AjaxDispatcher: __construct() • performAction() **AjaxResponse:** __construct($text) • setCacheDuration($duration) • setVary($vary) • setResponseCode($code) • setContentType($type) • disable() • addText($text) • printText() • sendHeaders() • checkLastModified ($timestamp) • loadFromMemcached($mckey, $touched) • storeInMemcached($mckey, $expiry) **AlphabeticPager:** getNavigationBar() • # getOrderTypeMessages()

APCBagOStuff: get($key) • set($key, $value, $exptime) • delete($key, $time) **Article:** __construct(Title $title, $oldId) • :: newFromID($id) • setRedirectedFrom($from) • getRedirectTarget() • insertRedirect() • followRedirect() • followRedirectText($text) • getTitle() • clear() • getContent() • getSection($text,$section) • getOldID() • getOldIDFromRequest() • loadContent() • pageData($dbr, $conditions) • pageDataFromTitle($dbr, $title) • pageDataFromId($dbr, $id) • loadPageData($data) • fetchContent($oldid) • forUpdate($x) • getDB() • getSelectOptions($options) • getID() • exists() • getCount() • isCountable($text) • isRedirect($text) • isCurrent() • loadLastEdit() • getTimestamp() • getUser() • getUserText() • getComment() • getMinorEdit() • getRevIdFetched() • getContributors($limit , $offset) • view() • # useParserCache($oldid) • viewRedirect($target, $appendSubtitle , $forceKnown) • addTrackbacks() • deletetrackback() • render() • purge() • doPurge() • insertOn($dbw) • updateRevisionOn(&$dbw, $revision, $lastRevision , $lastRevIsRedirect) • updateRedirectOn(&$dbw, $redirectTitle, $lastRevIsRedirect) • updateIfNewerOn(&$dbw, $revision) • replaceSection($section, $text, $summary , $edittime) • insertNewArticle($text, $summary, $isminor, $watchthis, $suppressRC, $comment, $bot) • updateArticle($text, $summary, $minor, $watchthis, $forceBot , $sectionanchor) • doEdit($text, $summary, $flags , $baseRevId , $user) • showArticle($text, $subtitle , $sectionanchor , $me2, $now, $summary, $oldid) • doRedirect($noRedir , $sectionAnchor , $extraQuery) • markpatrolled() • watch() • doWatch() • unwatch() • doUnwatch() • protect() • unprotect() • flattenRestrictions($limit) • generateReason(&$hasHistory) • delete() • isBigDeletion() • estimateRevisionCount() • getLastNAuthors($num, $revLatest) • confirmDelete($reason) • showLogExtract($out) • doDelete($reason, $suppress) • doDeleteArticle($reason, $suppress , $id) • doRollback($fromP, $summary, $token, $bot, &$resultDetails) • commitRollback($fromP, $summary, $bot, &$resultDetails) • rollback() • viewUpdates() • prepareTextForEdit($text, $revid) • editUpdates($text, $summary, $minoredit, $timestamp_of_pagechange, $newid, $changed) • createUpdates($rev) • setOldSubtitle($oldid) • preSaveTransform($text) • tryFileCache() • isFileCacheable() • checkTouched() • getTouched() • getLatest() • quickEdit($text, $comment , $minor) • incViewCount($id) • :: onArticleCreate($title) • :: onArticleDelete($title) • :: onArticleEdit($title, $transclusions) • revert() • info() • pageCountInfo($title) • getUsedTemplates() • getHiddenCategories() • :: getAutosummary($oldtext, $newtext, $flags) • outputWikiText($text, $cache) • updateCategoryCounts($added, $deleted) **AtomFeed:** formatTime($ts) • outHeader() • getFeedId() • getSelfUrl() • outItem($item) • outFooter() ?> **AuthPlugin:** userExists($username) • authenticate($username, $password) • modifyUITemplate(&$template) • setDomain($domain) • validDomain($domain) • updateUser(&$user) • autoCreate() • allowPasswordChange() • setPassword($user, $password) • updateExternalDB($user) • canCreateAccounts() • addUser($user, $password, $email, $realname) • strict() • strictUserAuth($username) • initUser(&$user, $autocreate) • getCanonicalName($username) • getUserInstance(User &$user) **AuthPluginUser:** __construct($user) • isLocked() • isHidden() **AutoLoader:** :: autoload($className) • :: loadAllExtensions() **Autopromote:** :: getAutopromoteGroups(User $user) • * :: recCheckCondition($cond, User $user) • * :: checkCondition($cond, User $user) **BagOStuff:** __construct() • set_debug($bool) • get($key) • set($key, $value, $exptime) • delete($key, $time) • lock($key, $timeout) • unlock($key) • keys() • get_multi($keys) • set_multi($hash, $exptime) • add($key, $value, $exptime) • add_multi($hash, $exptime) • delete_multi($keys, $time) • replace($key, $value, $exptime) • incr($key, $value) • decr($key, $value) • _debug($text) • :: convertExpiry($exptime)

Block: :: newFromDB($address, $user , $killExpired) • :: newFromID($id) • clear() • load($address , $user , $killExpired) • loadFromResult(ResultWrapper $res, $killExpired) • loadRange($address, $killExpired , $user) • isAddressInRange($addr, $range) • initFromRow($row) • initialiseRange() • /*static*/ enumBlocks($callback, $tag, $flags) • delete() • insert() • update() • * validateBlockParams() • doRetroactiveAutoblock() • isWhitelistedFromAutoblocks($ip) • doAutoblock($autoblockIP, $justInserted) • deleteIfExpired() • isExpired() • isValid() • updateTimestamp() • getBy() • getByName() • forUpdate($x) • fromMaster($x) • getRedactedName() • :: encodeExpiry($expiry, $db) • :: decodeExpiry($expiry, $timestampType) • :: getAutoblockExpiry($timestamp) • :: normaliseRange($range) • :: purgeExpired() • :: infinity() • :: formatExpiry($encoded_expiry) • :: parseExpiryInput($expiry_input) **CacheDependency:** abstract isExpired(); • loadDependencyValues() **Category:** * __construct() • # initialize() • :: newFromName($name) • :: newFromTitle($title) • :: newFromID($id) • :: newFromRow($row, $title) • getName() • getID() • getPageCount() • getSubcatCount() • getFileCount() • getTitle() • getMembers($limit , $offset) • * getX($key) • refreshCounts() **Categoryfinder:** __construct() • seed ($article_ids , $categories , $mode) • run () • scan_next_layer () **CategoryPage:** view() • isFileCacheable() • openShowCategory() • closeShowCategory() **CategoryViewer:** __construct($title, $from , $until) • getHTML() • clearCategoryState() • getSkin() • addSubcategoryObject($cat, $sortkey, $pageLength) • addSubcategory($title, $sortkey, $pageLength) • getSubcategorySortChar($title, $sortkey) • addImage(Title $title, $sortkey, $pageLength, $isRedirect) • addPage($title, $sortkey, $pageLength, $isRedirect) • finaliseCategoryState() • doCategoryQuery() • getCategoryTop() • getSubcategorySection() • getPagesSection() • getImageSection() • getCategoryBottom() • formatList($articles, $articles_start_char, $cutoff) • columnList($articles, $articles_start_char) • shortList($articles, $articles_start_char) • * getCountMessage($rescnt, $dbcnt, $type) **ChangesFeed:** __construct($format, $type) • getFeedObject($title, $description) • execute($feed, $rows, $limit , $hideminor , $lastmod) • saveToCache($feed, $timekey, $key) • loadFromCache($lastmod, $timekey, $key) • :: generateFeed($rows, &$feed) **ChangesList:** __construct(&$skin) • :: newFromUser(&$user) • * preCacheMessages() • # recentChangesFlags($new, $minor, $patrolled, $nothing , $bot) • beginRecentChangesList() • endRecentChangesList() • # insertMove(&$s, $rc) • # insertDateHeader(&$s, $rc_timestamp) • # insertLog(&$s, $title, $logtype) • # insertDiffHist(&$s, &$rc, $unpatrolled) • # insertArticleLink(&$s, &$rc, $unpatrolled, $watched) • # insertTimestamp(&$s, $rc) • insertUserRelatedLinks(&$s, &$rc) • # insertAction(&$s, &$rc) • # insertComment(&$s, &$rc) • :: usePatrol() • # numberofWatchingusers($count) • :: isDeleted($rc, $field) • :: userCan($rc, $field) **ChannelFeed:** outHeader() • outItem($item) • outFooter() • httpHeaders() • contentType() • outXmlHeader()

ConcatenatedGzipHistoryBlob: ConcatenatedGzipHistoryBlob() • addItem($text) • getItem($hash) • setText($text) • getText() • removeItem($hash) • compress() • uncompress() • __sleep() • __wakeup() • isHappy() **ConstantDependency:** __construct($name) • isExpired()

CreativeCommonsRdf: show() • # prologue() • # subPrologue($type, $url) • # subEpilogue($type) • # license($terms) • # term($term, $name) • # epilogue() **Credits:** :: showPage(Article $article) • :: getCredits($article, $cnt, $showIfMax) • # :: getAuthor(Article $article) • # :: getContributors(Article $article, $cnt, $showIfMax) • # :: link(User $user) • # :: userLink(User $user) • # :: othersLink(Article $article)

DBABagOStuff: __construct($handler , $dir) • encode($value, $expiry) • decode($blob) • getReader() • getWriter() • get($key) • set($key, $value, $exptime) • delete($key, $time) • add($key, $value, $exptime) • keys() **DependencyWrapper:** isExpired() • initialiseDeps() • getValue() • storeToCache($cache, $key, $expiry) **DiffHistoryBlob:** __construct() • addItem($text) • getItem($key) • setText($text) • getText() • compress() • diff($t1, $t2) • patch($base, $diff) • uncompress() • __sleep() • __wakeup() • isHappy() **DjVuImage:** __construct($filename) • isValid() • getImageSize() • dump() • * dumpForm($file, $length, $indent) • getInfo() • * readChunk($file) • * skipChunk($file, $chunkLength) • * getMultiPageInfo($file, $formLength) • * getPageInfo($file, $formLength) • retrieveMetaData() • convertDumpToXML($dump) • parseFormDjvu($line, &$xml) **DoubleRedirectJob:** :: fixRedirects($reason, $redirTitle, $destTitle) • __construct($title, $params , $id) • run() • :: getFinalDestination($title) • getUser() **DoubleReplacer:** __construct($from, $to, $index) • replace($matches) **DublinCoreRdf:** show() • # prologue() • # epilogue() **Dump7ZipOutput:** Dump7ZipOutput($file) **DumpBZip2Output:** DumpBZip2Output($file) **DumpFileOutput:** DumpFileOutput($file) • write($string) **DumpFilter:** DumpFilter(&$sink) • writeOpenStream($string) • writeCloseStream($string) • writeOpenPage($page, $string) • writeClosePage($string) • writeRevision($rev, $string) • writeLogItem($rev, $string) • pass($page) **DumpGZipOutput:** DumpGZipOutput($file) **DumpLatestFilter:** writeOpenPage($page, $string) • writeClosePage($string) • writeRevision($rev, $string) **DumpMultiWriter:** DumpMultiWriter($sinks) • writeOpenStream($string) • writeCloseStream($string) • writeOpenPage($page, $string) • writeClosePage($string) • writeRevision($rev, $string) **DumpNamespaceFilter:** DumpNamespaceFilter(&$sink, $param) • pass($page) **DumpNotalkFilter:** pass($page) **DumpOutput:** writeOpenStream($string) • writeCloseStream($string) • writeOpenPage($page, $string) • writeClosePage($string) • writeRevision($rev, $string) • writeLogItem($rev, $string) • write($string)

DumpPipeOutput: DumpPipeOutput($command, $file) **eAccelBagOStuff:** get($key) • set($key, $value, $exptime) • delete($key, $time) • lock($key, $waitTimeout) • unlock($key) **EditPage:** EditPage($article) • getContent($def_text) • # getPreloadedText($preload) • extractMetaDataFromArticle () • # wasDeletedSinceLastEdit() • submit() • edit() • # getEditPermissionErrors() • # previewOnOpen() • importFormData(&$request) • tokenOk(&$request) • # showIntro() • # showCustomIntro() • internalAttemptSave(&$result, $bot) • # userWasLastToEdit($id, $edittime) • :: matchSpamRegex($text) • initialiseForm() • setHeaders() • showEditForm($formCallback) • # showFormBeforeText() • # showTextbox1($classes) • # showTextbox2() • # displayPreviewArea($previewOutput, $isOnTop) • # showPreview($text) • doLivePreviewScript() • # showEditTools() • getLastDelete() • getPreviewText() • getTemplates() • blockedPage() • userNotLoggedInPage() • noSuchSectionPage() • spamPage($match) • mergeChangesInto(&$editText) • checkUnicodeCompliantBrowser() • pseudoParseSectionAnchor($text) • sectionAnchor($text) • :: getEditToolbar() • :: getCheckboxes(&$tabindex, $skin, $checked) • getEditButtons(&$tabindex) • livePreview() • showDiff() • safeUnicodeInput($request, $field) • safeUnicodeOutput($text) • makesafe($invalue) • unmakesafe($invalue) • noCreatePermission() • # showDeletionLog($out) • attemptSave() • getBaseRevision() **EmaillingJob:** __construct($title, $params, $id) • run() **EmailNotification:** notifyOnPageChange($editor, $title, $timestamp, $summary, $minorEdit, $oldid) • actuallyNotifyOnPageChange($editor, $title, $timestamp, $summary, $minorEdit, $oldid) • composeCommonMailtext() • compose($user) • sendMails() • sendPersonalised($watchingUser) • sendImpersonal($addresses) **EnhancedChangesList:** beginRecentChangesList() • recentChangesLine(&$baseRC, $watched) • # recentChangesBlockGroup($block) • # maybeWatchedLink($link, $watched) • # arrow($dir, $alt, $title) • # sideArrow() • # downArrow() • # spacerArrow() • # spacerIndent() • # recentChangesBlockLine($rcObj) • # recentChangesBlock() • endRecentChangesList() **EnotifNotifyJob:** __construct($title, $params, $id) • run() **ErrorPageError:** __construct($title, $msg) • report() **Exif:** __construct($file) • makeFlatExifTags() • extractTags(&$tagset) • makeFilteredData() • makeFormattedData( ) • getData() • getFilteredData() • getFormattedData() • :: version() • isByte($in) • isASCII($in) • isShort($in) • isLong($in) • isRational($in) • isUndefined($in) • isSlong($in) • isSrational($in) • validate($tag, $val) • debug($in, $fname, $action) • debugFile($fname, $io) **ExplodeIterator:** __construct($delim, $s) • rewind() • refreshCurrent() • current() • key() • next() • valid() **ExternalEdit:** __construct($article, $mode) • edit() **ExternalStore:** :: fetchFromURL($url) • :: getStoreObject($proto) • :: insert($url, $data) • :: insertToDefault($data) **ExternalStoreDB:** getTable(&$db) • fetchFromURL($url) • store($cluster, $data) **ExternalStoreHttp:** fetchFromURL($url) **FakeMemCachedClient:** add ($key, $val, $exp) • decr ($key, $amt) • delete ($key, $time) • disconnect_all () • enable_compress ($enable) • forget_dead_hosts () • get ($key) • get_multi ($keys) • incr ($key, $amt) • replace ($key, $value, $exp) • run_command ($sock, $cmd) • set ($key, $value, $exp) • set_compress_threshold ($thresh) • set_debug ($dbg) • set_servers ($list) **FakeTitle:** error() • __call($name, $args) • isLocal() • isTrans() • getText() • getPartialURL() • getDBkey() • getNamespace() • getNsText() • getSubjectNsText() • getInterwiki() • getFragment() • getDefaultNamespace() • getIndexTitle() • getPrefixedDBkey() • getPrefixedText() • getFullText() • getPrefixedURL() • getFullURL($query , $variant) • getLocalURL($query , $variant) • escapeLocalURL($query) • escapeFullURL($query) • getInternalURL($query , $variant) • getEditURL() • getEscapedText() • isExternal() • isSemiProtected($action) • isProtected($action) • userIsWatching() • userCan($action, $doExpensiveQueries) • userCanCreate() • userCanEdit($doExpensiveQueries) • userCanMove() • isMovable() • userCanRead() • isTalkPage() • isCssJsSubpage() • isValidCssJsSubpage() • getSkinFromCssJsSubpage() • isCssSubpage() • isJsSubpage() • userCanEditCssJsSubpage() • loadRestrictions($res) • getRestrictions($action) • isDeleted() • getArticleID($flags) • getLatestRevID() • resetArticleID($newid) • invalidateCache() • getTalkPage() • getSubjectPage() • getLinksTo() • getTemplateLinksTo() • getBrokenLinksFrom() • getSquidURLs() • moveNoAuth() • isValidMoveOperation() • moveTo() • moveOverExistingRedirect() • moveToNewTitle() • isValidMoveTarget() • getParentCategories() • getParentCategoryTree() • pageCond() • getPreviousRevisionID() • getNextRevisionID() • equals() • exists() • isAlwaysKnown() • touchLinks() • trackbackURL() • trackbackRDF() **FatalError:** getHTML() • getText() **FauxRequest:** FauxRequest($data, $wasPosted , $session) • notImplemented($method) • getText($name, $default) • getValues() • wasPosted() • checkSessionCookie() • getRequestURL() • appendQuery($query) • getHeader($name) • getSessionData($key) • setSessionData($key, $data)

FeedItem: __construct($Title, $Description, $Url, $Date , $Author , $Comments) • xmlEncode($string) • getTitle() • getUrl() • getDescription() • getLanguage() • getDate() • getAuthor() • getComments() **FeedUtils:** :: checkPurge($timekey, $key) • :: checkFeedOutput($type) • :: formatDiff($row) • :: formatDiffRow($title, $oldid, $newid, $timestamp, $comment, $actiontext) • :: applyDiffStyle($text) **FileDeleteForm:** __construct($file) • execute() • :: doDelete(&$title, &$file, &$oldimage, $reason, $suppress) • * showForm() • * showLogEntries() • * prepareMessage($message) • * setHeaders() • :: isValidOldSpec($oldimage) • :: haveDeletableFile(&$file, &$oldfile, $oldimage) • * getAction() • * getTimestamp() **FileDependency:** __construct($filename, $timestamp) • loadDependencyValues() • isExpired() **FileRevertForm:** __construct($file) • execute() • # showForm() • # setHeaders() • # isValidOldSpec() • # haveOldVersion() • # getAction() • # getTimestamp() • # getOldFile() **FileStore:** :: get($group) • * __construct($group, $directory, $path, $hash) • :: lock() • :: unlock() • * :: lockName() • insert($key, $sourcePath, $flags)

• export($key, $destPath, $flags) • * copyFile($sourcePath, $destPath, $flags) • delete($key) • :: deleteFile($path) • stream($key) • :: validKey($key) • :: calculateKey($path, $extension) • filePath($key) • urlPath($key) • * hashPath($key, $separator) **FormatExif:** FormatExif($exif) • getFormattedData() • msg($tag, $val, $arg) • formatNum($num) • formatFraction($num) • gcd($a, $b) **FormOptions:** add($name, $default, $type) • delete($name) • :: guessType($data) • validateName($name, $strict) • setValue($name, $value, $force) • getValue($name) • # getValueReal($option) • reset($name) • consumeValue($name) • consumeValues(/*Array*/ $names) • validateIntBounds($name, $min, $max) • getUnconsumedValues($all) • getChangedValues() • getAllValues() • fetchValuesFromRequest(WebRequest $r, $values) • offsetExists($name) • offsetGet($name) • offsetSet($name, $value) • offsetUnset($name) **FSException:** **FSTransaction:** add(FSTransaction $transaction) • commit() • rollback() • __construct() • addCommit($action, $path) • addRollback($action, $path) • * apply($actions) **global:** js_unescape($source, $iconv_to) • code2utf($num) • wfAjaxWatch($pagename , $watch) • wfLoadAllExtensions() • __autoload($class) • wfQuery($sql, $db, $fname) • wfSingleQuery($sql, $dbi, $fname) • wfIgnoreSQLErrors($newstate, $dbi) • wfFreeResult($res, $dbi) • wfFetchObject($res, $dbi) • wfFetchRow($res, $dbi) • wfNumRows($res, $dbi) • wfNumFields($res, $dbi) • wfFieldName($res, $n, $dbi) • wfInsertId($dbi) • wfDataSeek($res, $row, $dbi) • wfLastErrno($dbi) • wfLastError($dbi) • wfAffectedRows($dbi) • wfLastDBquery($dbi) • wfSetSQL($table, $var, $value, $cond, $dbi) • wfGetSQL($table, $var, $cond, $dbi) • wfFieldExists($table, $field, $dbi) • wfIndexExists($table, $index, $dbi) • wfInsertArray($table, $array, $fname , $dbi) • wfGetArray($table, $vars, $conds, $fname , $dbi) • wfUpdateArray($table, $values, $conds, $fname , $dbi) • wfTableName($name, $dbi) • wfStrencode($s, $dbi) • wfNextSequenceValue($seqName, $dbi) • wfUseIndexClause($index, $dbi) • wfInstallExceptionHandler() • wfReportException(Exception $e) • wfPrintError($message) • wfExceptionHandler($e) • xmlsafe($string) • iconv($from, $to, $string) • mb_substr($str, $start) • mb_strlen($str, $enc) • array_diff_key($left, $right) • wfArrayDiff2($a, $b) • wfArrayDiff2_cmp($a, $b) • wfClone($object) • wfSeedRandom() • wfRandom() • wfUrlencode($s) • wfDebug($text, $logonly) • wfDebugMem($exact) • wfDebugLog($logGroup, $text, $public) • wfLogDBError($text) • wfErrorLog($text, $file) • wfLogProfilingData() • wfReadOnly() • wfReadOnlyReason() • wfGetLangObj($langcode) • wfMsg($key) • wfMsgNoTrans($key) • wfMsgForContent($key) • wfMsgForContentNoTrans($key) • wfMsgNoDB($key) • wfMsgNoDBForContent($key) • wfMsgReal($key, $args, $useDB , $forContent, $transform) • wfMsgWeirdKey ($key) • wfMsgGetKey($key, $useDB, $langCode , $transform) • wfMsgReplaceArgs($message, $args) • wfMsgHtml($key) • wfMsgWikiHtml($key) • wfMsgExt($key, $options) • wfAbruptExit($error) • wfErrorExit() • wfDie($msg) • wfDebugDieBacktrace($msg) • wfHostname() • wfReportTime() • wfDebugBacktrace() • wfBacktrace() • wfShowingResults($offset, $limit) • wfShowingResultsNum($offset, $limit, $num) • wfViewPrevNext($offset, $limit, $link, $query , $atend) • wfNumLink($offset, $limit, &$title, $query) • wfClientAcceptsGzip() • wfCheckLimits($deflimit , $optionname) • wfEscapeWikiText($text) • wfQuotedPrintable($string, $charset) • wfTime() • wfSetVar(&$dest, $source) • wfSetBit(&$dest, $bit, $state) • wfArrayToCGI($array1, $array2) • wfCgiToArray($query) • wfAppendQuery($url, $query) • wfExpandUrl($url) • wfPurgeSquidServers ($urlArr) • wfEscapeShellArg( ) • wfMerge($old, $mine, $yours, &$result) • wfDiff($before, $after, $params) • wfVarDump($var) • wfHttpError($code, $label, $desc) • wfResetOutputBuffers($resetGzipEncoding) • wfClearOutputBuffers() • wfAcceptToPrefs($accept, $def) • mimeTypeMatch($type, $avail) • wfNegotiateType($cprefs, $sprefs) • wfArrayLookup($a, $b) • wfTimestampNow() • wfSuppressWarnings($end) • wfRestoreWarnings() • wfTimestamp($outputtype,$ts) • wfTimestampOrNull($outputtype , $ts) • wfIsWindows() • swap(&$x, &$y) • wfGetCachedNotice($name) • wfGetNamespaceNotice() • wfGetSiteNotice() • wfTempDir() • wfMkdirParents($fullDir, $mode) • wfIncrStats($key) • wfPercent($nr, $acc , $round) • wfEncryptPassword($userid, $password) • wfAppendToArrayIfNotDefault($key, $value, $default, &$changed) • wfEmptyMsg($msg, $wfMsgOut) • in_string($needle, $str) • wfSpecialList($page, $details) • wfUrlProtocols() • wfIniGetBool($setting) • wfShellExec($cmd, &$retval) • wfInitShellLocale() • wfUsePHP($req_ver) • wfUseMW($req_ver) • wfRegexReplacement($string) • wfBaseName($path, $suffix) • wfRelativePath($path, $from) • wfArrayMerge($array1/* ... */) • wfMergeErrorArrays(/*...*/) • wfMakeUrlIndex($url) • wfDoUpdates() • wfExplodeMarkup($separator, $text) • wfBaseConvert($input, $sourceBase, $destBase, $pad, $lowercase) • wfCreateObject($name, $p) • wfGetHTTP($url, $timeout) • wfIsLocalURL($url) • wfHttpOnlySafe() • wfSetupSession() • wfGetPrecompiledData($name) • wfGetCaller($level) • wfGetAllCallers() • wfFormatStackFrame($frame) • wfMemcKey(/*... */) • wfForeignMemcKey($db, $prefix /*, ... */) • wfWikiID($db) • wfSplitWikiID($wiki) • wfGetLB($wiki) • wfFindFile($title, $time , $flags) • wfLocalFile($title) • wfQueriesMustScale() • wfScript($script) • wfBoolToStr($value) • wfLoadExtensionMessages($extensionName, $langcode) • wfGetNull() • wfMaxlagError($host, $lag, $maxLag) • wfDeprecated($function) • wfWaitForSlaves($maxLag) • wfGenerateToken($salt) • wfStripIllegalFilenameChars($name) • addItem($text); • getItem($key); • setText($text); • getText(); • wfScaleSVGUnit($length) • wfGetSVGsize($filename) • wfIsBadImage($name, $contextTitle) • wfFitBoxWidth($boxWidth, $boxHeight, $maxHeight) • memsess_key($id) • memsess_open($save_path, $session_name) • memsess_close() • memsess_read($id) • memsess_write($id, $data) • memsess_destroy($id) • memsess_gc($maxlifetime) • wfOutputHandler($s) • wfRequestExtension() • wfGzipHandler($s) • wfMangleFlashPolicy($s) • wfDoContentLength($length) • wfHtmlValidationHandler($s) • getNavigationBar(); • getBody(); • wfProfileIn($functionname) • wfProfileOut($functionname) • wfGetProfilingOutput($start, $elapsed) • wfProfileClose() • memory_get_usage() • wfProfileIn($fn) • wfProfileOut($fn) • wfGetProfilingOutput($s, $e) • wfProfileClose() • wfGetForwardedFor() • wfGetAgent() • wfGetIP() • wfIsTrustedProxy($ip) • wfProxyCheck() • wfParseCIDR($range) • wfIsLocallyBlockedProxy($ip) • setupUserCss(OutputPage $out) • setupSkinUserCss(OutputPage $out) • getBodyOptions() • getPageClasses($title) • getLogo() • beforeContent() • doBeforeContent() • getCategoryLinks() • drawCategoryBrowser($tree, &$skin) • getCategories() • getQuickbarCompensator($rows) • # afterContentHook() • afterContent() • bottomScripts() • printSource() • printFooter() • doAfterContent() • pageTitleLinks() • getUndeleteLink() • printableLink() • pageTitle() • pageSubtitle() • subPageSubtitle() • showIPinHeader() • nameAndLogin() • getSearchLink() • escapeSearchLink() • searchForm() • topLinks() • extensionTabLinks() • variantLinks() • bottomLinks() • pageStats() • getCopyright($type) • getCopyrightIcon() • getPoweredBy() • lastModified() • logoText($align) • specialPagesList() • mainPageLink() • copyrightLink() • * footerLink ($desc, $page) • privacyLink() • aboutLink() • disclaimerLink() • editThisPage() • editUrlOptions() • deleteThisPage() • protectThisPage() • watchThisPage() • moveThisPage() • historyLink() • whatLinksHere() • userContribsLink() • showEmailUser($id) • emailUserLink() • watchPageLinksLink() • trackbackLink() • otherLanguages() • bugReportsLink() • talkLink() • commentLink() • :: makeMainPageUrl($urlaction) • :: makeSpecialUrl($name, $urlaction) • :: makeSpecialUrlSubpage($name, $subpage, $urlaction) • :: makeI18nUrl($name, $urlaction) • :: makeUrl($name, $urlaction) • :: makeInternalOrExternalUrl($name) • :: makeNSUrl($name, $urlaction , $namespace) • :: makeUrlDetails($name, $urlaction) • :: makeKnownUrlDetails($name, $urlaction) • :: checkTitle(&$title, $name) • buildSidebar() • wfGetType($filename) • wfRFC822Phrase($s) • userMailer($to, $from, $subject, $body, $replyto) • wfElement($element, $attribs , $contents) • wfOpenElement($element, $attribs) • wfCloseElement($element) • HTMLnamespaceselector($selected , $allnamespaces) • wfAttrib($name, $present) • wfLabel($label, $id) • wfEscapeJsString($string) • wfIsWellFormedXml($text) • wfIsWellFormedXmlFragment($text) • wfBuildForm($fields, $submitLabel) **GlobalDependency:** __construct($name) • isExpired() **HashBagOStuff:** __construct() • _expire($key) • get($key) • set($key,$value,$exptime) • delete($key,$time) • keys() **HashtableReplacer:** __construct($table, $index) • replace($matches) **HistoryBlobCurStub:** HistoryBlobCurStub($curid) • setLocation($id) • getText() **HistoryBlobStub:** HistoryBlobStub($hash , $oldid) • setLocation($id) • setReferrer($id) • getReferrer() • getText() • getHash() **HTMLCacheUpdate:** __construct($titleTo, $table) • doUpdate() • # insertJobs(ResultWrapper $res) • # getPrefix() • getFromField() • getToCondition() • invalidateIDs(ResultWrapper $res) **HTMLCacheUpdateJob:** __construct($title, $params, $id) • run() **HTMLFileCache:** HTMLFileCache(&$title) • fileCacheName() • isFileCached() • fileCacheTime() • isFileCacheGood($timestamp) • useGzip() • fetchRawText() • fetchPageText() • loadFromFileCache() • checkCacheDirs() • saveToFileCache($origtext) **Http:** :: isLocalURL($url) **ImageGallery:** __construct( ) • setParser($parser) • setHideBadImages($flag) • setCaption($caption) • setCaptionHtml($caption) • setPerRow($num) • setWidths($num) • setHeights($num) • useSkin($skin) • getSkin() • add($title, $html) • insert($title, $html) • isEmpty() • setShowBytes($f) • setShowFilename($f) • setAttributes($attribs) • toHTML() • count() • setContextTitle($title) • getContextTitle() **ImageHistoryList:** __construct($imagePage) • getImagePage() • getSkin() • getFile() • beginImageHistoryList() • endImageHistoryList() • imageHistoryLine($iscur, $file) **ImagePage:** __construct($title) • # loadFile() • render() • view() • getRedirectTarget() • followRedirect() • isRedirect($text) • isLocal() • getFile() • getDisplayedFile() • getDuplicates() • showTOC($metadata) • makeMetadataTable($metadata) • getContent() • openShowImage() • printSharedImageText() • checkSharedConflict() • checkSharedConflictCallback($repo) • getUploadUrl() • uploadLinksBox() • closeShowImage() • imageHistory() • imageLinks() • imageRedirects() • imageDupes() • delete() • revert() • doPurge() • showError($description) **ImageQueryPage:** # outputResults($out, $skin, $dbr, $res, $num, $offset) • * prepareImage($row) • # getCellHtml($row) **ImportStreamSource:** __construct($handle) • atEnd() • readChunk() • :: newFromFile($filename) • :: newFromUpload($fieldname) • :: newFromURL($url, $method) • :: newFromInterwiki($interwiki, $page, $history) **ImportStringSource:** __construct($string) • atEnd() • readChunk() **IncludableSpecialPage:** IncludableSpecialPage($name, $restriction , $listed , $function , $file) **IndexPager:** __construct() • doQuery() • getResult() • setOffset($offset) • setLimit($limit) • extractResultInfo($offset, $limit, ResultWrapper $res) • reallyDoQuery($offset, $limit, $descending) • # preprocessResults($result) • getBody() • makeLink($text, $query , $type) • getStartBody() • getEndBody() • getEmptyBody() • getTitle() • getSkin() • getDefaultQuery() • getNumRows() • getPagingQueries() • getLimitLinks() • abstract formatRow($row); • abstract getQueryInfo(); • abstract getIndexField(); • # getDefaultDirections() **Interwiki:** __construct($prefix , $url , $local , $trans) • :: isValidInterwiki($prefix) • :: fetch($prefix) • # :: getInterwikiCached($prefix) • # :: getInterwikiCacheEntry($prefix) • # :: load($prefix) • # :: loadFromArray($mc) • getURL($title) • isLocal() • isTranscludable() **IP:** :: isIPAddress($ip) • :: isIPv6($ip) • :: isIPv4($ip) • :: IPv4toIPv6($ip) • :: toUnsigned6($ip) • :: sanitizeIP($ip) • :: toOctet($ip_int) • :: HextoOctet($ip_hex) • :: hexToQuad($ip) • :: parseCIDR6($range) • :: parseRange6($range) • :: isValid($ip) • :: isValidBlock($ipblock) • :: isPublic($ip) • :: toArray($ipblock) • :: toHex($ip) • :: toUnsigned($ip) • :: toSigned($ip) • :: parseCIDR($range) • :: parseRange($range) • :: isInRange($addr, $range) • :: canonicalize($addr) **Job:** abstract run(); • :: pop_type($type) • :: pop($offset) • :: factory($command, $title, $params , $id) • :: makeBlob($params) • :: extractBlob($blob) • :: batchInsert($jobs) • __construct($command, $title, $params , $id) • insert() • # insertFields() • toString() • # setLastError($error) • getLastError() **License:** License($str) **Licenses:** __construct($str) • makeLicenses() • trimStars($str) • stackItem(&$list, $path, $item) • makeHtml(&$tagset, $depth) • outputOption($val, $attribs , $depth) • msg($str) • getLicenses() • getHtml() **LinkBatch:** addObj($title) • add($ns, $dbkey) • setArray($array) • isEmpty() • getSize() • execute() • # executeInto(&$cache) • addResultToCache($cache, $res) • doQuery() • constructSet($prefix, &$db) **LinkCache:** __construct() • forUpdate($update) • getGoodLinkID($title) • getGoodLinkFieldObj($title, $field) • isBadLink($title) • addGoodLinkObj($id, $title, $len , $redir) • addBadLinkObj($title) • clearBadLink($title) • clearLink($title) • getGoodLinks() • getBadLinks() • addLink($title, $len , $redir) • addLinkObj(&$nt, $len , $redirect) • clear() **Linker:** __construct() • postParseLinkColour($s) • getExternalLinkAttributes($title, $unused , $class) • getInterwikiLinkAttributes($title, $unused , $class) • getInternalLinkAttributes($title, $unused , $class) • getInternalLinkAttributesObj($nt, $unused , $class , $title) • * getLinkAttributesInternal($title, $class, $classDefault) • getLinkColour($t, $threshold) • * linkUrl($target, $query, $options) • * linkAttribs($target, $attribs, $options) • * linkText($target) • makeLink($title, $text , $query , $trail) • makeKnownLink($title, $text , $query , $trail , $prefix ,$aprops) • makeBrokenLink($title, $text , $query , $trail) • makeStubLink($title, $text , $query , $trail) • makeLinkObj($nt, $text, $query , $trail , $prefix) • makeKnownLinkObj($title, $text , $query , $trail , $prefix , $aprops , $style) • makeBrokenLinkObj($title, $text , $query , $trail , $prefix) • makeStubLinkObj($nt, $text , $query , $trail , $prefix) • makeColouredLinkObj($nt, $colour, $text , $query , $trail , $prefix) • makeSizeLinkObj($size, $nt, $text , $query , $trail , $prefix) • makeSelfLinkObj($nt, $text , $query , $trail , $prefix) • normaliseSpecialPage(Title $title) • fnamePart($url) • makeImage($url, $alt) • makeExternalImage($url, $alt) • makeBrokenImageLinkObj($title, $text , $query , $trail , $prefix , $time) • makeMediaLink($name, $unused , $text , $time) • makeMediaLinkObj($title, $text , $time) • specialLink($name, $key) • userLink($userId, $userText) • userToolLinks($userId, $userText, $redContribsWhenNoEdits , $flags , $edits) • userToolLinksRedContribs($userId, $userText, $edits) • userTalkLink($userId, $userText) • blockLink($userId, $userText) • revUserLink($rev, $isPublic) • revUserTools($rev, $isPublic) • formatComment($comment, $title , $local) • * formatAutocomments($comment, $title , $local) • * formatAutocommentsCallback($match) • formatLinksInComment($comment) • # formatLinksInCommentCallback($match) • commentBlock($comment, $title , $local) • revComment(Revision $rev, $local , $isPublic) • formatRevisionSize($size) • tocIndent() • tocUnindent($level) • tocLine($anchor, $tocline, $tocnumber, $level) • tocLineEnd() • tocList($toc) • editSectionLinkForOther($title, $section) • editSectionLink(Title $nt, $section, $hint) • doEditSectionLink(Title $nt, $section, $tooltip) • makeHeadline($level, $attribs, $anchor, $text, $link) • :: splitTrail($trail) • generateRollback($rev) • buildRollbackLink($rev) • formatTemplates($templates, $preview , $section) • formatHiddenCategories($hiddencats) • formatSize($size) • tooltipAndAccesskey($name) • tooltip($name, $options) • titleAttrib($name, $options) • accesskey($name) **LinkFilter:** :: matchEntry($text, $filterEntry) • * :: makeRegex($filterEntry) • :: makeLike($filterEntry , $prot) **LinksUpdate:** LinksUpdate($title, $parserOutput, $recursive) • setRecursiveTouch($val) • doUpdate() • # doIncrementalUpdate() • # doDumbUpdate() • queueRecursiveJobs() • invalidatePages($namespace, $dbkeys) • invalidateCategories($cats) • updateCategoryCounts($added, $deleted) • invalidateImageDescriptions($images) •

dumbTableUpdate($table, $insertions, $fromField) • makeWhereFrom2d(&$arr, $prefix) • incrTableUpdate($table, $prefix, $deletions, $insertions) • getLinkDeletions($existing) • getTemplateDeletions($existing) • getImageDeletions($existing) • getExternalDeletions($existing) • getCategoryDeletions($existing) • getInterlangDeletions($existing) • getPropertyDeletions($existing) • getExistingLinks() • getExistingTemplates() • getExistingImages() • getExistingExternals() • getExistingCategories() • getExistingInterlangs() • getExistingProperties() • getTitle() • invalidateProperties($changed) **LogEventsList:** __construct($skin, $out, $flags) • * preCacheMessages() • showHeader($type) • * getFilterLinks($logType, $filter) • * getDefaultQuery() • * getTypeMenu($queryType) • * getUserInput($user) • * getTitleInput($title) • * getDateMenu($year, $month) • * getTitlePattern($pattern) • beginLogEventsList() • endLogEventsList() • logLine($row) • * getShowHideLinks($row) • :: typeAction($row, $type, $action) • :: userCan($row, $field) • :: isDeleted($row, $field) • :: getExcludeClause($db) **LogPage:** __construct($type, $rc) • # saveContent() • getRcComment() • getComment() • :: validTypes() • :: isLogType($type) • :: logName($type) • :: logHeader($type) • # :: getTitleLink($type, $skin, $title, &$params) • :: makeParamBlob($params) • :: extractParams($blob) • :: formatBlockFlags($flags, $forContent) • :: formatBlockFlag($flag, $forContent) **LogPager:** getDefaultQuery() • getFilterParams() • * limitType($type) • * limitUser($name) • * limitTitle($page, $pattern) • getQueryInfo() • getIndexField() • getStartBody() • formatRow($row) • getType() • getUser() • getPage() • getPattern() • getYear() • getMonth() **LogReader:** __construct($request) • hasRows() **LogViewer:** __construct(&$reader, $flags) • show() • showList(&$out) **MacBinary:** __construct($filename) • open($filename) • isValid() • dataForkLength() • extractData($destination) • close() • loadHeader() • calcCRC($data, $seed) • copyBytesTo($destination, $bytesToCopy) • hexdump($data) **MagicWord:** :: getVariableIDs() • :: getCacheTTL($id) • :: getDoubleUnderscoreArray() • load($id) • initRegex() • getRegex() • getRegexCase() • getRegexStart() • getBaseRegex() • match($text) • matchStart($text) • matchVariableStartToEnd($text) • matchAndRemove(&$text) • matchStartAndRemove(&$text) • pregRemoveAndRecord( ) • replace($replacement, $subject, $limit) • substituteCallback($text, $callback) • getVariableRegex() • getVariableStartToEndRegex() • getSynonym($i) • getSynonyms() • getWasModified() • replaceMultiple($magicarr, $subject, &$result) • addToArray(&$array, $value) • isCaseSensitive() • getId() **MagicWordArray:** add($name) • addArray($names) • getHash() • getBaseRegex() • getRegex() • getVariableRegex() • getVariableStartToEndRegex() • parseMatch($m) • matchVariableStartToEnd($text) • matchStartToEnd($text) • matchAndRemove(&$text) **MailAddress:** __construct($address, $name , $realName) • toString() • __toString() **MathRenderer:** setOutputMode($mode) • render() • _error($msg, $append) • _recall() • _doRender() • _linkToMathImage() • _getHashPath() **MediaTransformError:** __construct($msg, $width, $height /*, ... */) • toText() • getHtmlMsg() • isError() **MediaTransformOutput:** getWidth() • getHeight() • getUrl() • getPath() • isError() • # linkWrap($linkAttribs, $contents) • getDescLinkAttribs($alt , $params) **MediaWiki:** __construct() • setVal($key, &$value) • getVal($key, $default) • initialize(&$title, &$article, &$output, &$user, $request) • checkMaxLag($maxLag) • checkInitialQueries($title, $action) • preliminaryChecks(&$title, &$output, $request) • initializeSpecialCases(&$title, &$output, $request) • :: articleFromTitle(&$title) • initializeArticle(&$title, $request) • finalCleanup (&$deferredUpdates, &$output) • doUpdates(&$updates) • doJobs() • restInPeace() • performAction(&$output, &$article, &$title, &$user, &$request) **MediaWikiBagOStuff:** _getDB() • _begin() • _commit() • _doquery($sql) • _doinsert($t, $v) • _fetchobject($result) • _freeresult($result) • _dberror($result) • _maxdatetime() • _fromunixtime($ts) • _readonly() • _strencode($s) • _blobencode($s) • _blobdecode($s) • getTableName() MediaWiki_I18N: set($varName, $value) • translate($value) **memcached:** memcached ($args) • add ($key, $val, $exp) • decr ($key, $amt) • delete ($key, $time) • disconnect_all () • enable_compress ($enable) • forget_dead_hosts () • get ($key) • get_multi ($keys) • incr ($key, $amt) • replace ($key, $value, $exp) • run_command ($sock, $cmd) • set ($key, $value, $exp) • set_compress_threshold ($thresh) • set_debug ($dbg) • set_servers ($list) • set_timeout ($seconds, $microseconds) • _close_sock ($sock) • _connect_sock (&$sock, $host) • _dead_sock ($sock) • get_sock ($key) • _hashfunc ($key) • _incrdecr ($cmd, $key, $amt) • _load_items ($sock, &$ret) • _set ($cmd, $key, $val, $exp) • sock_to_host ($host) • _debugprint($str) • _safe_fwrite($f, $buf, $len) • _flush_read_buffer($f) **MemCachedClientforWiki:** _debugprint($text) **MessageCache:** __construct(&$memCached, $useDB, $expiry, /*ignored*/ $memcPrefix) • getParserOptions() • loadFromLocal($hash, $code) • saveToLocal($serialized, $hash, $code) • saveToScript($array, $hash, $code) • escapeForScript($string) • setCache($cache, $code) • load($code) • loadFromDB($code) • replace($title, $text) • # saveToCaches($cache, $memc , $code) • lock($key) • unlock($key) • get($key, $useDB , $langcode , $isFullKey) • getMsgFromNamespace($title, $code) • transform($message, $interface) • disable() • enable() • disableTransform() • enableTransform() • setTransform($x) • getTransform() • addMessage($key, $value, $lang) • addMessages($messages, $lang) • addMessagesByLang($messages) • getExtensionMessagesFor($lang) • clear() • loadAllMessages($lang) • loadMessagesFile($filename, $langcode) • processMessagesArray($messages, $langcode) • figureMessage($key) **MimeMagic:** __construct() • getExtensionsForType($mime) • getTypesForExtension($ext) • guessTypesForExtension($ext) • isMatchingExtension($extension, $mime) • isPHPImageType($mime) • isRecognizableExtension($extension) • guessMimeType($file, $ext) • doGuessMimeType($file, $ext) • detectZipType($header) • detectMimeType($file, $ext) • getMediaType($path , $mime) • findMediaType($extMime) **MWException:** useOutputPage() • useMessageCache() • msg($key, $fallback /*[, params...] */) • getHTML() • getText() • getPageTitle() • getLogMessage() • reportHTML() • report() • htmlHeader() • htmlFooter() **MWNamespace:** :: isMovable($index) • :: isMain($index) • :: isTalk($index) • :: getTalk($index) • :: getSubject($index) • :: getCanonicalName($index) • :: getCanonicalIndex($name) • :: canTalk($index) • :: isContent($index) • :: isWatchable($index) • :: hasSubpages($index) **MySQLSearchResultSet:** MySQLSearchResultSet($resultSet, $terms) • termMatches() • numRows() • next() • free() **Namespace:** **OldChangesList:** recentChangesLine(&$rc, $watched) **OracleSearchResultSet:** __construct($resultSet, $terms) • termMatches() • numRows() • next() **OutputPage:** __construct() • redirect($url, $responsecode) • getRedirect() • setStatusCode($statusCode) • addMeta($name, $val) • addKeyword($text) • addScript($script) • addExtensionStyle($url) • addScriptFile($file) • addInlineScript($script) • getScript() • getHeadItems() • addHeadItem($name, $value) • hasHeadItem($name) • setETag($tag) • setArticleBodyOnly($only) • getArticleBodyOnly($only) • addLink($linkarr) • getExtStyle() • addMetadataLink($linkarr) • checkLastModified ($timestamp) • setPageTitleActionText($text) • getPageTitleActionText () • setRobotPolicy($policy) • setIndexPolicy($policy) • setFollowPolicy($policy) • setHTMLTitle($name) • setPageTitle($name) • getHTMLTitle() • getPageTitle() • setSubtitle($str) // @bug 2514 • appendSubtitle($str) // @bug 2514 • getSubtitle() • isArticle() • setPrintable() • isPrintable() • setSyndicated($show) • isSyndicated() • setFeedAppendQuery($val) • getFeedAppendQuery() • setOnloadHandler($js) • getOnloadHandler() • disable() • setArticleRelated($v) • setArticleFlag($v) • isArticleRelated() • getLanguageLinks() • addLanguageLinks($newLinkArray) • setLanguageLinks($newLinkArray) • getCategoryLinks() • addCategoryLinks($categories) • setCategoryLinks($categories) • suppressQuickbar() • isQuickbarSuppressed() • disallowUserJs() • isUserJsAllowed() • prependHTML($text) • addHTML($text) • clearHTML() • getHTML() • debug($text) • setParserOptions($options) • parserOptions($options) • setRevisionId($revid) • getRevisionId() • addWikiText($text, $linestart) • addWikiTextWithTitle($text, &$title, $linestart) • addWikiTextTitleTidy($text, &$title, $linestart) • addWikiTextTitle($text, &$title, $linestart, $tidy) • addParserOutputNoText(&$parserOutput) • addParserOutput(&$parserOutput) • addPrimaryWikiText($text, $article, $cache) • addSecondaryWikiText($text, $linestart) • addWikiTextTidy($text, $linestart) • addTemplate(&$template) • parse($text, $linestart , $interface) • tryParserCache(&$article, $user) • setSquidMaxage($maxage) • enableClientCache($state) • getCacheVaryCookies() • uncacheableBecauseRequestVars() • haveCacheVaryCookies() • getXVO() • sendCacheControl() • output() • out($ins) • :: setEncodings() • reportTime() • blockedPage($return) • showPermissionsErrorPage($errors, $action) • errorpage($title, $msg) • versionRequired($version) • permissionRequired($permission) • sysopRequired() • developerRequired() • loginToUse() • databaseError($fname, $sql, $error, $errno) • formatPermissionsErrorMessage($errors, $action) • fatalError($message) • unexpectedValueError($name, $val) • fileCopyError($old, $new) • fileRenameError($old, $new) • fileDeleteError($name) • fileNotFoundError($name) • showFatalError($message) • showUnexpectedValueError($name, $val) • showFileCopyError($old, $new) • showFileRenameError($old, $new) • showFileDeleteError($name) • showFileNotFoundError($name) • addReturnTo($title) • returnToMain($unused , $returnto) • * addKeywords(&$parserOutput) • headElement(Skin $sk) • # addDefaultMeta() • getHeadLinks() • getSyndicationLinks() • * feedLink($type, $url, $text) • addStyle($style, $media, $condition, $dir) • buildCssLinks() • # styleLink($style, $options) • transformCssMedia($media) • rateLimited() • showNewSectionLink() • showLagWarning($lag) • addWikiMsg(/*...*/) • wrapWikiMsg($wrap /*, .../) **PageHistory:** __construct($article) • getArticle() • getTitle() • preCacheMessages() • history() • * getDateMenu($year, $month) • beginHistoryList() • endHistoryList() • historyLine($row, $next, $counter , $notificationtimestamp , $latest , $firstInList) • revLink($rev) • curLink($rev, $latest) • lastLink($prevRev, $next, $counter) • diffButtons($rev, $firstInList, $counter) • getLatestId() • fetchRevisions($limit, $offset, $direction) • getNotificationTimestamp() • feed($type) • feedEmpty() • feedItem($row) • stripComment($text) **PageHistoryPager:** __construct($pageHistory, $year, $month) • getQueryInfo() • getIndexField() • formatRow($row) • getStartBody() • getEndBody() **PageQueryPage:** formatResult($skin, $row) **PasswordError:** **PatrolLog:** :: record($change, $auto) • :: makeActionText($title, $params, $skin) • * :: buildParams($change, $auto) **PostgresSearchResult:** __construct($row) • getScore() **PostgresSearchResultSet:** __construct($resultSet, $terms) • termMatches() • numRows() • next() **PrefixSearch:** # :: searchBackend($namespaces, $search, $limit) • # :: specialSearch($search, $limit) • # :: defaultSearchBackend($namespaces, $search, $limit) • # :: validateNamespaces($namespaces) **Profiler:** __construct() • profileIn($functionname) • profileOut($functionname) • close() • getOutput() • getCallTree() • remapCallTree($stack) • getCallTreeLine($entry) • getTime() • getUserTime() • getFunctionReport() • calltreeCount($stack, $start) • :: logToDB($name, $timeSum, $eventCount, $memorySum) • getCurrentSection() • :: getCaller($level) • debug($s) **ProfilerSimple:** __construct() • setMinimum($min) • setProfileID($id) • getProfileID() • profileIn($functionname) • profileOut($functionname) • getFunctionReport() • getCpuTime($ru) • getTime($time) **ProfilerSimpleText:** getFunctionReport() • :: sort($a,$b) • :: format($item,$key) **ProfilerSimpleUDP:** getFunctionReport() **ProtectionForm:** __construct(Article $article) • getExpiry($action) • execute() • show($err) • save() • buildForm() • buildSelector($action, $selected) • * getOptionLabel($permission) • buildScript() • buildCleanupScript() • showLogExtract(&$out) **QueryPage:** setListoutput($bool) • getName() • getTitle() • getSQL() • sortDescending() • getOrder() • isExpensive( ) • isCached() • isSyndicated() • formatResult($skin, $result) • getPageHeader( ) • linkParameters() • tryLastResult( ) • recache($limit, $ignoreErrors) • doQuery($offset, $limit, $shownavigation) • # outputResults($out, $skin, $dbr, $res, $num, $offset) • openList($offset) • closeList() • preprocessResults($db, $res) • doFeed($class , $limit) • feedResult($row) • feedItemDesc($row) • feedItemAuthor($row) • feedTitle() • feedDesc() • feedUrl() **QuickTemplate:** QuickTemplate() • set($name, $value) • setRef($name, &$value) • setTranslator(&$t) • execute() • text($str) • jstext($str) • html($str) • msg($str) • msgHtml($str) • msgWiki($str) • haveData($str) • haveMsg($str) **RawPage:** __construct(&$article, $request) • view() • getRawText() • getArticleText() • parseArticleText($text) **RCCacheEntry:** :: newFromParent($rc) **RdfMetaData:** __construct(Article $article) • abstract show(); • # setup() • # reallyFullUrl() • # basics() • # element($name, $value) • # date($timestamp) • # pageOrString($name, $page, $str) • # page($name, $title) • # url($name, $url) • # person($name, User $user) • # rights() • # getTerms($url) • # getKnownLicenses() **RecentChange:** :: newFromRow($row) • :: newFromCurRow($row) • :: newFromId($rcid) • :: newFromConds($conds, $fname) • setAttribs($attribs) • setExtra($extra) • getMovedToTitle() • save() • :: sendToUDP($line, $address , $prefix) • :: cleanupForIRC($text) • :: markPatrolled($change, $auto) • doMarkPatrolled($auto) • reallyMarkPatrolled() • :: notifyMove($timestamp, &$oldTitle, &$newTitle, &$user, $comment, $ip, $overRedir) • :: notifyMoveToNew($timestamp, &$oldTitle, &$newTitle, &$user, $comment, $ip) • :: notifyMoveOverRedirect($timestamp, &$oldTitle, &$newTitle, &$user, $comment, $ip) • loadFromRow($row) • loadFromCurRow($row) • getAttribute($name) • diffLinkTrail($forceCur) • # getIRCLine() • getCharacterDifference($old , $new) **RefreshLinksJob:** __construct($title, $params , $id) • run() RefreshLinksJob2: __construct($title, $params, $id) • run() **RegexlikeReplacer:** __construct($r) • replace($matches) **ReplacementArray:** __sleep() • __wakeup() • setArray($data) • getArray() • setPair($from, $to) • mergeArray($data) • merge($other) • removePair($from) • removeArray($data) • replace($subject) **Replacer:** cb() **ReverseChronologicalPager:** __construct() • getNavigationBar() • getDateCond($year, $month) **Revision:** :: newFromId($id) • :: newFromTitle($title, $id) • :: loadFromId($db, $id) • :: loadFromPageId($db, $pageid, $id) • :: loadFromTitle($db, $title, $id) • :: loadFromTimestamp($db, $title, $timestamp) • * :: newFromConds($conditions) • * :: loadFromConds($db, $conditions) • :: fetchAllRevisions($title) • :: fetchRevision($title) • * :: fetchFromConds($db, $conditions) • :: selectFields() • :: selectTextFields() • :: selectPageFields() • Revision($row) • getId() • getTextId() • getParentId() • getSize() • getTitle() • setTitle($title) • getPage() • getUser($audience) • getRawUser() • getUserText($audience) • getRawUserText() • getComment($audience) • getRawComment() • isMinor() • isDeleted($field) • getVisibility() • getText($audience) • revText() • getRawText() • getTimestamp() • isCurrent() • getPrevious() • getNext() • * getPreviousRevisionId($db) • :: getRevisionText($row, $prefix) • :: compressRevisionText(&$text) • insertOn($dbw) • * loadText() • :: newNullRevision($dbw, $pageId, $summary, $minor) • userCan($field) • :: getTimestampFromId($title, $id) • :: countByPageId($db, $id) • :: countByTitle($db, $title) **RSSFeed:** formatTime($ts) • outHeader() • outItem($item) • outFooter() **Sanitizer:** :: removeHTMLcomments($text) • :: validateTagAttributes($attribs, $element) • :: validateAttributes($attribs, $whitelist) • :: mergeAttributes($a, $b) • :: checkCss($value) • :: fixTagAttributes($text, $element) • ::

encodeAttribute($text) • :: safeEncodeAttribute($text) • :: escapeId($id, $flags) • :: escapeClass($class) • :: escapeHtmlAllowEntities($html) • * :: armorLinksCallback($matches) • :: decodeTagAttributes($text) • * :: getTagAttributeCallback($set) • * :: normalizeAttributeValue($text) • * :: normalizeWhitespace($text) • :: normalizeCharReferences($text) • :: normalizeCharReferencesCallback($matches) • :: normalizeEntity($name) • :: decCharReference($codepoint) • :: hexCharReference($codepoint) • * :: validateCodepoint($codepoint) • :: decodeCharReferences($text) • :: decodeCharReferencesCallback($matches) • :: decodeChar($codepoint) • :: decodeEntity($name) • :: attributeWhitelist($element) • :: setupAttributeWhitelist() • :: stripAllTags($text) • :: hackDocType() • :: cleanUrl($url) **SearchEngine:** searchText($term) • searchTitle($term) • :: getNearMatch($searchterm) • :: legalSearchChars() • setLimitOffset($limit, $offset) • setNamespaces($namespaces) • replacePrefixes($query) • :: searchableNamespaces() • :: userNamespaces(&$user) • :: userHighlightPrefs(&$user) • :: defaultNamespaces() • filter($text) • :: create() • update($id, $title, $text) • updateTitle($id, $title) • :: getOpenSearchTemplate() • :: getMWSuggestTemplate() **SearchEngineDummy:** **SearchHighlighter:** SearchHighlighter($cleanupWikitext) • highlightText($text, $terms, $contextlines, $contextchars) • splitAndAdd(&$extracts, &$count, $text) • caseCallback($matches) • extract($text, $start, $end, &$posStart , &$posEnd) • position($text, $point, $offset) • process($pattern, $extracts, &$linesleft, &$contextchars, &$out, &$offsets) • removeWiki($text) • linkReplace($matches) • highlightSimple($text, $terms, $contextlines, $contextchars) **SearchMySQL:** __construct($db) • parseQuery($filteredText, $fulltext) • :: legalSearchChars() • searchText($term) • searchTitle($term) • queryRedirect() • queryNamespaces() • queryLimit() • queryRanking($filteredTerm, $fulltext) • getQuery($filteredTerm, $fulltext) • getIndexField($fulltext) • queryMain($filteredTerm, $fulltext) • update($id, $title, $text) • updateTitle($id, $title) SearchMySQL4: **SearchOracle:** __construct($db) • searchText($term) • searchTitle($term) • queryRedirect() • queryNamespaces() • queryLimit($sql) • queryRanking($filteredTerm, $fulltext) • getQuery($filteredTerm, $fulltext) • getIndexField($fulltext) • queryMain($filteredTerm, $fulltext) • parseQuery($filteredText, $fulltext) • update($id, $title, $text) • updateTitle($id, $title) **SearchPostgres:** __construct($db) • searchTitle($term) • searchText($term) • parseQuery($term) • searchQuery($term, $fulltext, $colname) • update($pageid, $title, $text) • updateTitle($id, $title) **SearchResult:** SearchResult($row) • isBrokenTitle() • isMissingRevision() • getTitle() • getScore() • # initText() • getTextSnippet($terms) • getTitleSnippet($terms) • getRedirectSnippet($terms) • getRedirectTitle() • getSectionSnippet() • getSectionTitle() • getTimestamp() • getWordCount() • getByteSize() • hasRelated() • getInterwikiPrefix() **SearchResultSet:** termMatches() • numRows() • hasResults() • getTotalHits() • hasSuggestion() • getSuggestionQuery() • getSuggestionSnippet() • getInfo() • getInterwikiResults() • hasInterwikiResults() • next() • free() **SearchResultTooMany:** **SearchUpdate:** SearchUpdate($id, $title, $text) • doUpdate() **SearchUpdateMyISAM:** **SiteConfiguration:** # getSetting($settingName, $wiki, /*array*/ $params) • doReplace($from, $to, $in) • initialise() • extractGlobalSetting($setting, $wiki, $params) • # getWikiParams($wiki) • # mergeParams($wiki, $suffix, /*array*/ $params, /*array*/ $wikiTags) • siteFromDB($db) • isLocalVHost($vhost) • :: arrayMerge($array1/* ... */) **SiteStats:** :: recache() • :: load($recache) • :: loadAndLazyInit() • :: doLoad($db) • :: views() • :: edits() • :: articles() • :: pages() • :: users() • :: activeUsers() • :: images() • :: admins() • :: numberingroup($group) • :: jobs() • :: pagesInNs($ns) • * :: isSane($row) **SiteStatsUpdate:** __construct($views, $edits, $good, $pages , $users) • appendUpdate(&$sql, $field, $delta) • doUpdate() • :: cacheUpdate($dbw) **Skin:** :: getSkinNames() • :: getUsableSkins() • :: normalizeKey($key) • getStylesheet() • getSkinName() • qbSetting() • initPage(OutputPage $out) • preloadExistence() • addMetadataLinks(OutputPage $out) • setMembers() • outputPage(OutputPage $out) • :: makeVariablesScript($data) • :: makeGlobalVariablesScript($data) • getHeadScripts($allowUserJs) • userCanPreview($action) • generateUserJs() • generateUserStylesheet() • # reallyGenerateUserStylesheet() **SkinTemplate:** initPage(OutputPage $out) • setupSkinUserCss(OutputPage $out) • setupTemplate($classname, $repository, $cache_dir) • outputPage(OutputPage $out) • printOrError($str) • buildPersonalUrls() • tabAction($title, $message, $selected, $query, $checkEdit) • makeTalkUrlDetails($name, $urlaction) • makeArticleUrlDetails($name, $urlaction) • buildContentActionUrls() • buildNavUrls() • getNameSpaceKey() • setupUserJs($allowUserJs) • setupPageCss() **SpecialMycontributions:** __construct() • getRedirect($subpage) **SpecialMypage:** __construct() • getRedirect($subpage) **SpecialMytalk:** __construct() • getRedirect($subpage) **SpecialPage:** :: initList() • :: initAliasList() • :: resolveAlias($alias) • :: resolveAliasWithSubpage($alias) • :: addPage(&$page) • :: setGroup($page, $group) • :: getGroup(&$page) • :: removePage($name) • :: exists($name) • :: getPage($name) • :: getPageByAlias($alias) • :: getUsablePages() • :: getRegularPages() • :: getRestrictedPages() • :: executePath(&$title, $including) • :: capturePath(&$title) • :: getLocalNameFor($name, $subpage) • :: getTitleFor($name, $subpage) • :: getSafeTitleFor($name, $subpage) • :: getTitleForAlias($alias) • SpecialPage($name , $restriction , $listed , $function , $file , $includable) • getName() • getRestriction() • getFile() • isListed() • name($x) • restrictions($x) • listed($x) • func($x) • file($x) • includable($x) • including($x) • getLocalName() • isRestricted() • userCanExecute($user) • displayRestrictionError() • setHeaders() • execute($par) • outputHeader() • getDescription() • getTitle($subpage) • setListed($listed) • getRedirect($subpage) • getRedirectQuery() **SpecialRedirectToSpecial:** getRedirect($subpage) **SqlBagOStuff:** __construct($tablename) • get($key) • set($key,$value,$exptime) • delete($key,$time) • keys() • getTableName() • _query($sql) • _strencode($str) • _blobencode($str) • _blobdecode($str) • abstract _doinsert($table, $vals); • abstract _doquery($sql); • abstract _readonly(); • _begin() • _commit() • _freeresult($result) • _dberror($result) • abstract _maxdatetime(); • abstract _fromunixtime($ts); • garbageCollect() • expireall() • deleteall() • _serialize(&$data) • _unserialize($serial) **SquidUpdate:** :: newFromLinksTo(&$title) • :: newSimplePurge(&$title) • doUpdate() • :: purge($urlArr) • :: HTCPPurge($urlArr) • debug($text) • :: expand($url) **Status:** :: newFatal($message /*, parameters...*/) • :: newGood($value) • setResult($ok, $value) • isGood() • isOK() • warning($message /*, parameters... */) • error($message /*, parameters... */) • fatal($message /*, parameters... */) • # cleanParams($params) • # getItemXML($item) • getXML() • getWikiText($shortContext , $longContext) • merge($other, $overwriteValue) • getErrorsArray() • hasMessage($msg) **StringUtils:** :: hungryDelimiterReplace($startDelim, $endDelim, $replace, $subject) • :: delimiterReplaceCallback($startDelim, $endDelim, $callback, $subject, $flags) • :: delimiterReplace($startDelim, $endDelim, $replace, $subject, $flags) • :: explodeMarkup($separator, $text) • :: escapeRegexReplacement($string) • :: explode($separator, $subject) **StubContLang:** __construct() • __call($name, $args) • _newObject() **StubObject:** :: isRealObject($obj) • _call($name, $args) • _newObject() • __call($name, $args) • _unstub($name , $level) **StubUser:** __construct() • __call($name, $args) • _newObject() **StubUserLang:** __construct() • __call($name, $args) • _newObject() **TablePager:** __construct() • getStartBody() • getEndBody() • getEmptyBody() • formatRow($row) • getIndexField() • getTableClass() • getNavClass() • getSortHeaderClass() • getNavigationBar() • getLimitSelect() • getLimitForm() • abstract isFieldSortable($field); • abstract formatValue($name, $value); • abstract getDefaultSort(); • abstract getFieldNames(); **ThumbnailImage:** ThumbnailImage($file, $url, $width, $height, $path , $page) **Title:** /* * */ __construct() • :: newFromDBkey($key) • :: newFromText($text, $defaultNamespace) • :: newFromURL($url) • :: newFromID($id, $flags) • :: newFromIDs($ids) • :: newFromRow($row) • :: makeTitleSafe($ns, $title, $fragment) • :: newMainPage() • :: newFromRedirect($text) • :: nameOf($id) • :: legalChars() • :: indexTitle($ns, $title) • :: makeName($ns, $title, $fragment) • getInterwikiLink($key) • isLocal() • isTrans() • :: escapeFragmentForURL($fragment) • getText() • getPartialURL() • getDBkey() • getNamespace() • getNsText() • getUserCaseDBKey() • getSubjectNsText() • getTalkNsText() • canTalk() • getInterwiki() • getFragment() • getFragmentForURL() • getDefaultNamespace() • getIndexTitle() • getPrefixedDBkey() • getPrefixedText() • getFullText() • getBaseText() • getSubpageText() • getSubpageUrlForm() • getPrefixedURL() • getFullURL($query , $variant) • getLocalURL($query , $variant) • escapeLocalURL($query) • escapeFullURL($query) • getInternalURL($query , $variant) • getEditURL() • getEscapedText() • isExternal() • isSemiProtected($action) • isProtected($action) • userIsWatching() • quickUserCan($action) • isNamespaceProtected() • userCan($action, $doExpensiveQueries) • * getUserPermissionsErrorsInternal($action, $user, $doExpensiveQueries) • * getTitleProtection() • updateTitleProtection($create_perm, $reason, $expiry) • deleteTitleProtection() • userCanEdit($doExpensiveQueries) • userCanCreate($doExpensiveQueries) • userCanMove($doExpensiveQueries) • isMovable() • userCanRead() • isTalkPage() • isSubpage() • hasSubpages() • isCssOrJsPage() • isCssJsSubpage() • isValidCssJsSubpage() • getSkinFromCssJsSubpage() • isCssSubpage() • isJsSubpage() • userCanEditCssJsSubpage() • isCascadeProtected() • getCascadeProtectionSources($get_pages) • areRestrictionsCascading() • * loadRestrictionsFromRow($res, $oldFashionedRestrictions) • loadRestrictions($oldFashionedRestrictions) • :: purgeExpiredRestrictions() • getRestrictions($action) • getRestrictionExpiry($action) • isDeleted() • getArticleID($flags) • isRedirect($flags) • getLength($flags) • getLatestRevID($flags) • resetArticleID($newid) • invalidateCache() • /* * */ prefix($name) • * secureAndSplit() • setFragment($fragment) • getTalkPage() • getSubjectPage() • getLinksTo($options , $table , $prefix) • getTemplateLinksTo($options) • getBrokenLinksFrom($options) • getSquidURLs() • purgeSquid() • moveNoAuth(&$nt) • isValidMoveOperation(&$nt, $auth , $reason) • moveTo(&$nt, $auth , $reason , $createRedirect) • * moveOverExistingRedirect(&$nt, $reason , $createRedirect) • * moveToNewTitle(&$nt, $reason , $createRedirect) • isValidMoveTarget($nt) • isWatchable() • getParentCategories() • pageCond() • getPreviousRevisionID($revId, $flags) • getNextRevisionID($revId, $flags) • getEarliestRevTime() • countRevisionsBetween($old, $new) • equals(Title $title) • :: compare($a, $b) • __toString() • exists() • isAlwaysKnown() • touchLinks() • getTouched($db) • trackbackURL() • trackbackRDF() • getNamespaceKey() • isSpecial($name) • fixSpecialName() • isContentPage() • getRedirectsHere($ns) **TitleArray:** :: newFromResult($res) • # :: newFromResult_internal($res) **TitleArrayFromResult:** __construct($res) • # setCurrent($row) • count() • current() • key() • next() • rewind() • valid() **TitleDependency:** __construct(Title $title) • loadDependencyValues() • __sleep() • getTitle() • isExpired() **TitleListDependency:** __construct(LinkBatch $linkBatch) • calculateTimestamps() • loadDependencyValues() • __sleep() • getLinkBatch() • isExpired() **TransformParameterError:** __construct($params) **TurckBagOStuff:** get($key) • set($key, $value, $exptime) • delete($key, $time) • lock($key, $waitTimeout) • unlock($key) **UnlistedSpecialPage:** UnlistedSpecialPage($name, $restriction , $function , $file) **UploadBase:** :: isEnabled() • :: isAllowed($user) • :: createFromRequest(&$request, $type) • :: isValidRequest($request) • __construct() • initialize($name, $tempPath, $fileSize, $removeTempFile) • fetchFile() • verifyUpload() • # verifyFile($tmpfile) • verifyPermissions($user) • checkWarnings() • performUpload($comment, $pageText, $watch, $user) • getTitle() • getLocalFile() • saveTempUploadedFile($saveName, $tempName) • stashSession() • unsaveUploadedFile() • cleanupTempFile() • getTempPath() • splitExtensions($filename) • checkFileExtension($ext, $list) • checkFileExtensionList($ext, $list) • :: verifyExtension($mime, $extension) • detectScript($file, $mime, $extension) • detectVirus($file) • checkMacBinary() • checkOverwrite() • :: userCanReUpload(User $user, $img) • :: getExistsWarning($file) • :: isThumbName($filename) • :: getFilenamePrefixBlacklist() **UploadFromStash:** :: isValidSessionKey($key, $sessionData) • :: isValidRequest($request) • initialize($name, $sessionData) • initializeFromRequest(&$request) • # verifyFile($tmpfile) • checkWarnings() **UploadFromUpload:** initializeFromRequest(&$request) • :: isValidRequest($request) **UploadFromUrl:** :: isAllowed($user) • :: isEnabled() • initialize($name, $url) • fetchFile() • * curlCopy() • uploadCurlCallback($ch, $data) **User:** User() • load() • loadFromId() • saveToCache() • :: newFromName($name, $validate) • :: newFromId($id) • :: newFromConfirmationCode($code) • :: newFromSession() • :: newFromRow($row) • :: whoIs($id) • :: whoIsReal($id) • :: idFromName($name) • :: isIP($name) • :: isValidUserName($name) • :: isUsableName($name) • :: isCreatableName($name) • isValidPassword($password) • :: isValidEmailAddr($addr) • :: getCanonicalName($name, $validate) • :: edits($uid) • :: randomPassword() • loadDefaults($name) • SetupSession() • * loadFromSession() • loadFromDatabase() • loadFromRow($row) • loadGroups() • clearInstanceCache($reloadFrom) • :: getDefaultOptions() • :: getDefaultOption($opt) • :: getToggles() • getBlockedStatus($bFromSlave) • inSorbsBlacklist($ip) • inDnsBlacklist($ip, $base) • isPingLimitable() • pingLimiter($action) • isBlocked($bFromSlave) // hacked from false due to horrible probs on site • isBlockedFrom($title, $bFromSlave) • blockedBy() • blockedFor() • isBlockedGlobally($ip) • isLocked() • isHidden() • getId() • setId($v) • getName() • setName($str) • getTitleKey() • getNewtalk() • getNewMessageLinks() • checkNewtalk($field, $id, $fromMaster) • updateNewtalk($field, $id) • deleteNewtalk($field, $id) • setNewtalk($val) • * :: newTouchedTimestamp() • * clearSharedCache() • invalidateCache() • validateCache($timestamp) • getTouched() • setPassword($str) • setInternalPassword($str) • getToken() • setToken($token) • setCookiePassword($str) • setNewpassword($str, $throttle) • isPasswordReminderThrottled() • getEmail() • getEmailAuthenticationTimestamp() • setEmail($str) • getRealName() • setRealName($str) • getOption($oname, $defaultOverride) • getBoolOption($oname) • getIntOption($oname, $defaultOverride) • setOption($oname, $val) • getDatePreference() • getRights() • getGroups() • getEffectiveGroups($recache) • getEditCount() • addGroup($group) • removeGroup($group) • isLoggedIn() • isAnon() • isBot() • isAllowed($action) • useRCPatrol() • useNPPatrol() • isWatched($title) • addWatch($title) • removeWatch($title) • clearNotification(&$title) • clearAllNotifications($currentUser) • encodeOptions() • decodeOptions($str) • # setCookie($name, $value, $exp) • # clearCookie($name) • setCookies() • logout() • doLogout() • saveSettings() • idForName() • addToDatabase() • spreadBlock() • getPageRenderingHash() • isBlockedFromCreateAccount() • isBlockedFromEmailuser() • isAllowedToCreateAccount() • setLoaded($loaded) • getUserPage() • getTalkPage() • getMaxID() • isNewbie() • isActiveEditor() • checkPassword($password) • checkTemporaryPassword($plaintext) • editToken($salt) • generateToken($salt) • matchEditToken($val, $salt) • matchEditTokenNoSuffix($val, $salt) • sendConfirmationMail() • sendMail($subject, $body, $from , $replyto) • confirmationToken(&$expiration) • confirmationTokenUrl($token) • invalidationTokenUrl($token) • # getTokenUrl($page, $token) • confirmEmail() • invalidateEmail() • setEmailAuthenticationTimestamp($timestamp) • canSendEmail() • canReceiveEmail() • isEmailConfirmed() • isEmailConfirmationPending() • getRegistration() • ::

getGroupPermissions($groups) • :: getGroupsWithPermission($role) • :: getGroupName($group) • :: getGroupMember($group) • :: getAllGroups() • :: getAllRights() • :: getImplicitGroups() • :: getGroupPage($group) • :: makeGroupLinkHTML($group, $text) • :: makeGroupLinkWiki($group, $text) • incEditCount() • :: getRightDescription($right) • :: oldCrypt($password, $userId) • :: crypt($password, $salt) • :: comparePasswords($hash, $password, $userId) • addNewUserLogEntry($byEmail) • addNewUserLogEntryAutoCreate() **UserArray:** :: newFromResult($res) • # :: newFromResult_internal($res) **UserArrayFromResult:** __construct($res) • # setCurrent($row) • count() • current() • key() • next() • rewind() • valid() **UserMailer:** # :: sendWithPear($mailer, $dest, $headers, $body) • :: send($to, $from, $subject, $body, $replyto, $contentType) • :: errorHandler($code, $string) • :: rfc822Phrase($phrase) **UserRightsProxy:** * __construct($db, $database, $name, $id) • :: validDatabase($database) • :: whoIs($database, $id) • :: newFromId($database, $id) • :: newFromName($database, $name) • * :: newFromLookup($database, $field, $value) • :: getDB($database) • getId() • isAnon() • getName() • getUserPage() • getGroups() • addGroup($group) • removeGroup($group) • invalidateCache() **WatchedItem:** :: fromUserTitle($user, $title) • isWatched() • addWatch() • removeWatch() • :: duplicateEntries($ot, $nt) • * :: doDuplicateEntries($ot, $nt) **WatchlistEditor:** execute($user, $output, $request, $mode) • * checkToken($request, $user) • * extractTitles($list) • * showTitles($titles, $output, $skin) • * countWatchlist($user) • * getWatchlist($user) • * getWatchlistInfo($user) • * showItemCount($output, $user) • * clearWatchlist($user) • * watchTitles($titles, $user) • * unwatchTitles($titles, $user) • * showNormalForm($output, $user) • * getNamespaceHeading($namespace) • * buildRemoveLine($title, $redirect, $skin) • showRawForm($output, $user) • :: getMode($request, $par) • :: buildTools($skin) **WebRequest:** __construct() • interpolateTitle() • * extractTitle($path, $bases, $key) • checkMagicQuotes() • normalizeUnicode($data) • getGPCVal($arr, $name, $default) • getVal($name, $default) • setVal($key, $value) • getArray($name, $default) • getIntArray($name, $default) • getInt($name, $default) • getIntOrNull($name) • getBool($name, $default) • getCheck($name) • getText($name, $default) • getValues() • wasPosted() • checkSessionCookie() • getRequestURL() • getFullRequestURL() • appendQuery($query) • escapeAppendQuery($query) • appendQueryValue($key, $value, $onlyquery) • appendQueryArray($array, $onlyquery) • getLimitOffset($deflimit , $optionname) • getFileTempname($key) • getFileSize($key) • getUploadError($key) • getFileName($key) • response() • getHeader($name) • getSessionData($key) • setSessionData($key, $data) **WebResponse:** header($string, $replace) • setcookie($name, $value, $expire) **WikiError:** __construct($message) • getMessage() • toString() • :: isError($object) **WikiErrorMsg:** WikiErrorMsg($message/*, ... */) **WikiExporter:** setOutputSink(&$sink) • openStream() • closeStream() • allPages() • pagesByRange($start, $end) • pageByTitle($title) • pageByName($name) • pagesByName($names) • allLogs() • logsByRange($start, $end) • # do_list_authors($page , $revision , $cond) • # dumpFrom($cond) • # outputPageStream($resultset) • # outputLogStream($resultset) **WikiImporter:** __construct($source) • throwXmlError($err) • doImport() • debug($data) • notice($data) • setDebug($debug) • setPageCallback($callback) • setPageOutCallback($callback) • setRevisionCallback($callback) • setUploadCallback($callback) • setLogItemCallback($callback) • setTargetNamespace($namespace) • importRevision($revision) • importLogItem($rev) • importUpload($revision) • debugRevisionHandler(&$revision) • pageCallback($title) • pageOutCallback($title, $origTitle, $revisionCount, $successCount) • donothing($parser, $x, $y) • in_start($parser, $name, $attribs) • in_mediawiki($parser, $name, $attribs) • out_mediawiki($parser, $name) • in_siteinfo($parser, $name, $attribs) • out_siteinfo($parser, $name) • in_page($parser, $name, $attribs) • out_page($parser, $name) • in_nothing($parser, $name, $attribs) • char_append($parser, $data) • out_append($parser, $name) • in_revision($parser, $name, $attribs) • out_revision($parser, $name) • in_logitem($parser, $name, $attribs) • out_logitem($parser, $name) • in_upload($parser, $name, $attribs) • out_upload($parser, $name) • in_contributor($parser, $name, $attribs) • out_contributor($parser, $name) • * push($name) • * pop() • * parentTag() **WikiRevision:** setTitle($title) • setID($id) • setTimestamp($ts) • setUsername($user) • setUserIP($ip) • setText($text) • setComment($text) • setMinor($minor) • setSrc($src) • setFilename($filename) • setSize($size) • setType($type) • setAction($action) • setParams($params) • getTitle() • getID() • getTimestamp() • getUser() • getText() • getComment() • getMinor() • getSrc() • getFilename() • getSize() • getType() • getAction() • getParams() • importOldRevision() • importLogItem() • importUpload() • downloadSource() **WikiXmlError:** WikiXmlError($parser, $message , $context , $offset) • getMessage() • _extractContext($context, $offset) **XCacheBagOStuff:** get($key) • set($key, $value, $expire) • delete($key, $time) **Xml:** :: element($element, $attribs , $contents , $allowShortTag) • :: expandAttributes($attribs) • :: openElement($element, $attribs) • :: closeElement($element) • :: tags($element, $attribs , $contents) • :: namespaceSelector($selected , $all , $element_name , $label) • :: monthSelector($selected , $allmonths , $id) • :: languageSelector($selected, $customisedOnly) • :: attrib($name, $present) • :: label($label, $id) • :: listDropDown($name, $list , $other , $selected , $class , $tabindex) • :: escapeJsString($string) • :: encodeJsVar($value) • :: isWellFormed($text) • :: isWellFormedXmlFragment($text) • :: escapeTagsOnly($in) • :: buildForm($fields, $submitLabel) **XmlDumpWriter:** schemaVersion() • openStream() • siteInfo() • sitename() • generator() • homelink() • caseSetting() • namespaces() • closeStream() • openPage($row) • closePage() • writeRevision($row) • writeLogItem($row) • writeTimestamp($timestamp) • writeContributor($id, $text) • writeUploads($row) • writeUpload($file) **XmlSelect:** __construct($name , $id , $default) • setDefault($default) • setAttribute($name, $value) • addOption($name, $value) • getHTML() **XmlTypeCheck:** __construct($file, $softNamespaces) • getRootElement() • * run($fname) • * elementOpen($parser, $name, $attribs) **ZhClient:** ZhClient($host, $port) • isconnected() • connect() • query($request) • convert($text, $tolang) • convertToAllVariants($text) • segment($text) • close()