Hooks

Note: Hook names are case-insensitive. Depending on your MindTouch version, different hooks and arguments may be available.

Trunk reference

Find all the latest PHP hooks in deki_plugin.php from MindTouch's trunk SVN branch.

https://svn.mindtouch.com/source/public/dekiwiki/trunk/web/deki/plugins/deki_plugin.php

Minneopa 9.08

Special

```
/**
  * General special namespace handler
  * @param $pageName - requested special page name, e.g. Special:Foo => 'Foo'
  * @return N/A
  */
const SPECIAL_PAGE = 'Special:';

/**
  * Application special pages
  * Special pages are a class of plugins. They are only loaded when a request to
Special:
  * is made. Each special page has the same arguments passed in.
  *
  * @param $pageName - requested page name
  * @param &$pageTitle - page title to display
  * @param &$html - page html to display
  * @param &$subhtml - page subhtml to display
  * @return N/A
  */
const SPECIAL_ABOUT = 'Special:About';
const SPECIAL_ADMIN = 'Special:Admin';
const SPECIAL_ARTICLE = 'Special:Article';
const SPECIAL_CONTRIBUTIONS = 'Special:Contributions';
const SPECIAL_EVENTS = 'Special:Events';
const SPECIAL_LIST_RSS = 'Special:ListRss';
const SPECIAL_LIST_USERS = 'Special:Listusers';
```

```
const SPECIAL_PACKAGE = 'Special:Package';
const SPECIAL_PAGE_ALERTS = 'Special:PageAlerts';
const SPECIAL_PAGE_PROPERTIES = 'Special:PageProperties';
const SPECIAL_PAGE_RESTRICT = 'Special:PageRestrictions';
const SPECIAL_PAGE_EMAIL = 'Special:PageEmail';
const SPECIAL_POPULAR_PAGES = 'Special:PopularPages';
const SPECIAL_RECENT_CHANGES = 'Special:Recentchanges';
const SPECIAL_SEARCH = 'Special:Search';
const SPECIAL_SITEMAP = 'Special:Sitemap';
/**
  * Triggered when Special: is requested
  * @return N/A
  */
const SPECIAL_SPECIAL_PAGES = 'Special:SpecialPages';
const SPECIAL_TAGS = 'Special:Tags';
const SPECIAL_USER_BAN = 'Special:UserBan';
const SPECIAL_USER_LOGIN = 'Special:UserLogin';
const SPECIAL_USER_LOGOUT = 'Special:UserLogout';
const SPECIAL_USER_PASSWORD = 'Special:UserPassword';
const SPECIAL_USER_REGISTRATION = 'Special:UserRegistration';
const SPECIAL_USER_PREFERENCES = 'Special:UserPreferences';
const SPECIAL_WATCH_LIST = 'Special:Watchlist';
const SPECIAL_WATCHED_PAGES = 'Special:Watchedpages';
```

Page

```
/**
  * Triggered when any page is saved.
  * @note guerrics: Consider moving to a notification hook?
  *
  * @param Article $Article - article that was just saved
  * @return N/A
  */
const PAGE_SAVE = 'Page:Save';

/**
  * Control the comments html display
  *
  * @param Title $Title - title corresponding to the rendered article
  * @param string &$pluginHtml - comments html
  * @return HANDLED_HALT to halt internal html generation
  */
const PAGE_RENDER_COMMENTS = 'Page:RenderComments';

/**
  * Control the files html display
  *
  * @param Title $Title - title corresponding to the rendered article
  * @param string &$pluginHtml - files html
  * @return HANDLED_HALT to halt internal html generation
  */
const PAGE_RENDER_FILES = 'Page:RenderFiles';
```

```
/**
  * Control the image gallery html display
  *
  * @param Title $Title - title corresponding to the rendered article
  * @param string &$pluginHtml - image gallery html
  * @return HANDLED_HALT to halt internal html generation
  */
const PAGE_RENDER_IMAGES = 'Page:RenderImages';

/**
  * Control the tag html display
  *
  * @param Title $Title - title corresponding to the rendered article
  * @param string &$pluginHtml - tags html
  * @return HANDLED_HALT to halt internal html generation
  */
const PAGE_RENDER_TAGS = 'Page:RenderTags';
```

Editor

```
/**
  * Triggered when the default editor makes a request for javascript files.
  *
  * @param Article $Article - article content that will be loaded
  * @param array &$editorScripts - url's to javascript files to load
  * @return anything but UNHANDLED or UNREGISTERED will halt the default editor loading
  */
const EDITOR_LOAD = 'Editor:Load';

/**
  * Triggered after the editor form is posted back.
  * @param string $textareaContents -unparsed/raw contents
  * @param string &$contents - parsed contents of the page/section
  * @param string &$displayTitle - parsed display title of the page being edited
  * @param int &$section - section being edited
  * @param string &$summary - edit summary
  * @param int &$pageId - destination page id
  */
const EDITOR_PROCESS_FORM = 'Editor:ProcessForm';

/**
  * Triggered when the default editor begins rendering the editor form html.
  *
  * @param string &$textareaContents - unencoded/raw textarea contents
  * @param string $forEdit - source contents
  * @param string $displayTitle - source display title
  * @param int $section - source section
  * @param int $pageId - source page id
  * @param string &$prependHtml - html inside form tag before textarea
  * @param string &$appendHtml - html inside form tag after textarea
  * @return N/A
  */
const EDITOR_FORM = 'Editor:Form';
```

Main

```
/**
  * Main Hooks
  *
  * These hooks are trigger for core application actions and general hooks fired from
index.php
  */

/**
  * Triggered early in the initialization of index.php. Allows custom url rewriting or
entry point handling.
  *
  * @param WebRequest $Request
  * @param Title $Title
  * @return N/A
  */
const MAIN_PROCESS_TITLE = 'Main:ProcessTitle';

/**
  * Triggered after the page has been rendered. Allows final modification to the output
buffer.
  *
  * @param OutputPage $Out - Output buffer
  * @return N/A
  */
const MAIN_PROCESS_OUTPUT = 'Main:ProcessOutput';

/**
  * Triggered when the user's information has been cleared by the user registration
process.
  *
  * string &$username
  * string &$password
  * string &$email
  * @return HANDLED_HALT to halt the internal user creation process, still fires
CREATE_USER_COMPLETED
  */
const MAIN_CREATE_USER = 'Main:CreateUser';

/**
  * Newly created user object is passed in for any additional setup. If this
  * call is halted then no redirect gets issued and it is up to the plugin
  * to determine where to redirect the user after creation.
  *
  * @param DekiUser $User
  * @return HANDLED_HALT to halt the default redirect behavior. Has no effect if
CREATE_USER was already halted.
  */
const MAIN_CREATE_USER_COMPLETE = 'Main:CreateUserComplete';

/**
  * Triggered when a user posts login information to Special:UserLogin.
```

```
  *
  * @param string $username
  * @param string $password
  * @param int $authId
  * @return HANDLED_HALT to halt internal login processes
  */
const MAIN_LOGIN = 'Main:Login';

/**
  * User should be fully logged in at this call.
  *
  * @param DekiUser $User - newly logged in user
  * @return N/A
  */
const MAIN_LOGIN_COMPLETE = 'Main:LoginComplete';

/**
  * Triggered when a user visits Special:UserLogin and is already logged in.
  *
  * @param DekiUser $User - currently logged in user
  * @param string $html - html to output or prepend to the default output depending on
return value.
  * @return HANDLED_HALT to avoid outputting the internal display HTML.
  */
const MAIN_LOGIN_REFRESH = 'Main:LoginRefresh';

/**
  * Triggered before the current user is logged out.
  *
  * @param DekiUser $User - user that is attempting to logout
  * @return HANDLED_HALT to halt internal logout processes
  */
const MAIN_LOGOUT = 'Main:Logout';

/**
  * Triggered after the current user has been logged out.
  *
  * @param DekiUser $User - user that completed the logout process
  * @return N/A
  */
const MAIN_LOGOUT_COMPLETE = 'Main:LogoutComplete';

/**
  * Triggered when a user posts a comment to a page, action=comment
  *
  * @param Title $Title - page title receiving the comment POST data
  * @return HANDLED_HALT to avoid the internal comment processing
  */
const MAIN_ACTION_COMMENT = 'Main:ActionComment';

/**
  * Default action handler, hook to handle unknown actions.
  *
  * @param string $action - name of the requested action, e.g.
```

```
comment,newpage,mycustomaction
  * @param Title $Title - currently requested page title
  * @param Article $Article - currently requested article
  */
const MAIN_ACTION = 'Main:Action';
```

Skin

```
/**
  * Skin Hooks
  *
  * These are skin components. You won't be able to set headers in any
  * skin hooks since the headers will already be sent.
  */

/**
  * Nav pane hook allows a custom navigation module to be loaded.
  *
  * @param Title $Title - corresponds to current page
  * @param string &$navText - text to be injected into the 'sitenavtext' template
variable
  * @return N/A
  */
const SKIN_NAVIGATION_PANE = 'Skin:NavigationPane';

/**
  * Hook is called before the skin html gets rendered which allows certain template
variables
  * to be overriden or new variables to be added. Hook is dependent upon template
creators to
  * add to the template. See fiesta.php or deuce.php for details on adding this hook.
  *
  * @param SkinTemplate &$template - instance of the current template being rendered
  * @return N/A
  */
const SKIN_OVERRIDE_VARIABLES = 'Skin:OverrideVariables';

/**
  * Notify Hooks
  *
  * Triggered when the user is being notified via email.
  */

/**
  * Triggered when an email is being sent.
  *
  * @param string &$to - receiptiant email address
  * @param strubg &$from - sender's email address
  * @param string &$subject - email subject
  * @param string &$body - email body
  * @param string &$bodyHtml - alternate html email body
  *
  * @return HANDLED_HALT to halt internal email sending. Will not trigger email
```

```
complete hook unless
  * the plugin fires it manually.
  */
const NOTIFY_EMAIL = 'Notify:Email';

/**
  * Triggered after an email has been sent successfully.
  *
  * @param string $to
  * @param string $from
  * @param string $subject
  * @param string &$body
  * @param string &$body_html));
  *
  * @return N/A
  */
const NOTIFY_EMAIL_COMPLETE = 'Notify:EmailComplete';
```

Data

```
/**
  * Data Hooks
  *
  * Used for obtaining information from active plugins.
  */

/**
  * Retrieve a plugin generated list of pages that the current user can access.
  *
  * @param array &$pages - list of pages
  * @note $pages is a 2 dimensional array with a format like
  *    $pages['Page Name'] = array('name' => 'Page Name', 'href' => 'http://foo/
Special:PageName');
  * @return N/A
  */
const DATA_GET_SPECIAL_PAGES = 'Data:GetSpecialPages';
```