Moodle Output APIs

Page API

Think of the global variable `$PAGE` as the container for the page's content – usually HTML.

$PAGE

- The `$PAGE` variable is an instance of the `moodle_page` class defined in the ***/lib/pagelib.php*** file. It is created very early in the request cycle in **setup.php**.
- It includes the theme and layout, the JavaScript and CSS requirements, titles, headings, navigation position, and things like blocks to display. In other words, the scaffolding for the actual output.
- The page URL and the context are always required to display output.

$PAGE (cont)

- The `$PAGE->set_url()` sets the page url – usually you pass a `moodle_url()` object to this function
- The `$PAGE->set_context()` sets the context – as the contexts discussed previously There are some other methods, including `$PAGE->set_course()` and `$PAGE->set_cm` (course module). Both of these automatically set the page context as well.
- The `$PAGE->set_pagelayout()` method sets the page layout
- The `$PAGE->set_title()` method sets the page's title, i.e. the text in the <title> HTML tags
- The `$PAGE->set_heading()` sets the title for the page. The display of this is theme-dependant.

$PAGE (cont)

You can also get page information such as:

- `$PAGE->course`
- `$PAGE->cm` (course module)
- `$PAGE->title`
- `$PAGE->headerprinted` - determine if the header has been printed
- `$PAGE->navbar` and `$PAGE->navigation` which let you access the navigation components.

The Output API

This API handles most of the Moodle output. It provides several generic functions and encompasses additional output components, JavaScript output, and renderers and templates.

Output API Files

There are several files in the **/lib** folder relating to the Moodle output, many of which you will not need. The ones of note are:

- **outputcomponents.php**: Classes representing HTML elements, used by global `$OUTPUT` methods includes the `html_writer` class
- **weblib.php**: Library of functions for web output library of all general-purpose Moodle PHP functions and constants that produce HTML output
- **outputrenderers.php**: Classes for rendering HTML output, including the renderer base class that is available via the `$OUTPUT` global

outputcomponents.php

This file defines a number of interfaces and classes representing HTML elements extensively used by global `$OUTPUT` methods, but available for use generally.

- Interfaces: `renderable{}` and `templatable{}`

outputcomponents.php (cont)

- **Pictures/Icons** classes: `user_picture{}`, `help_icon{}`, `pix_icon_font{}`, `pix_icon_fontawesome{}`, `pix_icon{}`, `image_icon{}`, `pix_emoticon{}`
- **HTML Bars** classes: `progress_bar{}`, `paging_bar{}`, `initials_bar{}`
- **Tab** classes: `tabtree{}`, `tabobject{}`
- Other classes for blocks, action menus, preferences and others.

outputcomponents.php (cont)

- Form related classes:
  - `file_picker{}`: A class representing a file picker as per the ‘File API’ and ‘Form API’ to be discussed
  - `single_button{}`: A class representing a simple form with only one button
  - `single_select{}`: Simple form with just one select field that gets submitted automatically
  - `url_select{}`: Simple URL selection widget

outputcomponents.php (cont)

- The HTML and JavaScript classes
  - `js_writer{}`: Simple Javascript output class – YUI JavaScript only; methods include `function_call()`, `object_init()`, `set_variable()`
  - `html_writer{}`: Simple html output class, 28 methods including `tag()`, `start_tag()`, `end_tag()`, `link()`, etc
  - `html_table{}`: Holds all the information required to render a HTML table; outputted by `html_writer::table()`
  - `html_table_cell{}`: Component representing a table cell
  - `html_table_row{}`: Component representing a table row

weblib.php

This library defines several classes

- The base `progress_trace` and several extensions to output progress status (such as `html_progress_trace`)
- The `moodle_url` class provides useful functions to work with Moodle URLs which we discuss further in this course

weblib.php (cont)

- Functions :
  - `s()` – adds quotes to HTML characters
  - `p()` – prints quotes to HTML characters
  - `format_text()` – when given text in a variety of format coding, this function returns the text as safe HTML
  - `clean_text()` – cleans raw text, removing nasties
  - `get_local_referer()` – returns the cleaned local URL of the `HTTP_REFERER`, less the URL query string parameters if required
  - `html_to_text()` – when given HTML text, it converts it into plain text
  - `is_https()` – check if the site is running under SSL

weblib.php (cont)

- `is_in_popup()` and `close_window()` – check if in a pop-up and close the pop-up window respectively
- `me()` – returns the name of the current script, WITH the query string portion and `qualified_me()`, which guesses the full URL of the current script
- `notice()` – prints a message and exits
- `redirect()` – redirects the user to another page, after printing a notice
- `strip_links()` – when given a string, it replaces all <a>.* by .* and returns the string
- `text_to_html()` – when given plain text, it converts it into HTML as neatly as possible
- `validate_email()` – validates an email address

Links

For more information on the Output API, please see the ‘Output API’ page

https://docs.moodle.org/dev/Output_API

and the ‘Output Functions’ page

https://docs.moodle.org/dev/Output_functions

Renderers & Templates

- When a plugin needs to display visual output within a page or layout, the recommended method is to use a renderer

  https://docs.moodle.org/dev/Output_renderers

- Renderers were introduced in Moodle 2.0 with the introduction of the global `$OUTPUT` object which points to Moodle's core renderer, can be used for other non-display content generation such as emails and data exports. They may also use templates to generate output.
- *It is worth noting that theme designers can also define renderers and templates as well as override plugin renderers.*

Templates

- Moodle 2.9 introduced the concept of templates.
- Templates are written as `mustache` (http://mustache.github.io/mustache.5.html) templates, a 'Logic-less' templating system. Templates should be saved in the plugin's **templates** folder and must have a ***.mustache*** extension.
- Renderers use templates via the `render_from_template()` method in the form:

  ```
  $this->output->render_from_template($templatename, $data)
  ```

  where `$data` is the context referred to in the Mustache's documentation and is the data to be used by the template.
- Visit the Moodle 'Templates' documentation page (https://docs.moodle.org/dev/Templates) for information on the how and the where, and coding styles. In this course, templates are not covered in further detail.

Renderers

- Renderers are defined in ***renderer.php*** files in the plugin's top folder or, since Moodle 2.8, in the **classes*/*** subdirectory, following the Moodle's auto-loading rules.
- The plugin's renderer class will normally extend the core `plugin_renderer_base` class. Code in the renderer should not refer to the two globals `$PAGE` or `$OUTPUT`, but access the core functionality via `$this->page` and `$this->output` variables.
- The renderer is obtained by the `$PAGE->get_renderer()` method and there is a sort of unwritten rule about naming the renderer variable `$output` (lower case) to show its relation to the global `$OUTPUT`.