# Zend Manual

## No code!

1. **1af4bc5f-73a3-4930-91a7-1d5f552f06a4**  
   The optional second argument is an array of values to bind to parameter placeholders in the SQL string
   1. same statement multiple times. Use the execute() method of the statement object. The single argument is an array of value to bind to parameter placeholders in the statement
   2. The first argument is a string containing a SQL expression. The optional second argument is a value that is used to replace a positional parameter placeholder in the SQL expression.
   3. first argument, and a string containing an SQL statement as the second argument. The statement is prepared, but not executed. Using a SQL statement constructor Executing a Statement
   4. The first argument is a string containing an SQL statement. The optional second argument is an array of values to bind to parameter placeholders in the SQL string.
2. The rest of this chapter will assume that you are familiar the authentication schemes available and how to create an appropriate authenticated connection. For more information, please see section the Authentication section of this manual or the Authentication Overview in the Google Data API Developer&#39;s Guide.
   1. The rest of this chapter will assume that you are familiar the authentication schemes available and how to create an appropriate authenticated connection. For more information, please see section the Authentication section of this manual or the Authentication Overview in the Google Data API Developer&#39;s Guide.
   2. Как 2.1
   3. The rest of this chapter will assume that you are familiar the authentication schemes available and how to create an appropriate authenticated connection. For more information, please see section. or the Authentication Overview in the Google Data API Developer&#39;s Guide.
3. Returns true if and only if $value …
   1. Returns true if and only if $value is between the minimum and maximum boundary values.
   2. true, if $part of $date having localе
   3. returns true if and only if the
   4. true, if $part of this object&#39
   5. Returns true if and only if $value is a valid IP address.
   6. Returns true if and only if $value is a floating-point value.
   7. Returns true if and only if $value contains only hexadecimal digit characters.
   8. returns true if and only if the result represents a successful authentication attempt
   9. Returns true if and only if $value contains only alphabetic characters.
   10. data in an array of values.
   11. Thе true if all checks are ok. It
   12. Returns true if and only if $value matches against a regular expression pattern.
   13. Returns true if and only if $value is a valid integer.
   14. Returns true if and only if $value contains only alphabetic and digit characters.
   15. Returns true if and only if $value contains only hexadecimal digit characters.
   16. …
4. **5adee098-648a-461e-a0d8-b93a41585036**  
   The JSON context sets the &#39;Content-Type&#39; response header to &#39;application/json&#39;, and the view script suffix to &#39;json.phtml&#39;.
   1. The XML context sets the &#39;Content-Type&#39; response header to &#39;text/xml&#39;, and the view script suffix to &#39;xml.phtml&#39;.
   2. The JSON context sets the &#39;Content-Type&#39; response header to &#39;application/json&#39;, and the view script suffix to &#39;json.phtml&#39;.
   3. currently enabled. Sets the &#39;Content-Type&#39; response header to &#39;application/json&#39;. By default, immediately returns the response, without waiting for the action
   4. Set the &#39;Content-Type&#39; response header to &#39;text/xml&#39;. Change the view suffix to &#39;xml.phtml&#39;
5. **ef6e1b60-96fb-4ccd-890f-cc9789fdb4a8**  
   Below is a table that lists all available Zend Framework Components and which PHP extension they need. This can help guide you to know which extensions are required for your application. Not all extensions used by Zend Framework are required for every application.
   1. Below is a table that lists all available Zend Framework Components and which PHP extension they need. This can help guide you to know which extensions are required for your application. Not all extensions used by Zend Framework are required for every application.
   2. Below is a table that lists all extensions typically found in PHP, and how they are used in Zend Framework. This can help guide you to know which extensions are required for your application. Not all extensions used by Zend Framework are required for every application.
6. **165eb643-6643-449f-bc2d-017f799946d6**  
   Control statements based on the if and elseif constructs must have a single space before the opening parenthesis of the conditional, and a single space after the closing parenthesis.
   1. Control statements based on the if and elseif constructs must have a single space before the opening parenthesis of the conditional, and a single space after the closing parenthesis.
   2. Control statements written with the &quot;switch&quot; construct must have a single space before the opening parenthesis of the conditional statement, and also a single space after the closing parenthesis.
7. **98827db0-acbe-413b-82f9-5e0bc02e1e92**  
   In this example there are configuration data for both a production system and for a staging system
   1. This example illustrates a basic use of Zend\_Config\_Xml for loading configuration data from an XML file. In this example there are configuration data for both a production system and for a staging system. Because the staging system configuration data are very similar to those for production, the staging section inherits from the production section. In this case, the decision is arbitrary and could have been written conversely, with the production section inheriting from the staging section, though this may not be the case for more complex situations. Suppose, then, that the following configuration data are contained in /path/to/config.xml :
   2. This example illustrates a basic use of Zend\_Config\_Ini for loading configuration data from an INI file. In this example there are configuration data for both a production system and for a staging system. Because the staging system configuration data are very similar to those for production, the staging section inherits from the production section. In this case, the decision is arbitrary and could have been written conversely, with the production section inheriting from the staging section, though this may not be the case for more complex situations. Suppose, then, that the following configuration data are contained in /path/to/config.ini :
8. **151ec719-8811-42a3-80bd-6cbff560c560**  
   Placing scripts at the end of your document is a good practice for speeding up delivery of your page
   1. Use InlineScript for HTML Body Scripts HeadScript &#39;s sibling helper, InlineScript , should be used when you wish to include scripts inline in the HTML body . Placing scripts at the end of your document is a good practice for speeding up delivery of your page, particularly when using 3rd party analytics scripts.
   2. Use InlineScript for HTML Body Scripts InlineScript , should be used when you wish to include scripts inline in the HTML body . Placing scripts at the end of your document is a good practice for speeding up delivery of your page, particularly when using 3rd party analytics scripts.
9. **52ad9b94-4fa6-4bfe-b178-e0fd7da934da**  
   Zend\_Date::DAY\_SHORT is extracted from $date if the $date operand is an instance of Zend\_Date or a numeric string. Otherwise, an attempt is made to extract the day according to the conventions documented for these constants: Zend\_Date::WEEKDAY\_NARROW , Zend\_Date::WEEKDAY\_NAME , Zend\_Date::WEEKDAY\_SHORT , Zend\_Date::WEEKDAY (Gregorian calendar assumed)
   1. Zend\_Date::DAY\_SHORT is extracted from $date if the $date operand is an instance of Zend\_Date or a numeric string. Otherwise, an attempt is made to extract the day according to the conventions documented for these constants: Zend\_Date::WEEKDAY\_NARROW , Zend\_Date::WEEKDAY\_NAME , Zend\_Date::WEEKDAY\_SHORT , Zend\_Date::WEEKDAY (Gregorian calendar assumed)
   2. Zend\_Date::WEEK is extracted from $date if the $date operand is an instance of Zend\_Date or a numeric string. Otherwise an exception is raised. (Gregorian calendar assumed)
   3. Zend\_Date::WEEKDAY\_DIGIT is extracted from $date , if the $date operand is an instance of Zend\_Date or a numeric string. Otherwise, an attempt is made to extract the day according to the conventions documented for these constants: Zend\_Date::WEEKDAY\_NARROW , Zend\_Date::WEEKDAY\_NAME , Zend\_Date::WEEKDAY\_SHORT , Zend\_Date::WEEKDAY (Gregorian calendar assumed)
10. **5b46dfa3-be8d-4023-a833-6b32eef058a2**  
    An exception will be thrown if Zend\_Session is not marked as readable e g before Zend\_Session has been started
    1. An exception will be thrown if the namespace is not writable (e.g., after destroy() ). namespaceGet($namespace) DEPRECATED: Use getIterator() in Zend\_Session\_Namespace . This method returns an array of the contents of $namespace . If you have logical reasons to keep this method publicly accessible, please provide feedback to the fw-auth@lists.zend.com mail list. Actually, all participation on any relevant topic is welcome :)
    2. An exception will be thrown if Zend\_Session is not marked as readable (e.g., before namespaceUnset($namespace) Use Zend\_Session::namespaceUnset($namespace) to efficiently remove an entire namespace and its contents. As with all arrays in PHP, if a variable containing an array is unset, and the array contains other objects, those objects will remain available, if they were also stored by reference in other array/objects that remain accessible via other variables. So namespaceUnset() does not perform a &quot;deep&quot; unsetting/deleting of the contents of the entries in the namespace. For a more detailed explanation, please see References Explained in the PHP manual.
    3. An exception will be thrown if Zend\_Session is not marked as readable (e.g., before Zend\_Session has been started). getIterator() Use getIterator() to obtain an array containing the names of all namespaces.
    4. An exception will be thrown if Zend\_Session is not marked as readable (e.g., before Zend\_Session has been started).
11. **de70b989-9e86-48cd-995a-bcb859e63449**  
    The Google Calendar API, like all GData APIs, is based off of the Atom Publishing Protoco (APP), an XML based format for managing web-based resources. Traffic between a client and the Google Calendar servers occurs over HTTP and allows for both authenticated and unauthenticated connections.
    1. The Google Calendar API, like all GData APIs, is based off of the Atom Publishing Protoco (APP), an XML based format for managing web-based resources. Traffic between a client and the Google Calendar servers occurs over HTTP and allows for both authenticated and unauthenticated connections.
    2. The Google Base API, like all GData APIs, is based off of the Atom Publishing Protocol (APP), an XML based format for managing web-based resources. Traffic between a client and the Google Base servers occurs over HTTP and allows for both authenticated and unauthenticated connections.
    3. The Picasa Web Albums API, like all GData APIs, is based off of the Atom Publishing Protocol (APP), an XML based format for managing web-based resources. Traffic between a client and the servers occurs over HTTP and allows for both authenticated and unauthenticated connections.
12. **9208c047-8361-4b54-8180-4f6e2eb2cbfe**
    1. preDispatch() is called before an action is dispatched by the dispatcher . This callback allows for proxy or filter behavior. By altering the request and resetting its dispatched flag (via Zend\_Controller\_Request\_Abstract::setDispatched(false) ), the current action may be skipped and/or replaced.
    2. postDispatch() is called after an action is dispatched by the dispatcher . This callback allows for proxy or filter behavior. By altering the request and resetting its dispatched flag (via Zend\_Controller\_Request\_Abstract::setDispatched(false) ), a new action may be specified for dispatching.
13. **ccc6343d-9532-4c33-a3fb-c1d6fc0375fa**
    1. Single structured source Positive: All source files for every languages can be found in one directory. No splitting of related files.
    2. Language structured source Positive: Every language is based in one directory. Easy translation as only one directory has to be translated by a language team. Also the usage of multiple files is transparent.
    3. Application structured source Positive: All source files for every languages can be found in one directory. No splitting of related files. Negative: Having multiple files for the same language is problematic.
    4. Gettext structured source Positive: Old gettext sources can be used without changing structure. Negative: Having sub-sub directories may be confusing for people who have not used gettext before.
    5. File structured source Positive: Every file is related to its own translation source. Negative: Multiple small translation source files make it harder to translate. Also every file has to be added as translation source.
14. **aad8b94d-e749-46d2-81d5-fb0784472eb3**  
    This example shows getting a Row object from the table Accounts
    1. This example shows getting a Row object from the table Accounts , and finding the Bugs reported by that account.
    2. This example shows geting a Row object from from the origin table Bugs , and finding rows from the destination table Products , representing products related to that bug.
    3. This example shows getting a Row object from the table Bugs , and finding the account for the engineer assigned to fix that bug. The rule key string that corresponds to this reference relationship in this example is &#39;Engineer&#39; .
    4. This example shows getting a Row object from the table Accounts , and finding the Bugs assigned to be fixed by the user of that account. The rule key string that corresponds to this reference relationship in this example is &#39;Engineer&#39; .
    5. This example shows geting a Row object from from the origin table Bugs , and finding rows from the destination table Products , representing products related to that bug.
    6. This example shows getting a Row object from the table Bugs (for example one of those bugs with status &#39;NEW&#39;), and finding the row in the Accounts table for the user who reported the bug.
15. **230684d3-00ac-4429-86af-cb7ffe14180a**  
    does not support sequences so lastInsertId ignores its arguments and always returns the last value generated for an auto increment key The lastSequenceId method returns null
    1. Microsoft SQL Server does not support sequences, so lastInsertId() ignores its arguments and always returns the last value generated for an auto-increment key. The lastSequenceId() method returns null .
    2. SQLite does not support sequences, so lastInsertId() ignores its arguments and always returns the last value generated for an auto-increment key. The lastSequenceId() method returns null .
    3. MySQL does not support sequences, so lastInsertId() ignores its arguments and always returns the last value generated for an auto-increment key. The lastSequenceId() method returns null .
    4. MySQL does not support sequences, so lastInsertId() ignores its arguments and always returns the last value generated for an auto-increment key. The lastSequenceId() method returns null .
16. 41e8c6f9-ab2e-42b4-bba7-8128c4938b3f
    1. artistGetRelatedArtists() : Returns a SimpleXML object containing a list of Artists similar to the current Artist.
    2. artistGetTopFans() : Returns a SimpleXML object containing a list of Users who listen most to the current Artist.
    3. artistGetTopTracks() : Returns a SimpleXML object containing a list of the current Artist&#39;s top-rated Tracks.
    4. artistGetTopAlbums() : Returns a SimpleXML object containing a list of the current Artist&#39;s top-rated Albums.
    5. artistGetTopTags() : Returns a SimpleXML object containing a list of the Tags most frequently applied to current Artist.
    6. userGetRecentWeeklyTrackChart() : Returns a SimpleXML object containing the most recent Weekly Track Chart for the current user.
    7. userGetRecentWeeklyAlbumChart() : Returns a SimpleXML object containing the most recent Weekly Album Chart for the current user.
    8. userGetRecentWeeklyArtistChart() : Returns a SimpleXML object containing the most recent Weekly Artist Chart for the current user.
    9. userGetWeeklyChartList() : Returns a SimpleXML object containing a list of weeks for which there exist Weekly Charts for the current user.
    10. userGetRecentJournals() : Returns a SimpleXML object containing a list of the current user&#39;s most recent journal entries.
    11. userGetRecentLovedTracks() : Returns a SimpleXML object containing a list of the 10 tracks most recently loved by the current user.
    12. userGetRecentBannedTracks() : Returns a SimpleXML object containing a list of the 10 tracks most recently banned by the current user.
    13. userGetRecentTracks() : Returns a SimpleXML object containing the 10 tracks most recently played by the current user.
    14. userGetNeighbours() : Returns a SimpleXML object containing the user names of people with similar listening habits to the current user.
    15. userGetFriends() : Returns a SimpleXML object containing the user names of the current user&#39;s friends.
    16. userGetTopTagsForTrack() : Requires that a track be set via setTrack() . Returns a SimpleXML object containing the tags most applied to the current track by the current user.
    17. userGetTopTagsForAlbum() : Requires that an album be set via setAlbum() . Returns a SimpleXML object containing the tags most applied to the current album by the current user.
    18. userGetTopTagsForArtist() : Requires that an artist be set via setArtist() . Returns a SimpleXML object containing the tags most applied to the current artist by the current user.
    19. userGetTopTags() : Returns a SimpleXML object containing a list of tags most applied by the current user.
    20. userGetTopTracks() : Returns a SimpleXML object containing a list of the current user&#39;s most listened to tracks.
    21. userGetTopAlbums() : Returns a SimpleXML object containing a list of the current user&#39;s most listened to albums.
    22. userGetTopArtists() : Returns a SimpleXML object containing a list of the current user&#39;s most listened to artists.
    23. userGetProfileInformation() : Returns a SimpleXML object containing the current user&#39;s profile information.
17. 21c40b54-c2bb-4110-9b33-4d9638e137f4
    1. XLIFF is a XML file based format, which is related to TMX but simpler as it does not support all possibilities of it. XML files are human-readable, but the parsing is not as fast as with gettext files.
    2. TMX is a XML file based format, which is announced to be the next industry standard. XML files are human-readable, but the parsing is not as fast as with gettext files.
    3. QT is a XML file based format. XML files are human-readable, but the parsing is not as fast as with gettext files.
    4. TBX is a XML file based format and a completly new format. XML files are human-readable, but the parsing is not as fast as with gettext files.
18. **f459f975-efa7-4dbe-aca4-92f4e52aa092**
    1. Searching Yahoo! Site Explorer Inbound Links is simple; just use the inlinkDataSearch() method, as in the following example. For full details, please see the Yahoo! Site Explorer Inbound Links Documentation .
    2. Searching Yahoo! Site Explorer&#39;s PageData is simple; just use the pageDataSearch() method, as in the following example. For full details, please see the Yahoo! Site Explorer PageData Documentation .
    3. Searching Yahoo! News Searching Yahoo! News is simple; just use the newsSearch() method, as in the following example. For full details, please see the Yahoo! News Search Documentation .
19. f9fb6396-00fa-49de-bcd6-cd13956953fc
    1. Am Returns a translation for &#39;AM&#39; in a expected locale. If you want to receive data for other calendars an string with the expected calendar. If you omit the value then the &#39;gregorian&#39; calendar will be used. Use Zend\_Date for simplicity
    2. DefaultDate Returns the default date format which is used within the given calendar. If you omit the value the &#39;gregorian&#39; calendar will be used. Use Zend\_Date for simplicity
    3. DefaultDay Returns the default format for days which is used within the given calendar. If you omit the value the &#39;gregorian&#39; calendar will be used. Use Zend\_Date for simplicity
    4. DefaultTime Returns the default time format which is used within the given calendar. If you omit the value the &#39;gregorian&#39; calendar will be used. Use Zend\_Date for simplicity
    5. Returns the time format for an given calendar or format within a locale If you omit the value then the 39 gregorian 39 calendar will be used with the 39 medium
    6. Date Returns the date format for an given calendar or format within a locale If you omit the value then the 39 gregorian 39 calendar will be used with the 39
    7. DefaultMonth Returns the default format for months which is used within the given calendar. If you omit the value the &#39;gregorian&#39; calendar will be used. Use Zend\_Date for simplicity
    8. Pm Returns a translation for &#39;PM&#39; in a expected locale. If you want to receive data for other calendars an string with the expected calendar. If you omit the value then the &#39;gregorian&#39; calendar will be used. Use Zend\_Date for simplicity
    9. MonthContext Returns the default context for months which is used within the given calendar. If you omit the value the &#39;gregorian&#39; calendar will be used. Use Zend\_Date for simplicity
    10. DateChars Returns a character table which contains all characters used when displaying dates DefaultCalendar Returns the default calendar for the given locale. For most locales this will be &#39;gregorian&#39;. Use Zend\_Date for simplicity
    11. how to display date with times in the same string within the given calendar If you omit the value the 39 gregorian 39 calendar will be used Use Zend\_Date for simplicity
    12. DayContext Returns the default context for Â´days which is used within the given calendar. If you omit the value the &#39;gregorian&#39; calendar will be used. Use Zend\_Date for simplicity
20. **0a97d757-97c4-4e57-938e-04dd06fb1427**  
    for which the target LDAP server is an authority
    1. accountDomainName The **FQDN domain for which** the target LDAP server is an authority (e.g., example.com).
    2. accountDomainNameShort The &#39;short&#39; domain for which the target LDAP server is an authority (e.g., FOO).
    3. accountDomainNameShort The &#39;short&#39; domain for which the target LDAP server is an authority.
    4. accountDomainName **The FQDN domain name for which** the target LDAP server is an authority (e.g., example.com ).
    5. on which the LDAP server is listening. If useSsl
21. **3f0f1eb1-028e-484b-a502-00541d40cd78**
    1. IBM DB2 supports both sequences and auto-incrementing keys. Therefore the arguments to lastInsertId() are optional. If you give no arguments, the Adapter returns the last value generated for an auto-increment key. If you give arguments, the Adapter returns the last value generated by the sequence named according to the convention &#39; table \_ column \_seq&#39;.
    2. PostgreSQL supports both sequences and auto-incrementing keys. Therefore the arguments to lastInsertId() are optional. If you give no arguments, the Adapter returns the last value generated for an auto-increment key. If you give arguments, the Adapter returns the last value generated by the sequence named according to the convention &#39; table \_ column \_seq&#39;.
22. **22b42e63-574c-4c9d-88fd-4e7ba1ad0b8c**  
    way to provide a database adapter to a Table class is by
    1. The first way to provide a database adapter to a Table class is by passing it as an object of type Zend\_Db\_Adapter\_Abstract in the options array, identified by the key &#39;db&#39; .
    2. The second way to provide a database adapter to a Table class is by declaring an object of type Zend\_Db\_Adapter\_Abstract to be a default database adapter for all subsequent instances of Tables in your application. You can do this with the static method Zend\_Db\_Table\_Abstract::setDefaultAdapter() . The argument is an object of type Zend\_Db\_Adapter\_Abstract.
    3. The third way to provide a database adapter to a Table class is by passing a string in the options array, also identified by the &#39;db&#39; key. The string is used as a key to the static Zend\_Registry instance, where the entry at that key is an object of type Zend\_Db\_Adapter\_Abstract.
    4. ways of specifying the database adapter to a Table class and these
23. **4b98cf11-f7d0-4d1b-8488-6732f5b7f557**
    1. isYesterday() isYesterday() Tests if yesterday&#39;s year, month, and day match this object&#39;s date value, using this object&#39;s timezone.isYesterday() isYesterday() Tests if yesterday&#39;s year, month, and day match this object&#39;s date value, using this object&#39;s timezone.
    2. isTomorrow() isTomorrow() Tests if tomorrow&#39;s year, month, and day match this object&#39;s date value, using this object&#39;s timezone.
    3. isToday() isToday() Tests if today&#39;s year, month, and day match this object&#39;s date value, using this object&#39;s timezone.
24. **a85c9f1e-075e-4979-b7ba-53afd930f0db**
    1. Default Decorators Do Not Need to Be Loaded By default, the default decorators are loaded during object initialization. You can disable this by passing the &#39;disableLoadDefaultDecorators&#39; option to the constructor: This option may be mixed with any other options you pass, both as array options or in a Zend\_Config object.
    2. Default Decorators Do Not Need to Be Loaded By default, the default decorators are loaded during object initialization. You can disable this by passing the &#39;disableLoadDefaultDecorators&#39; option to the constructor: This option may be mixed with any other options you pass, both as array options or in a Zend\_Config object.
    3. Default Decorators Do Not Need to Be Loaded By default, the default decorators are loaded during object initialization. You can disable this by passing the &#39;disableLoadDefaultDecorators&#39; option when creating a display group: This option may be mixed with any other options you pass, both as array options or in a Zend\_Config object.
25. **ff04542f-c241-43f9-9419-581943a0fc5c**
    1. Retrieving videos uploaded by a user You can retrieve a list of videos uploaded by a particular user using a simple helper method. This example retrieves videos uploaded by the user &#39;liz&#39;.
    2. Retrieving videos favorited by a user You can retrieve a list of a user&#39;s favorite videos using a simple helper method. This example retrieves videos favorited by the user &#39;liz&#39;.
    3. Retrieving video responses for a video You can retrieve a list of a video&#39;s video responses using a simple helper method. This example retrieves video response for a video with the ID &#39;abc123813abc&#39;.
26. **c2a80f3b-da51-407e-b964-6f52fb3c0b9e**
    1. tableName : This is the name of the database table that contains the authentication credentials, and against which the database authentication query is performed.
    2. identityColumn : This is the name of the database table column used to represent the identity. The identity column must contain unique values, such as a username or e-mail address.
    3. credentialColumn : This is the name of the database table column used to represent the credential. Under a simple identity and password authentication scheme, the credential value corresponds to the password. See also the credentialTreatment option.
27. **22ba0507-82eb-41ad-b5fe-63b226a56955**
    1. You can update rows in a database table using the update() method of an Adapter. This method takes three arguments: the first is the name of the table; the second is an associative array mapping columns to change to new values to assign to these columns. The values in the data array are treated as string literals. See for information on using SQL expressions in the data array. The third argument is a string containing an SQL expression that is used as criteria for the rows to change. The values and identifiers in this argument are not quoted or escaped. You are responsible for ensuring that any dynamic content is interpolated into this string safely.
    2. You can delete rows from a database table using the delete() method. This method takes two arguments: the first is a string naming the table. The second argument is a string containing an SQL expression that is used as criteria for the rows to delete. The values and identifiers in this argument are not quoted or escaped. You are responsible for ensuring that any dynamic content is interpolated into this string safely. See for methods to help you do this. The return value is the number of rows affected by the delete operation.
28. **19abd82e-f603-43ca-a9fa-2a40f2cffde4**
    1. If you want to receive data for other calendars, contexts or format, then you must give an array instead of an integer with the expected values.
    2. If you want to receive data for other calendars, contexts or formats, then you must give an array instead of an integer with the expected values.
    3. If you want to receive data for other calendars or formats, then you must give an array instead of the era number with the expected values.
    4. If you want to receive data for other calendars, contexts or formats, then you must give an array instead of an integer with the expected values.