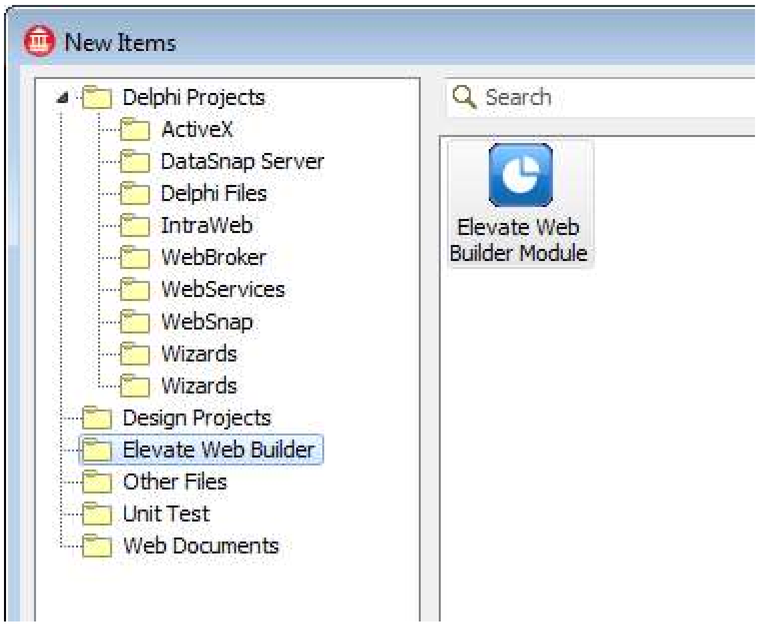
1.1 Creating a Module

To create a new module, please complete the following steps:

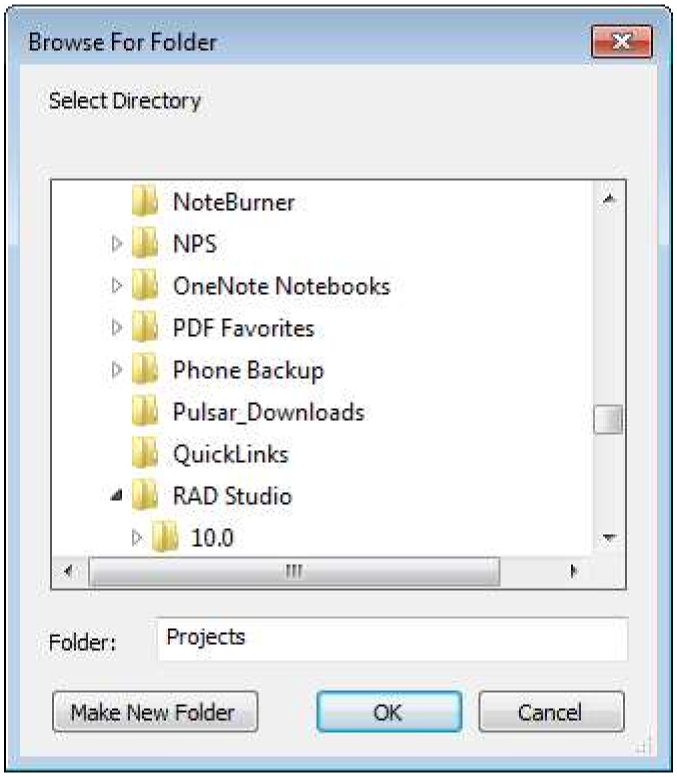
1. In the RAD Studio IDE, select the **File/New/Other** menu option from the main menu.



2. The **New Items** dialog will then appear. Select the **Elevate Web Builder** option folder from the list on the left-hand side of the dialog. Click on the **Elevate Web Builder Module** and then click on the OK button.



3. The **Browse for Folder** dialog will then appear. Select the desired target folder for the new module project and click on the OK button.



Folder: Projects

4. The new module project will be created in the desired target folder and opened as the active project in the RAD Studio IDE.

Please see the Handling Requests topic for more information on adding the appropriate code for handling incoming requests to the module.

1.2 Database Modules

Database modules are exactly like any other module, but are coded to handle database operations using the TEWBDatabaseAdapter and TEWBDataSetAdapter components. Specifically, any incoming request can be passed directly to the TEWBDatabaseAdapter HandleRequest method and the database adapter component will completely handle the database request, calling the applicable TEWBDatabaseAdapter and TEWBDataSetAdapter methods and events as necessary to complete the request and send back a response.

Please see the **Configuring the Web Server** section in the Elevate Web Builder manual that is included with the Elevate Web Builder IDE for more information on configuring the database modules resource name used with database module requests.

Database Module Example

The Elevate Web Builder 2 Modules installation includes an example database module project that shows how to load a dataset from an ElevateDB database. This project also demonstrates the usage of the TEWBDatabaseAdapter and TEWBDataSetAdapter components for generating/consuming JSON from TDataSet-descendant component instances in Embarcadero RAD Studio and Delphi. It is installed into the \examples\databasemodule subdirectory under the main installation directory of the Elevate Web Builder 2 Modules installation.

In addition, a database module client example project called databaseclient.wbp is automatically installed with the example applications that come with Elevate Web Builder 2. It provides the front-end for accessing the dataset that is handled by the database module example project. A pre-compiled copy (databasemodule.dll) of the databasemodule example project will be also be located in the \bin\databasemodule\win32 subdirectory under the main installation directory of Elevate Web Builder 2, and this pre-compiled database module will be added to the IDE during the example installation so that it can be used with the database module client example project.

1.3 Handling Requests

Note
Before reading the following material, please be sure that you understand how HTTP requests work by reading the **Using Server Requests** and **Using the Web Server** sections of the Elevate Web Builder manual that is included with the Elevate Web Builder IDE. These sections contain important information such as how server requests work and how HTTP headers are formatted.

Incoming Requests

When an HTTP request is received by the Elevate Web Builder Web Server, the URL of the request is checked to see if the request is for static content or a database request. If the URL for the request is not that of static content or a database request, then the URL is checked to see if it is an Elevate Web Builder module request. A module request is any request that uses one of the following URL structures:

Normal Module

```
http://<Domain Name>/<Modules Resource Name>/<Module Name> or
https://<Domain Name>/<Modules Resource Name>/<Module Name>
```

Database Module

```
http://<Domain Name>/<Database Modules Resource Name>/<Database Module Name>
      or
https://<Domain Name>/<Database Modules Resource Name>/<Database Module Name>
```

Note
The <Module Resource Name> and <Database Module Resource Name> components of the URL are the default resource names for modules and database modules defined in the Elevate Web Builder Web Server, but can be changed in the web server configuration. If you've changed the default modules resource name of 'modules', then please replace any subsequent references to the default 'modules' resource name in the following examples with the resource name that you're using instead. The same holds true for the default database modules resource name of 'databasemodules'.

If the URL matches either of the above patterns, the web server will automatically instantiate a TEWBModule component to represent the module specified in the URL for use with the request. The web server will then pass the request information to the module instance by firing its OnExecute event handler. The request is passed to the OnExecute event handler as an instance of the TEWBServerRequest object. This TEWBServerRequest object instance is used both to retrieve information about the request and to send content as a response to the request.

For database module requests, you can use the OnExecute event handler to pass the incoming request directly to the TEWBDatabaseAdapter HandleRequest method and the database adapter component will completely handle the database request, calling the applicable TEWBDatabaseAdapter and TEWBDataSetAdapter methods and events as necessary to complete the request and send back a response.

Warning
Each module instance is executed in a separate thread, so you must make sure that all code included in an OnExecute event handler is completely thread-safe. Also, all modules in the web server run in-process, so a fatal error such as a memory overwrite due to improper threading code in a module could cause the server to fail.

Determining if a Request Is Secure

You can determine if a request is secure (using HTTPS) by examining the RequestSecure property of the incoming request.

Note
The web server does not currently support secure requests.

Determining the Request Type

Module requests can be HTTP HEAD, GET, POST, PUT, and DELETE requests. You can determine the request type by examining the RequestMethod property of the incoming request. HEAD requests are normally associated with static resources such as files, but they can be sent to modules when the content type of the resource being handled by the module is unknown to the client application. PUT and DELETE requests are normally associated with REST APIs, which are possible to handle using modules.

Determining the Browser Type

You can determine the type of browser that is submitting the request by examining the following properties of the incoming request:

- RequestIE
- RequestFireFox
- RequestChrome
- RequestOpera
- RequestSafari

Note
These properties are set by the web server examining the RequestHeaders that are passed to the web server as part of the request, and therefore can be easily forged by any client that is trying to impersonate a particular browser. However, for the majority of clients, these properties will accurately reflect the browser being used on the client.

Reading URL Parameters

Parameters are specified in a URL in the following manner:

```
<BaseURL>?<Params>

<Params> = <Param> [? <Param>]

<Param> = <Key>=<Value>
```

For example, the following URL will result in a request to the module called "Compute" with parameters for X and Y axis values:

```
http://www.domain.com/modules/compute?x=100&y=200
```

The URL and any parameters included in the URL are specified in the request's RequestURL and RequestURLParams properties, respectively. The RequestURLParams property, however, is in its raw form. An easier way to examine the URL parameters would be to use the RequestParams property. The RequestParams property is a TStrings instance of key-value pairs that represent all URL parameters. To use the above example, the following values would be present in the RequestParams property for the specified URL:

```
x=100
y=200
```

Reading Request Cookies

Cookies are simple textual data that is persisted in the client across connections to the web server. Any cookies that apply to the request URL will automatically be sent by the client and will be available in the TServerRequest RequestCookies property. The RequestCookies property is a TStrings instance of key-value pairs that represent all applicable cookies. See the **Sending a Response** section below for information on how to set cookies in responses.

Authenticating a Request

The Elevate Web Builder Web Server supports two different forms of automated authentication:

- Custom HTTP headers
- URL Parameters

Custom HTTP Authentication Headers

The custom HTTP authentication headers must be sent to the web server as follows:

```
X-EWBUser: <UserName>
X-EWBPassword: <Password>
```

URL Authentication Parameters

The URL authentication parameters must be sent to the web server as follows:

```
user=<UserName>&password=<Password>
```

Note
By default, all database requests sent by an Elevate Web Builder application use custom HTTP authentication headers. However, database requests can also be configured to use URL parameters for authentication.

If the Elevate Web Builder application, or any client application, sends authentication information using either method described above, then the Elevate Web Builder Web Server will automatically populate the incoming TServerRequest RequestUser and RequestPassword properties with the authentication information provided by the client application.

In order to handle verifying the provided authentication information, you can call the AuthenticateUser method. This method call will trigger the OnAuthenticateUser event, and if the event handler sets the Result parameter to True, then the module will populate the UserName and Password properties with the authentication information.

Reading Request Content

POST requests are normally accompanied by content that is submitted as part of the request. The RequestContentType, RequestContentLength, and RequestContent properties of the request contain the content type, content length, and actual content.

HTML Form Submittals

When an HTML form is submitted to the URL for the module, the form data may be sent over using one of two formats. Depending upon value of the RequestContentType property, the web server will automatically perform some pre-processing of the incoming conent in the following ways:

- **application/x-www-form-urlencoded**

  This is the default encoding for form submittals, and common for form submittals that do not include file uploads. The web server will pre-parse all textual form variables and they are accessible via the RequestContentParams property as key-value pairs

multipart/form-data

If a form submittal includes one or more file uploads, then the browser will use a special MIME encoding. The web server will pre-parse all textual form variables as with the default encoding, but will also include any file content in the RequestMIMEParts property, with each file in its own TEWBMIMEPart instance. Each TEWBMimePart instance includes both Headers and Content. The Headers property contains MIME headers, which are formatted in the same fashion as HTTP headers. These headers can be used to determine how to deal with the Content property, and the most useful are:

Header	Description
Content-Transfer-Encoding	This header identifies the encoding of the file content contained in the Content property. This is essential in knowing how to deal with the content.
Content-Disposition	This header identifies the actual name of the file. This is useful if the file content must be saved under its source name and then later retrieved using its file name.

Sending a Response

The web server doesn't automatically send a response to an HTTP request once the OnExecute event handler terminates, so it is important that you send a response from within the OnExecute event handler. You can use one of several methods to send a response to the client's request:

Method	Description
SendContent	Sends Unicode content as a UTF-8 string along with an optional HTTP status code and message.
SendContentHeader	Sends an HTTP status code and message in response to a HEAD request.
SendContentStream	Sends content as a binary stream along with an optional HTTP status code and message.
SendCustomContent	Sends Unicode or ANSI content with a specific disposition. The disposition is used to give the client an indication of how the content should be saved, such as the file name for a file. **Note** There are two SendCustomContent methods, one for Unicode strings and one for ANSI strings.
SendCustomContentHeader	Sends a content type and disposition in response to a HEAD request. The disposition is used to give the client an indication of how the content should be saved, such as the file name for a file.
SendCustomContentStream	Sends content as a binary stream with a specific disposition. The disposition is used to give the client an indication of how the content should be saved, such as the file name for a file.
SendRedirect	Sends the new location (URL) for a resource (if its location has been changed) along with a optional Unicode content as a UTF-8 string, HTTP status code, and message.
SendError	Sends a Unicode error message along with an HTTP status (error) code.

Warning
Once one of the above methods is called, you should clean up any allocated resources and exit the OnExecute event handler. Not only is it important that you call one of the Send methods, it is also equally important that you **do not** call one of the above methods more than once, or more than one method per request.

In an Elevate Web Builder module, there are defined constants that represent the common HTTP status codes, and you will find these constants in the ewbhttpcommon unit, which is distributed as a .dcu (Delphi compiled unit) with the Elevate Web Builder Modules installlation.

The constants are defined as follows:

```
   HTTP_NONE = 0;
   HTTP_CONTINUE = 100;
   HTTP_OK = 200;
   HTTP_MOVED_PERMANENTLY = 301;
   HTTP_FOUND = 302;
```

```
    HTTP_SEE_OTHER = 303;
    HTTP_NOT_MODIFIED = 304;
    HTTP_MOVED_TEMPORARILY = 307;
    HTTP_BAD_REQUEST = 400;
    HTTP_NOT_FOUND = 404;
    HTTP_NO_LENGTH = 411;
    HTTP_INTERNAL_ERROR = 500;
    HTTP_NOT_IMPLEMENTED = 501;
    HTTP_SERVICE_UNAVAILABLE = 503;
```

Setting Cookies

As indicated above, any cookies that are stored on the client and are applicable to the request URL can be found in the RequestCookies property. You can add or modify such cookies by assigning key-value pairs to the TServerRequest ResponseCookies or ResponseSessionCookies properties. The key is the name of the cookie, and the value is the value to assign to that cookie. To "delete" a cookie, simply set its value to nothing.

Note
You must set any cookies **before** calling any of the TServerRequest Send* methods detailed above, or the cookies will not be set properly on the client.

There are three cookie attributes that are also important. Cookie attributes are specified after the value of a cookie and are always prefixed by a semicolon.

Attribute	Description
expires	Specifies the date/time that the cookie expires as a GMT string. To convert a TDateTime into a GMT string, use the TServerRequest DateTimeToGMTStr method. If you do not specify an expiration attribute, then the web server will automatically set the expires attribute as 1460 days from the current date/time for all cookies specified in the ResponseCookies property. The web server will not set the expires attribute for any cookies specified in the ResponseSessionCookies property, which means that they will automatically not persist past the current client session. This is useful for session IDs and other types of information that you do not want to be present in a new or different client session.
domain	Specifies the domain that the cookie applies to. If you do not specify a domain attribute, then the web server will automatically set the domain attribute to that of the configured domain for the web server. Please see the Elevate Web Builder manual for more information on configuring the web server domain.
path	Specifies the path attribute that the cookie applies to. If you do not specify a path attribute, then the web server will automatically set the path attribute to a single forward slash, meaning that the cookie applies to all paths within the cookie's domain.

This page intentionally left blank

Chapter 2
Component Reference

2.1 TEWBDatabaseAdapter Component

Unit: ewbdatasetadapter

Inherits From TComponent

The TEWBDatabaseAdapter component is used in conjunction with various TEWBDataSetAdapter components to create and load JSON dataset columns and rows to/from generic TDataSet descendant components, as well as commit transaction operations to one or more datasets. It is used with database modules that need to deal with data from a data source that isn't supported by Elevate Web Builder Web Server's automatic database handling or need to perform special processing/authentication for database requests.

Properties	Methods	Events
InTransaction	AuthenticateUser	OnAuthenticateUser
Password	BuildDataSets	OnCommit
UserName	Commit	OnGetDataSetAdapter
	Create	OnGetDataSets
	GetDataSetAdapter	OnRollback
	GetDataSets	OnStartTransaction
	HandleRequest	

TEWBDatabaseAdapter.InTransaction Property

```
property InTransaction: Boolean
```

Specifies whether the database adapter is currently processing a transaction request. This is useful when you wish to use different datasets, or dataset adapters, with transactions than with requests for columns, rows, or BLOB values. You can read this property in an OnGetDataSetAdapter event handler in order to perform this type of conditional processing.

TEWBDatabaseAdapter.Password Property

```
property Password: String
```

Specifies the authenticated password included with a server request, if any was included. Authentication information can be passed to a web server module from an Elevate Web Builder application via HTTP headers or via HTTP URL parameters.

Note
This property will be blank until the HandleRequest method is called, and the server request has been successfully authenticated.

TEWBDatabaseAdapter.UserName Property

```
property UserName: String
```

Specifies the authenticated user name included with a server request, if any was included. Authentication information can be passed to a web server module from an Elevate Web Builder application via HTTP headers or via HTTP URL parameters.

Note
This property will be blank until the HandleRequest method is called, and the server request has been successfully authenticated.

TEWBDatabaseAdapter.AuthenticateUser Method

```
function AuthenticateUser(Request: TEWBServerRequest): Boolean
```

Use this method to trigger the OnAuthenticateUser event and authenticate the user name/password information provided with the Request parameter. Authentication information can be passed to a web server module from an Elevate Web Builder application via HTTP headers or via HTTP URL parameters, and this method will automatically handle the retrieving the authentication information from the request and passing it to the OnAuthenticateUser event handler.

Note
This method will only be automatically called when the HandleRequest method is called. In any other situation, you will need to manually call this method to authenticate a request.

TEWBDatabaseAdapter.BuildDataSets Method

```
function BuildDataSets: String
```

Use this method to build a JSON string containing all of the datasets for the database.

Note
This method will be automatically called when the HandleRequest method is called, so you normally will not need to call this method directly.

TEWBDatabaseAdapter.Commit Method

```
procedure Commit(const Operations: String)

procedure Commit(const Operations: AnsiString)
```

Use this method to commit a transaction received from an Elevate Web Builder client application into a database.

This method will automatically call the GetDataSetAdapter method as necessary to get access to the dataset adapters required to perform the insert, update, and delete row operations in the transaction. If a dataset adapter cannot be accessed for a particular dataset, then an exception will be raised and the transaction commit will fail.

Please see the **JSON Reference** section in the Elevate Web Builder manual that is included with the Elevate Web Builder IDE for more information on the structure of the JSON used with database transactions.

Note
This method will only be automatically called when the HandleRequest method is called. In any other situation, you will need to manually call this method to commit transactions.

TEWBDatabaseAdapter.Create Method

```
constructor Create(AOwner: TComponent)
```

Call the constructor to create an instance of the TEWBDatabaseAdapter component.

TEWBDatabaseAdapter.GetDataSetAdapter Method

```
function GetDataSetAdapter(const DataSetName: String):
      TEWBDataSetAdapter
```

Use this method to trigger the OnGetDataSetAdapter event and retrieve the dataset adapter instance for the specified dataset name.

Note
This method will only be automatically called when the HandleRequest method is called. In any other situation, you will need to manually call this method to retrieve a dataset adapter for a dataset.

TEWBDatabaseAdapter.GetDataSets Method

```
procedure GetDataSets(DataSets: TStrings)
```

Use this method to trigger the OnGetDataSets event and retrieve the available dataset names for the module.

Note
This method will only be automatically called when the HandleRequest method is called. In any other situation, you will need to manually call this method to retrieve a list of available dataset names.

TEWBDatabaseAdapter.HandleRequest Method

```
procedure HandleRequest(Request: TEWBServerRequest)
```

Use this method to automatically handle a database request from an Elevate Web Builder client application. Database requests use a standard request format in Elevate Web Builder. Unless you're using a different request format, you can use this method without having to worry about the details of the database request URLs, parameters, and authentication.

This method will automatically call the AuthenticateUser method first before attempting to perform any other functionality. If the AuthenticateUser method returns True, then the database adapter will continue processing the database request, triggering the GetDataSets, GetDataSetAdapter, and Commit methods as necessary.

Please see the **JSON Reference** section in the Elevate Web Builder manual that is included with the Elevate Web Builder IDE for more information on the structure of the JSON used with databases.

TEWBDatabaseAdapter.OnAuthenticateUser Event

```
property OnAuthenticateUser: TEWBAuthenticationEvent
```

This event is triggered before the database adapter handles any dataset request and provides an opportunity for the developer to authenticate the request based upon the authentication information provided with the request.

TEWBDatabaseAdapter.OnCommit Event

```
property OnCommit: TNotifyEvent
```

This event is triggered when the database adapter needs to commit a transaction. The developer can define an event handler for this event in order to handle committing a transaction using the database-specific component for handling transactions.

TEWBDatabaseAdapter.OnGetDataSetAdapter Event

```
property OnGetDataSetAdapter: TEWBGetDataSetAdapterEvent
```

This event is triggered when the GetDataSetAdapter method is called.

Warning
If the Adapter variable parameter is not assigned a value, then an exception can occur during transaction commit operations that reference the specified dataset name.

TEWBDatabaseAdapter.OnGetDataSets Event

```
property OnGetDataSets: TEWBGetDataSetsEvent
```

This event is triggered when a list request is handled by the database adapter. Usage of this event is completely optional, but the event is useful for allowing client applications to enumerate the available datasets.

TEWBDatabaseAdapter.OnRollback Event

```
property OnRollback: TNotifyEvent
```

This event is triggered when the database adapter needs to roll back a transaction. The developer can define an event handler for this event in order to handle rolling back a transaction using the database-specific component for handling transactions.

TEWBDatabaseAdapter.OnStartTransaction Event

```
property OnStartTransaction: TNotifyEvent
```

This event is triggered when the database adapter needs to start a transaction. The developer can define an event handler for this event in order to handle starting a transaction using the database-specific component for handling transactions.

2.2 TEWBDataSetAdapter Component

Unit: ewbdatasetadapter

Inherits From TComponent

The TEWBDataSetAdapter component is used in conjunction with the TEWBDatabaseAdapter component to create and load JSON dataset columns and rows to/from generic TDataSet descendant components, as well as commit transaction operations to one or more datasets. It is used with database modules that need to deal with data from a data source that isn't supported by the Elevate Web Builder Web Server's automatic database handling or need to perform special processing/authentication for database requests.

Properties	Methods	Events
DataSet	BuildColumns	OnFilterRows
KeyCaseInsensitive	BuildLoad	OnInitialize
KeyFields	BuildRows	
LocalizeDateTimeColumns	Create	
NumKeyFields	FilterRows	
	GetLoadContentType	
	Initialize	

TEWBDataSetAdapter.DataSet Property

```
property DataSet: TDataSet
```

Specifies the TDataSet-descendant component to use for the dataset adapter.

Note
The dataset must be capable of uniquely identifying rows using a primary key or some other method in order to work properly when inserting, updating, or deleting rows during transactions.

TEWBDataSetAdapter.KeyCaseInsensitive Property

```
property KeyCaseInsensitive: Boolean
```

Specifies whether the key fields for the dataset are case-insensitive.

Note
This property can be set directly at any time, but it will automatically be cleared if the Initialize method is called.

TEWBDataSetAdapter.KeyFields Property

```
property KeyFields: String
```

Specifies the key fields for the dataset, with each key field separated by a semicolon (;).

Note
This property can be set directly at any time, but it will automatically be cleared if the Initialize method is called.

TEWBDataSetAdapter.LocalizeDateTimeColumns Property

```
property LocalizeDateTimeColumns: Boolean
```

Specifies whether the adapter will localize date/time column values in the dataset when converting them to/from strings. The default value is False.

TEWBDataSetAdapter.NumKeyFields Property

```
property NumKeyFields: Integer
```

Specifies the number of key fields for the dataset.

Note
This property can be set directly at any time, but it will automatically be cleared if the Initialize method is called.

TEWBDataSetAdapter.BuildColumns Method

```
function BuildColumns: String
```

Use this method to build a JSON string containing all of the column definitions for the dataset specified in the DataSet property.

Please see the **JSON Reference** section in the Elevate Web Builder manual that is included with the Elevate Web Builder IDE for more information on the structure of the JSON used with dataset columns.

Note
This method will be automatically called when the HandleRequest method is called, so you normally will not need to call this method directly.

TEWBDataSetAdapter.BuildLoad Method

```
function BuildLoad(const Column: String; const RowKey: String):
      AnsiString
```

Use this method to build an ANSI string containing the value for the specified BLOB field in the dataset specified in the DataSet property. The RowKey parameter is a semicolon-delimited (;) list of key values that uniquely identify the row that contains the desired BLOB field data.

Note
This method will be automatically called when the HandleRequest method is called, so you normally will not need to call this method directly.

TEWBDataSetAdapter.BuildRows Method

```
function BuildRows(Params: TStrings=nil; const User: String='';
      const Password: String=''): String
```

Use this method to build a JSON string containing all of the rows for the dataset specified in the DataSet property.

The Params parameter is used to include any additional dataset filtering parameters with any BLOB load URLs that are included with the JSON row data built by this method. This ensures that any BLOB loads are executed using the same filtering parameters that were used to build the rows.

The User and Password parameters are used to include user and password authentication parameters with any BLOB load URLs that are included with the JSON row data built by this method. This ensures that any BLOB loads are authenticated using the same authentication information that was used to build the rows.

Please see the **JSON Reference** section in the Elevate Web Builder manual that is included with the Elevate Web Builder IDE for more information on the structure of the JSON used with dataset rows.

Note
This method will be automatically called when the HandleRequest method is called, so you normally will not need to call this method directly.

TEWBDataSetAdapter.Create Method

```
constructor Create(AOwner: TComponent)
```

Call the constructor to create an instance of the TEWBDataSetAdapter component.

TEWBDataSetAdapter.FilterRows Method

```
procedure FilterRows(Request: TEWBServerRequest)
```

Use this method to filter a dataset based upon the parameters available in the Request parameter.

Note
This method will be automatically called when the HandleRequest method is called, so you normally will not need to call this method directly.

TEWBDataSetAdapter.GetLoadContentType Method

```
function GetLoadContentType(const Column: String; const RowKey:
      String): String
```

Use this method to retrieve the MIME type for the specified BLOB field in the dataset specified in the DataSet property. The RowKey parameter is a semicolon-delimited (;) list of key values that uniquely identify the row that contains the desired BLOB field.

This method is a convenient way of associating MIME types with BLOB fields in a dataset. When this method is called, the TEWBDataSetAdapter will look for a field in the dataset with the name of:

<BLOB Field Name>_ContentType

and return its value as the result. If a field does not exist with that name, then the result will be a blank string.

Note
This method will be automatically called when the HandleRequest method is called, so you normally will not need to call this method directly.

TEWBDataSetAdapter.Initialize Method

```
procedure Initialize
```

Use this method to initialize a dataset's key field information. When this method is called, the NumKeyFields, KeyFields, and KeyCaseInsensitive properties are cleared and the OnInitialize event is triggered.

TEWBDataSetAdapter.OnFilterRows Event

```
property OnFilterRows: TEWBFilterDataSetRowsEvent
```

This event is triggered when the FilterRows method is called and provides an opportunity for the developer to filter the rows in the dataset according to parameters passed in via the dataset request.

TEWBDataSetAdapter.OnInitialize Event

```
property OnInitialize: TEWBInitializeDataSetAdapterEvent
```

This event is triggered when the Initialize method is called, and gives the developer an opportunity to initialize the NumKeyFields, KeyFields, and KeyCaseInsensitive properties of the dataset adapter.

2.3 TEWBJSONReader Component

Unit: ewbhttpcommon

Inherits From TObject

The TEWBJSONReader class is used to load class instance properties from JSON strings, and can be used as a general-purpose JSON reader in your applications.

When a TEWBJSONReader instance is created, the constructor allows you to specify the date-time format to use when reading date-time properties/values.

Properties	Methods	Events
Level	BeginArray	
RootObject	BeginObject	
	Create	
	EndArray	
	EndObject	
	EndOfArray	
	EndOfObject	
	GetPropertyName	
	Initialize	
	IsArray	
	IsNull	
	IsObject	
	MoreArrayElements	
	MoreProperties	
	ReadBoolean	
	ReadDateTime	
	ReadFloat	
	ReadInteger	
	ReadString	
	SkipArrayElement	
	SkipProperty	
	SkipPropertyName	
	SkipPropertySeparator	
	SkipPropertyValue	

TEWBJSONReader.Level Property

```
property Level: Integer
```

Indicates the current nesting level for any JSON objects and/or arrays. Whenever the TEWBJSONReader class begins to read an object or array using the BeginObject or BeginArray methods, the nesting level is incremented. Whenever the EndObject or EndArray methods are called, the nesting level is decremented.

TEWBJSONReader.RootObject Property

```
property RootObject: TComponent
```

Specifies an optional root object to use with the reader. The root object can be used by the developer to track which container object instance is being loaded. The default value is nil.

TEWBJSONReader.BeginArray Method

```
procedure BeginArray
```

Use this method to begin reading an array. If the current token in the incoming JSON string is not a left bracket ([), an exception will be raised.

TEWBJSONReader.BeginObject Method

```
procedure BeginObject
```

Use this method to begin reading an object. If the current token in the incoming JSON string is not a left brace ({), an exception will be raised.

TEWBJSONReader.Create Method

```
constructor Create(ADateTimeFormat:
      TEWBJSONDateTimeFormat=dtfRaw)
```

Use this method to create a new instance of the TJSONReader class. The optional ADateTimeFormat parameter indicates whether date and time values should be handled as an ISO 8601 date and time string value, or as a raw Unix date and time integer value (the number of milliseconds since midnight, January 1, 1970).

TEWBJSONReader.EndArray Method

```
procedure EndArray
```

Use this method to end reading an array. If the current token in the incoming JSON string is not a right bracket (]), an exception will be raised.

TEWBJSONReader.EndObject Method

```
procedure EndObject
```

Use this method to end reading an object. If the current token in the incoming JSON string is not a right brace (}), an exception will be raised.

TEWBJSONReader.EndOfArray Method

```
function EndOfArray: Boolean
```

Use this method to determine if the current token in the incoming JSON string is a right bracket (]).

TEWBJSONReader.EndOfObject Method

```
function EndOfObject: Boolean
```

Use this method to determine if the current token in the incoming JSON string is a right brace (}).

TEWBJSONReader.GetPropertyName Method

```
function GetPropertyName: String
```

Use this method to read a property name, without any enclosing double-quote (") characters (if applicable).

TEWBJSONReader.Initialize Method

```
procedure Initialize(const Value: String)
```

Use this method to initialize the reader with a JSON string to read.

TEWBJSONReader.IsArray Method

```
function IsArray: Boolean
```

Use this method to determine if the current token in the incoming JSON string is a left bracket ([).

TEWBJSONReader.IsNull Method

```
function IsNull: Boolean
```

Use this method to determine if the current token in the incoming JSON string is a null literal.

TEWBJSONReader.IsObject Method

```
function IsObject: Boolean
```

Use this method to determine if the current token in the incoming JSON string is a left brace ({).

TEWBJSONReader.MoreArrayElements Method

```
function MoreArrayElements: Boolean
```

Use this method to determine if the current token in the incoming JSON string is a comma (,).

TEWBJSONReader.MoreProperties Method

```
function MoreProperties: Boolean
```

Use this method to determine if the current token in the incoming JSON string is a comma (,).

TEWBJSONReader.ReadBoolean Method

```
function ReadBoolean: Boolean
```

Use this method to read a boolean value. If the current token in the incoming JSON string is not a valid JSON boolean literal (true, false), an exception will be raised.

TEWBJSONReader.ReadDateTime Method

```
function ReadDateTime: TDateTime
```

Use this method to read a date-time value. How a date-time value is read is controlled by the TEWBJSONDateTimeFormat parameter in the TEWBJSONReader class constructor.

TEWBJSONReader.ReadFloat Method

```
function ReadFloat: Double
```

Use this method to read a float value. If the current token in the incoming JSON string is not a valid JSON float or integer literal, an exception will be raised.

TEWBJSONReader.ReadInteger Method

```
function ReadInteger: Integer
```

Use this method to read an integer value. If the current token in the incoming JSON string is not a valid JSON integer literal, an exception will be raised.

TEWBJSONReader.ReadString Method

```
function ReadString: String
```

Use this method to read a string value, without any enclosing double-quote (") characters. If the current token in the incoming JSON string is not a valid JSON string literal, an exception will be raised.

TEWBJSONReader.SkipArrayElement Method

```
procedure SkipArrayElement
```

Use this method to skip over a JSON array element.

TEWBJSONReader.SkipProperty Method

```
procedure SkipProperty
```

Use this method to skip over a JSON object property.

TEWBJSONReader.SkipPropertyName Method

```
procedure SkipPropertyName
```

Use this method to skip over a JSON object property name.

TEWBJSONReader.SkipPropertySeparator Method

```
procedure SkipPropertySeparator
```

Use this method to skip over a JSON object property separator (:).

TEWBJSONReader.SkipPropertyValue Method

```
procedure SkipPropertyValue
```

Use this method to skip over a JSON object property value.

2.4 TEWBJSONWriter Component

Unit: ewbhttpcommon

Inherits From TObject

The TEWBJSONWriter class is used to save class instance properties to JSON strings, and can be used as a general-purpose JSON writer in your applications.

When a TEWBJSONWriter instance is created, the constructor allows you to specify:

- The date-time format to use when writing date-time properties.
- The number of spaces to use per indentation level, if the output is not compressed.
- Whether to include line feeds in the JSON output, if the output is not compressed.
- Whether to compress all whitespace in the JSON output. Compressing the whitespace removes any unnecessary whitespace in order to keep the size of the JSON output to a minimum.

Properties	Methods	Events
Output	BeginArray	
	BeginNewLine	
	BeginObject	
	BooleanProperty	
	BooleanValue	
	CancelNewLine	
	Create	
	DateTimeProperty	
	DateTimeValue	
	DecIndent	
	EndArray	
	EndObject	
	FloatProperty	
	FloatValue	
	IncIndent	
	Initialize	
	IntegerProperty	
	IntegerValue	
	Literal	
	NewLine	
	NullProperty	
	NullValue	
	ObjectProperty	
	PropertyName	
	Separator	
	StringProperty	
	StringValue	
	Whitespace	

TEWBJSONWriter.Output Property

```
property Output: String
```

Indicates the current JSON output.

TEWBJSONWriter.BeginArray Method

```
procedure BeginArray(HasElements: Boolean)
```

Use this method to begin writing a new array.

TEWBJSONWriter.BeginNewLine Method

```
procedure BeginNewLine
```

Use this method to set a flag requesting that the next property should be written on a new line.

Note
New lines are not used if the JSON output is being compressed.

TEWBJSONWriter.BeginObject Method

```
procedure BeginObject
```

Use this method to begin writing a new object.

TEWBJSONWriter.BooleanProperty Method

```
procedure BooleanProperty(const Name: String; Value: Boolean)

procedure BooleanProperty(const Name: String; Value: Boolean;
      DefaultValue: Boolean)
```

Use this method to write a boolean property.

TEWBJSONWriter.BooleanValue Method

```
procedure BooleanValue(Value: Boolean)
```

Use this method to write a boolean value.

TEWBJSONWriter.CancelNewLine Method

```
procedure CancelNewLine
```

Use this method to cancel a new line started via the NewLine method. This is useful if a new line is started for a property, but the property cannot be written for some reason.

TEWBJSONWriter.Create Method

```
constructor Create(ADateTimeFormat:
      TEWBJSONDateTimeFormat=dtfRaw; AIndentSpaces: Integer=3;
      AIncludeLineFeeds: Boolean=True; ACompressWhitespace:
      Boolean=False)
```

Use this method to create a new instance of the TEWBJSONWriter class. The optional ADateTimeFormat parameter indicates whether date and time values should be output as an ISO 8601 date and time string value, or as a raw Unix date and time integer value (the number of milliseconds since midnight, January 1, 1970). The optional AIndentSpaces parameter indicates how many spaces to use for indentation in the output, the optional AIncludeLineFeeds parameter indicates whether to include line feeds (CRLF) in the output, and the optional ACompressWhitespace parameter indicates whether any whitespace should be compressed (removed) from the output.

TEWBJSONWriter.DateTimeProperty Method

```
procedure DateTimeProperty(const Name: String; Value: TDateTime)

procedure DateTimeProperty(const Name: String; Value: TDateTime;
      DefaultValue: TDateTime)
```

Use this method to write a date-time property. How a date-time property is written is controlled by the first TDateTimeFormat parameter in the TEWBJSONWriter class constructor.

TEWBJSONWriter.DateTimeValue Method

```
procedure DateTimeValue(Value: TDateTime)
```

Use this method to write a date-time value. How a date-time value is written is controlled by the first TDateTimeFormat parameter in the TEWBJSONWriter class constructor.

TEWBJSONWriter.DecIndent Method

```
procedure DecIndent
```

Decrement the indentation level for the output.

Note
Indentation levels are not used if the JSON output is being compressed.

TEWBJSONWriter.EndArray Method

```
procedure EndArray(HasElements: Boolean)
```

Use this method to end writing an array.

TEWBJSONWriter.EndObject Method

```
procedure EndObject
```

Use this method to end writing an object.

TEWBJSONWriter.FloatProperty Method

```
procedure FloatProperty(const Name: String; Value: Double)

procedure FloatProperty(const Name: String; Value: Double;
      DefaultValue: Double)
```

Use this method to write a float property.

TEWBJSONWriter.FloatValue Method

```
procedure FloatValue(Value: Double)
```

Use this method to write a float value.

TEWBJSONWriter.IncIndent Method

```
procedure IncIndent
```

Increment the indentation level for the JSON output.

Note
Indentation levels are not used if the JSON output is being compressed.

TEWBJSONWriter.Initialize Method

```
procedure Initialize
```

Use this method to initialize the writer so that the Output property is blank.

TEWBJSONWriter.IntegerProperty Method

```
procedure IntegerProperty(const Name: String; Value: Int64)

procedure IntegerProperty(const Name: String; Value: Int64;
      DefaultValue: Int64)
```

Use this method to write an integer property.

TEWBJSONWriter.IntegerValue Method

```
procedure IntegerValue(Value: Int64)
```

Use this method to write an integer value.

TEWBJSONWriter.Literal Method

```
procedure Literal(const Value: String)
```

Use this method to write a literal.

TEWBJSONWriter.NewLine Method

```
procedure NewLine
```

Use this method to write a new line (CRLF).

Note
New lines are not written if the JSON output is being compressed, or if the writer was created with the new line option turned off.

TEWBJSONWriter.NullProperty Method

```
procedure NullProperty(const Name: String)
```

Use this method to write a null property.

TEWBJSONWriter.NullValue Method

```
procedure NullValue
```

Use this method to write a null value.

TEWBJSONWriter.ObjectProperty Method

```
procedure ObjectProperty(const Name: String)
```

Use this method to write an object property's name using the PropertyName method.

TEWBJSONWriter.PropertyName Method

```
procedure PropertyName(const Name: String)
```

Use this method to write a property name.

TEWBJSONWriter.Separator Method

```
procedure Separator
```

Use this method to write a separator (,).

TEWBJSONWriter.StringProperty Method

```
procedure StringProperty(const Name: String; const Value:
      String)

procedure StringProperty(const Name: String; const Value: String;
       const DefaultValue: String)
```

Use this method to write a string property.

TEWBJSONWriter.StringValue Method

```
procedure StringValue(const Value: String)
```

Use this method to write a string value.

TEWBJSONWriter.Whitespace Method

```
procedure Whitespace
```

Use this method to write a space ( ).

Note
Whitespace is not written if the JSON output is being compressed.

2.5 TEWBMIMEPart Component

Unit: ewbhttpcommon

Inherits From TObject

The TEWBMimePart object is used to represent a single MIME part in multi-part MIME-encoded content included with a web server request. Multi-part MIME content is normally used with HTML form submittals that include file uploads.

Properties	Methods	Events
Content	Create	
Disposition	Save	
Encoding		
FileName		
Headers		

TEWBMIMEPart.Content Property

```
property Content: AnsiString
```

Specifies the content of the MIME part encoded using the encoding specified in the Encodingproperty.

TEWBMIMEPart.Disposition Property

```
property Disposition: String
```

Specifies the disposition for the MIME content specified in the Content property. For HTML form submittals, this property will always be 'form-data'.

TEWBMIMEPart.Encoding Property

```
property Encoding: String
```

Specifies the encoding for the MIME content specified in the Content property.

TEWBMIMEPart.FileName Property

```
property FileName: String
```

Specifies the file name for the MIME content specified in the Content property.

TEWBMIMEPart.Headers Property

```
property Headers: TStrings
```

Specifies the headers for the MIME part. Each header has the format:

```
<Header>: <Value>[;<HeaderAttribute>=<Value>]
```

TEWBMIMEPart.Create Method

```
constructor Create(Owner: TEWBMIMEParts)
```

Use this method to create a new instance of the TEWBMIMEPart class. The Owner parameter indicates the MIME parts instance that will own and manage the MIME part.

TEWBMIMEPart.Save Method

```
procedure Save(Stream: TStream)
```

Use this method to save the content specified by the Content property to a stream.

2.6 TEWBMIMEParts Component

Unit: ewbhttpcommon

Inherits From TObject

The TEWBMimeParts object is used to represent all MIME parts in multi-part MIME-encoded content included with a web server request. Multi-part MIME content is normally used with HTML form submittals that include file uploads.

Properties	Methods	Events
Count	Create	
Items		

TEWBMIMEParts.Count Property

```
property Count: Integer
```

Indicates the number of MIME parts present in the request.

Note
This number does not include form values that aren't file attachments since they are automatically moved to the TEWBServerRequest RequestContentParams property as key-value pairs in order to make access to such values equivalent to requests that aren't MIME-encoded.

TEWBMIMEParts.Items Property

```
property Items[Index: Integer]: TEWBMIMEPart
```

Provides access to the MIME parts present in the request by ordinal index.

Note
Form values that aren't file attachments are not included here since they are automatically moved to the TEWBServerRequest RequestContentParams property as key-value pairs in order to make access to such values equivalent to requests that aren't MIME-encoded.

TEWBMIMEParts.Create Method

```
constructor Create
```

Use this method to create a new instance of the TEWBMIMEParts class.

2.7 TEWBModule Component

Unit: ewbhttpmodule

Inherits From TDataModule

The TEWBModule component represents an instance of a module in the web server. The web server automatically creates a new instance of a module for every executing thread that references the module in the URL for a new request and fires the OnExecute event when the new request is handled by the thread. This means that all code in any OnExecute event handler must be completely thread-safe.

Warning
All modules in the web server run in-process, so a fatal error such as a memory overwrite due to improper threading code in a module could cause the server to fail.

Properties	Methods	Events
Password	AuthenticateUser	OnAuthenticateUser
UserName	Create	OnExecute

TEWBModule.Password Property

```
property Password: String
```

Specifies the authenticated password included with a server request, if any was included. Authentication information can be passed to a web server module from an Elevate Web Builder application via HTTP headers or via HTTP URL parameters.

Note
This property will be blank until the server request has been successfully authenticated.

TEWBModule.UserName Property

```
property UserName: String
```

Specifies the authenticed user name included with a server request, if any was included. Authentication information can be passed to a web server module from an Elevate Web Builder application via HTTP headers or via HTTP URL parameters.

Note
This property will be blank until the server request has been successfully authenticated.

TEWBModule.AuthenticateUser Method

```
function AuthenticateUser(Request: TEWBServerRequest): Boolean
```

Use this method to trigger the OnAuthenticateUser event and authenticate the user name/password information provided with the Request parameter. Authentication information can be passed to a web server module from an Elevate Web Builder application via HTTP headers or via HTTP URL parameters, and this method will automatically handle the retrieving the authentication information from the request and passing it to the OnAuthenticateUser event handler.

TEWBModule.Create Method

```
constructor Create(AOwner: TComponent)
```

Use this method to create a new instance of the TEWBModule class. The AOwner parameter indicates the component instance that will own and manage the module instance.

TEWBModule.OnAuthenticateUser Event

```
property OnAuthenticateUser: TEWBAuthenticationEvent
```

This event is triggered when the AuthenticateUser method is called and provides an opportunity for the developer to authenticate the request based upon the authentication information provided with the request.

TEWBModule.OnExecute Event

```
property OnExecute: TEWBModuleExecuteEvent
```

This event fires when a new request is received by the web server.

2.8 TEWBServerRequest Component

Unit: ewbhttpmodule

Inherits From TObject

The TEWBServerRequest object represents a web server request to a module and is passed as a parameter to the TEWBModule OnExecute event. You can use the RequestMethod property to determine what type of request is being made, the RequestURLParams property to determine the parameters passed with the URL of the request, and the RequestContentType, RequestContentLength, RequestContent, RequestContentParams, RequestMIMEParts properties to access content that is included with the request:

- When the RequestContentType property equals 'application/x-www-form-urlencoded', then the content is automatically parsed and moved into the RequestContentParams property.

- When the RequestContentType property begins with 'multipart', then the content is automatically parsed and moved into the RequestMIMEParts property.

Properties	Methods	Events
RequestChrome	ComputeHash	
RequestClientAddress	Create	
RequestContent	DateTimeToGMTStr	
RequestContentLength	DateTimeToMSecs	
RequestContentParams	GMTDateTimeToLocal	
RequestContentType	HeaderExists	
RequestCookies	LocalDateTimeToGMT	
RequestFirefox	MSecsToDateTime	
RequestHeaders	ParseHeader	
RequestHost	ParseHeaderAttribute	
RequestIE	SendContent	
RequestMethod	SendContentHeader	
RequestMethodName	SendContentStream	
RequestMIMEParts	SendCustomContent	
RequestOpera	SendCustomContentHeader	
RequestParams	SendCustomContentStream	
RequestPassword	SendError	
RequestSafari	SendRedirect	
RequestSecure		
RequestURL		
RequestURLParams		
RequestUser		
RequestVersion		
ResponseCookies		
ResponseHeaders		
ResponseSessionCookies		

TEWBServerRequest.RequestChrome Property

```
property RequestChrome: Boolean
```

Indicates whether the client browser is Google Chrome.

TEWBServerRequest.RequestClientAddress Property

```
property RequestClientAddress: String
```

Indicates the client's IP address.

TEWBServerRequest.RequestContent Property

```
property RequestContent: AnsiString
```

Specifies any content sent with the request that is not 'multipart' or 'application/x-www-form-urlencoded' content, as indicated by the RequestContentType property.

TEWBServerRequest.RequestContentLength Property

```
property RequestContentLength: Integer
```

Specifies the length (in bytes) of the content in the RequestContent property.

Note
For 'multipart' or 'application/x-www-form-urlencoded' content, as indicated by the RequestContentType property, this property may not accurately reflect the length of the RequestContent property since that type of content is automatically moved to other properties by the web server.

TEWBServerRequest.RequestContentParams Property

```
property RequestContentParams: TStrings
```

For 'multipart' or 'application/x-www-form-urlencoded' content, as indicated by the RequestContentType property, this property will contain any form values submitted from an HTML form. The values are specified as key-value pairs in the form:

```
<Key>=<Value>
```

Note
For file uploads, the form value will simply be the form value name along with the name of the file being uploaded as the value. In order to access the actual uploaded file content, use the RequestMIMEParts property.

TEWBServerRequest.RequestContentType Property

```
property RequestContentType: String
```

Indicates the type and encoding of the content in the RequestContent property.

TEWBServerRequest.RequestCookies Property

```
property RequestCookies: TStrings
```

Specifies any cookies that are sent with the request by the client browser. The cookies are specified in key-value pairs in the form:

```
<Key>=<Value>
```

TEWBServerRequest.RequestFirefox Property

```
property RequestFirefox: Boolean
```

Indicates whether the client browser is Mozilla Firefox.

TEWBServerRequest.RequestHeaders Property

```
property RequestHeaders: TStrings
```

Specifies all of the raw HTTP headers included for the request.

TEWBServerRequest.RequestHost Property

```
property RequestHost: String
```

Indicates the value of the 'Host' header in the RequestHeaders property.

TEWBServerRequest.RequestIE Property

```
property RequestIE: Boolean
```

Indicates whether the client browser is Microsoft Internet Explorer.

TEWBServerRequest.RequestMethod Property

```
property RequestMethod: TEWBRequestMethod
```

Indicates the type of request.

TEWBServerRequest.RequestMethodName Property

```
property RequestMethodName: String
```

Indicates the type of request in its raw form.

TEWBServerRequest.RequestMIMEParts Property

```
property RequestMIMEParts: TEWBMIMEParts
```

For 'multipart' content, as indicated by the RequestContentType property, this property will contain any MIME content that isn't a textual form value.

TEWBServerRequest.RequestOpera Property

```
property RequestOpera: Boolean
```

Indicates whether the client browser is the Opera web browser.

TEWBServerRequest.RequestParams Property

```
property RequestParams: TStrings
```

Indicates the URL parameters for the request as key-value pairs in the form:

```
<Key>=<Value>
```

TEWBServerRequest.RequestPassword Property

```
property RequestPassword: String
```

Indicates the value of the 'X-EWB-Password' header in the RequestHeaders property. This header is normally sent by Elevate Web Builder applications for dataset requests, but can be used for other types of requests as well.

TEWBServerRequest.RequestSafari Property

```
property RequestSafari: Boolean
```

Indicates whether the client browser is Apple Safari.

TEWBServerRequest.RequestSecure Property

```
property RequestSecure: Boolean
```

Indicates whether the request is secure (HTTPS) or not.

Note
The web server does not currently support secure requests.

TEWBServerRequest.RequestURL Property

```
property RequestURL: String
```

Indicates the URL of the request without any URL parameters. The URL parameters can be retrieved via the RequestURLParams (raw) or RequestParams (key-value pairs) properties.

TEWBServerRequest.RequestURLParams Property

```
property RequestURLParams: String
```

Indicates the URL parameters for the request in their raw form. Use the RequestParams property to access the URL parameters as key-value pairs.

TEWBServerRequest.RequestUser Property

```
property RequestUser: String
```

Indicates the value of the 'X-EWB-User' header in the RequestHeaders property. This header is normally sent by Elevate Web Builder applications for dataset requests, but can be used for other types of requests as well.

TEWBServerRequest.RequestVersion Property

```
property RequestVersion: String
```

Indicates the HTTP version in use by the client browser, and is usually 'HTTP/1.1'.

TEWBServerRequest.ResponseCookies Property

```
property ResponseCookies: TStrings
```

Use this property to specify any persistent cookies that you want to add/modify in the client browser.

TEWBServerRequest.ResponseHeaders Property

```
property ResponseHeaders: TStrings
```

Use this property to specify any special response headers that you want to send to the client browser. By default, the following methods automatically send the indicated response headers:

Method	Headers Sent
SendContent	'Date' 'From' 'Server' 'Connection' 'Content-Type' (text/html; charset=utf-8) 'Content-Length'
SendCustomContent	'Date' 'From' 'Server' 'Connection' 'Content-Type' (specified in method) 'Content-Length' 'Content-Disposition' (specified in method)
SendRedirect	'Date' 'From' 'Server' 'Connection' 'Content-Type' (text/html; charset=utf-8) 'Content-Length' 'Location'
SendError	'Date' 'From' 'Server' 'Connection' 'Content-Type' (text/html; charset=utf-8) 'Content-Length'

In addition, any cookies specified in the ResponseCookies or ResponseSessionCookies are automatically sent using a 'Set-Cookie' header.

TEWBServerRequest.ResponseSessionCookies Property

```
property ResponseSessionCookies: TStrings
```

Use this property to specify any session-only cookies that you want to add/modify in the client browser. Session-only cookies persist only until the client browser session terminates, and then are cleared by the browser.

TEWBServerRequest.ComputeHash Method

```
function ComputeHash(AHashType: TEWBHashType; const Value:
      AnsiString): String
```

Use this method to compute a hash for a given input string. The AHashType parameter specifies the type of hash to compute. The output is a hex-encoded string (2 characters per byte) that represents the computed hash.

Warning
Do not use MD5 hashes for security purposes such as password hashes. Also, SHA-1 hashes are currently secure, but may be exploitable in the near future, so the stronger SHA-256 and SHA-512 hashes are recommended for stored passwords.

TEWBServerRequest.Create Method

```
constructor Create
```

Use this method to create a new instance of the TEWBServerRequest class.

TEWBServerRequest.DateTimeToGMTStr Method

```
function DateTimeToGMTStr(Value: TDateTime): String
```

Converts a TDateTime value into a GMT date/time string that adheres to RFC1123 and has the format:

```
dow, day month year hours:mins:secs GMT
```

TEWBServerRequest.DateTimeToMSecs Method

```
function DateTimeToMSecs(Value: TDateTime): Int64
```

Converts a TDateTime value into the number of milliseconds since midnight, January 1, 1970.

Note
The result is the native date/time representation used by JavaScript and Elevate Web Builder client applications.

TEWBServerRequest.GMTDateTimeToLocal Method

```
function GMTDateTimeToLocal(Value: TDateTime): TDateTime
```

Converts a GMT (UTC) TDateTime value into its local equivalent, optionally accounting for Daylight Savings Time.

TEWBServerRequest.HeaderExists Method

```
function HeaderExists(Headers: TStrings; const Header: String):
      Boolean
```

Use this method to determine if a specific HTTP header exists in the passed list of headers.

HTTP headers use the following format:

```
<Header Name>: <Header Value> [; <Attribute=<Attribute Value>...]
```

TEWBServerRequest.LocalDateTimeToGMT Method

```
function LocalDateTimeToGMT(Value: TDateTime): TDateTime
```

Converts a local TDateTime value into its GMT (UTC) equivalent, optionally accounting for Daylight Savings Time.

TEWBServerRequest.MSecsToDateTime Method

```
function MSecsToDateTime(Value: Int64): TDateTime
```

Converts the number of milliseconds since midnight, January 1, 1970 into a TDateTime value.

Note
The input is the native date/time representation used by JavaScript and Elevate Web Builder client applications.

TEWBServerRequest.ParseHeader Method

```
function ParseHeader(Headers: TStrings; const Header: String):
      String
```

Use this method to parse a specific HTTP header value from the passed list of headers.

HTTP headers use the following format:

```
<Header Name>: <Header Value> [; <Attribute=<Attribute Value>...]
```

TEWBServerRequest.ParseHeaderAttribute Method

```
function ParseHeaderAttribute(const Attribute: String; const
      Header: String): String
```

Use this method to parse a specific HTTP header attribute from the passed header value.

HTTP headers use the following format:

```
<Header Name>: <Header Value> [; <Attribute=<Attribute Value>...]
```

TEWBServerRequest.SendContent Method

```
procedure SendContent(const Content: String; StatusCode:
      Integer=HTTP_OK; const StatusMessage: String='')
```

Sends a response to the client browser using the passed content, HTTP status code, and status message. The Unicode content is automatically sent in UTF-8 format.

Please see the following link for a complete list of HTTP status/error codes:

Status Code and Reason Phrase

Note
Only one response can be sent per request. Attempting to send more than one response will cause errors in the client browser.

TEWBServerRequest.SendContentHeader Method

```
procedure SendContentHeader(StatusCode: Integer=HTTP_OK; const
      StatusMessage: String='')
```

Sends response headers to the client browser using the passed HTTP status code and status message. You should use this method when responding to HEAD requests to a module, which can occur when the content type of the resource being handled by the module is unknown to the client application.

Please see the following link for a complete list of HTTP status/error codes:

Status Code and Reason Phrase

Note
Only one response can be sent per request. Attempting to send more than one response will cause errors in the client browser.

TEWBServerRequest.SendContentStream Method

```
procedure SendContentStream(const Content: TStream; StatusCode:
      Integer=HTTP_OK; const StatusMessage: String='')
```

Sends a response to the client browser using the passed content stream, HTTP status code, and status message.

The content stream is sent to the client as-is (binary).

Please see the following link for a complete list of HTTP status/error codes:

Status Code and Reason Phrase

Note
Only one response can be sent per request. Attempting to send more than one response will cause errors in the client browser.

TEWBServerRequest.SendCustomContent Method

```
procedure SendCustomContent(const Content: String; const
      ContentType: String; const ContentDisposition: String)

procedure SendCustomContent(const Content: AnsiString; const
      ContentType: String; const ContentDisposition: String)
```

Sends a response to the client browser using the passed content, type, and disposition.

For the AnsiString version of this method, the content stream is sent to the client as-is (binary) and the content length is set based upon the length of the Content parameter.

For the Unicode string version of this method, the content length is set based upon the UTF-8 encoded version of the Content parameter, so be sure to set the UTF-8 encoding (; charset=utf-8) as part of the ContentType parameter with this version of the method.

Note
Only one response can be sent per request. Attempting to send more than one response will cause errors in the client browser.

TEWBServerRequest.SendCustomContentHeader Method

```
procedure SendCustomContentHeader(const ContentType: String;
      const ContentDisposition: String)
```

Sends response headers to the client browser using the passed content type and disposition. You should use this method when responding to HEAD requests to a module, which can occur when the content type of the resource being handled by the module is unknown to the client application.

Note
Only one response can be sent per request. Attempting to send more than one response will cause errors in the client browser.

TEWBServerRequest.SendCustomContentStream Method

```
procedure SendCustomContentStream(const Content: TStream; const
      ContentType: String; const ContentDisposition: String)
```

Sends a response to the client browser using the passed content stream, type, and disposition.

Note
Only one response can be sent per request. Attempting to send more than one response will cause errors in the client browser.

TEWBServerRequest.SendError Method

```
procedure SendError(ErrorCode: Integer; const ErrorMessage:
      String)
```

Sends an error response to the client browser using the passed HTTP error code and message. The Unicode message is sent as the content in UTF-8 format.

Please see the following link for a complete list of HTTP status/error codes:

Status Code and Reason Phrase

Note
Only one response can be sent per request. Attempting to send more than one response will cause errors in the client browser.

TEWBServerRequest.SendRedirect Method

```
procedure SendRedirect(const NewLocationURL: String; const
      Content: String=''; StatusCode: Integer=HTTP_FOUND; const
      StatusMessage: String='Redirecting')
```

Sends a redirect response to the client browser using the passed new location, content, HTTP status code and message. The Unicode content is automatically sent in UTF-8 format.

Please see the following link for a complete list of HTTP status/error codes:

Status Code and Reason Phrase

Note
Only one response can be sent per request. Attempting to send more than one response will cause errors in the client browser.

Chapter 3
Type Reference

3.1 TEWBAuthenticationEvent Type

Unit: ewbdatasetadapter

```
TEWBAuthenticationEvent = procedure(const RequestUserName:
      String; const RequestPassword: String; var Authenticated:
      Boolean) of object
```

This type is used for the TEWBDatabaseAdapter OnAuthenticateUser event.

The RequestUserName and RequestPassword parameters specify the user information, and the Authenticated variable parameter allows the developer to specify whether or not the authentication was successful.

Note
The default value of the Authenticated parameter is False, which means that you must authenticate the user and set the Authenticated parameter to True in order for any dataset requests to execute properly.

3.2 TEWBFilterDataSetRowsEvent Type

Unit: ewbdatasetadapter

```
TEWBFilterDataSetRowsEvent = procedure(Adapter:
      TEWBDataSetAdapter; Request: TEWBServerRequest) of object;
```

This type is used for the TEWBDataSetAdapter OnFilterRows event.

The Adapter parameter specifies the TEWBDataSetAdapter instance whose rows are being filtered, while the Request parameter specifies the TEWBServerRequest instance that is being handled by the dataset adapter.

3.3 TEWBGetDataSetAdapterEvent Type

Unit: ewbdatasetadapter

```
TEWBGetDataSetAdapterEvent = procedure(const DataSetName: String;
       var Adapter: TEWBDataSetAdapter) of object
```

This type is used for the TEWBDatabaseAdapter OnGetDataSetAdapter event.

The DataSetName parameter specifies the name of the dataset that requires an adapter, and the Adapter variable parameter should be assigned the TEWBDataSetAdapter instance that will be used to generate/consume the JSON for database requests.

3.4 TEWBGetDataSetsEvent Type

Unit: ewbdatasetadapter

```
TEWBGetDataSetsEvent = procedure(DataSets: TStrings) of object
```

This type is used for the TEWBDatabaseAdapter OnGetDataSets event.

The DataSets parameter specifies a TStrings instance to use for populating the list of available datasets.

3.5 TEWBHashType Type

Unit: ewbhttpmodule

```
TEWBHashType = (htMD5,htSHA1,htSHA256,htSHA512)
```

This type is used to represent the type of hash being computed with the TEWBServerRequest ComputeHash method.

Element	Description
htMD5	The hash will be computed as an MD5 hash.
htSHA1	The hash will be computed as an SHA-1 hash.
htSHA256	The hash will be computed as an SHA-256 hash.
htSHA512	The hash will be computed as an SHA-512 hash.

3.6 TEWBInitializeDataSetAdapterEvent Type

Unit: ewbdatasetadapter

```
TEWBInitializeDataSetAdapterEvent = procedure(Adapter:
      TEWBDataSetAdapter) of object
```

This type is used for the TEWBDataSetAdapter OnInitialize event.

The Adapter parameter specifies the TEWBDataSetAdapter instance that is being initialized.

3.7 TEWBJSONDateTimeFormat Type

Unit: ewbhttpcommon

```
TEWBJSONDateTimeFormat = (dtfRaw,dtfISO8601)
```

The TEWBJSONDateTimeFormat enumerated type is used with TEWBJSONReader and TEWBJSONWriter classes to specify how TDateTime values are handled when reading/writing to/from JSON.

Element	Description
dtfISO8601	Date-time values are handled as ISO-8601 date-time string values.
dtfRaw	Date-time values are handled as raw numeric Elevate Web Builder DateTime values (the default).

3.8 TEWBModuleExecuteEvent Type

Unit: ewbhttpmodule

```
TEWBModuleExecuteEvent = procedure (Request: TEWBServerRequest)
      of object
```

This type is used for the TEWBModule OnExecute event.

The Request parameter specifies the TEWBServerRequest instance that is being handled by the module.

3.9 TEWBRequestMethod Type

Unit: ewbhttpmodule

```
TEWBRequestMethod = (rmUnknown,rmGet,rmPost,rmHead,rmPut,
     rmDelete)
```

This type is used to represent the type of request in the TEWBServerRequest RequestMethod property.

Element	Description
rmDelete	The request is an HTTP DELETE request (not supported by the web server for modules at this time).
rmGet	The request is an HTTP GET request.
rmHead	The request is an HTTP HEAD request (not supported by the web server for modules at this time).
rmPost	The request is an HTTP POST request.
rmPut	The request is an HTTP PUT request (not supported by the web server for modules at this time).
rmUnknown	The request is an unknown HTTP request (not used by the web server for modules - an error will occur before the module is called if the request type is unknown).