Cómo hacer que tus plugins funcionen en Moodle Mobile

Juan Leyva & Dani Palou
Mobile Team

#MootES18

How it works

- Moodle plugins can extend different areas in the app with just PHP server side code and Ionic 3 markup (custom html elements that are called components) using a set of custom Ionic directives and components.
- Developing Web Services functions is not required.
- Javascript may not be required for simple plugins.
- Dynamic behaviour is achieved thanks to components and directives.

How it works

Developers need to:

- Create a db/mobile.php file describing the plugin supported features and including configuration options.
- Develop functions in a reserved namespace that will return full mobile pages rendered at server level (using the app component - html attributes)

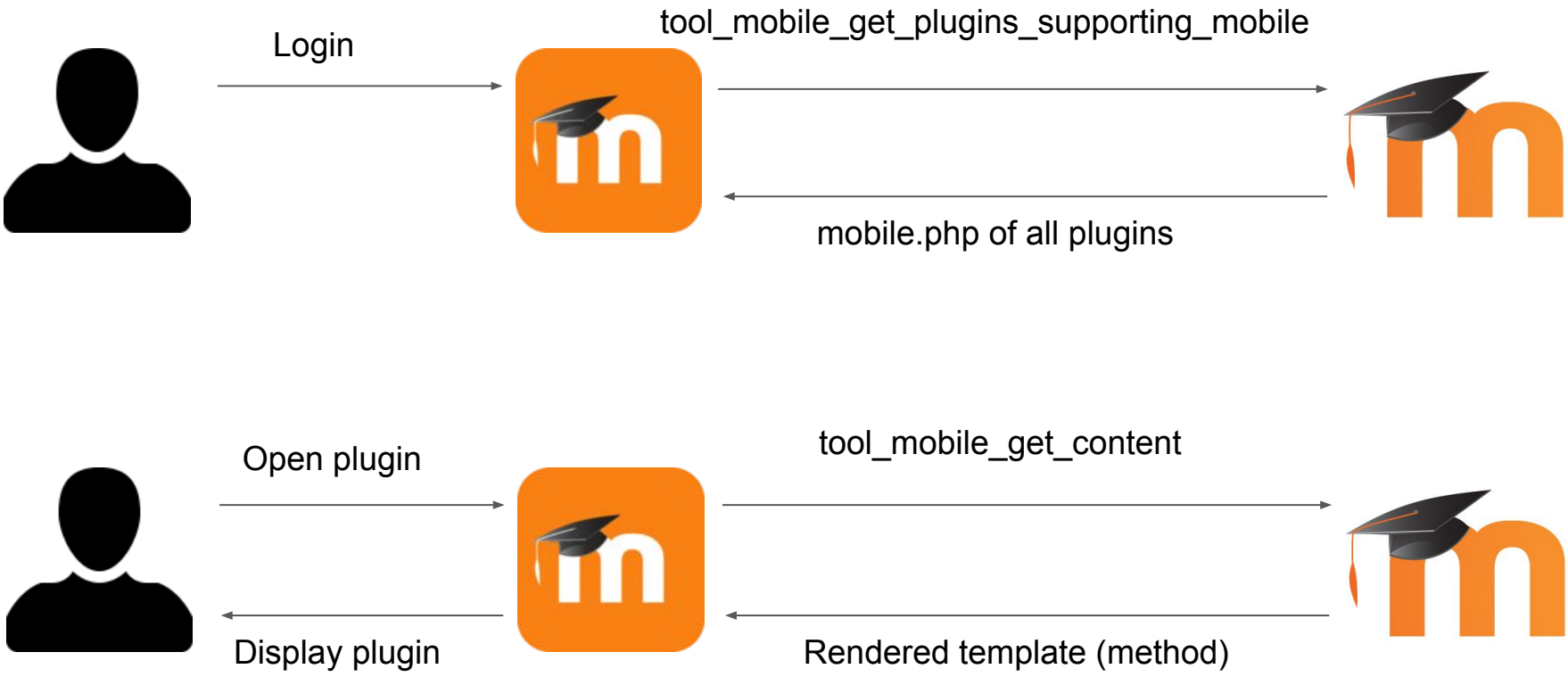
Types of plugins

1. Templates are generated and downloaded when the user opens the plugin. E.g.: activity modules, course formats, main menu options, etc.

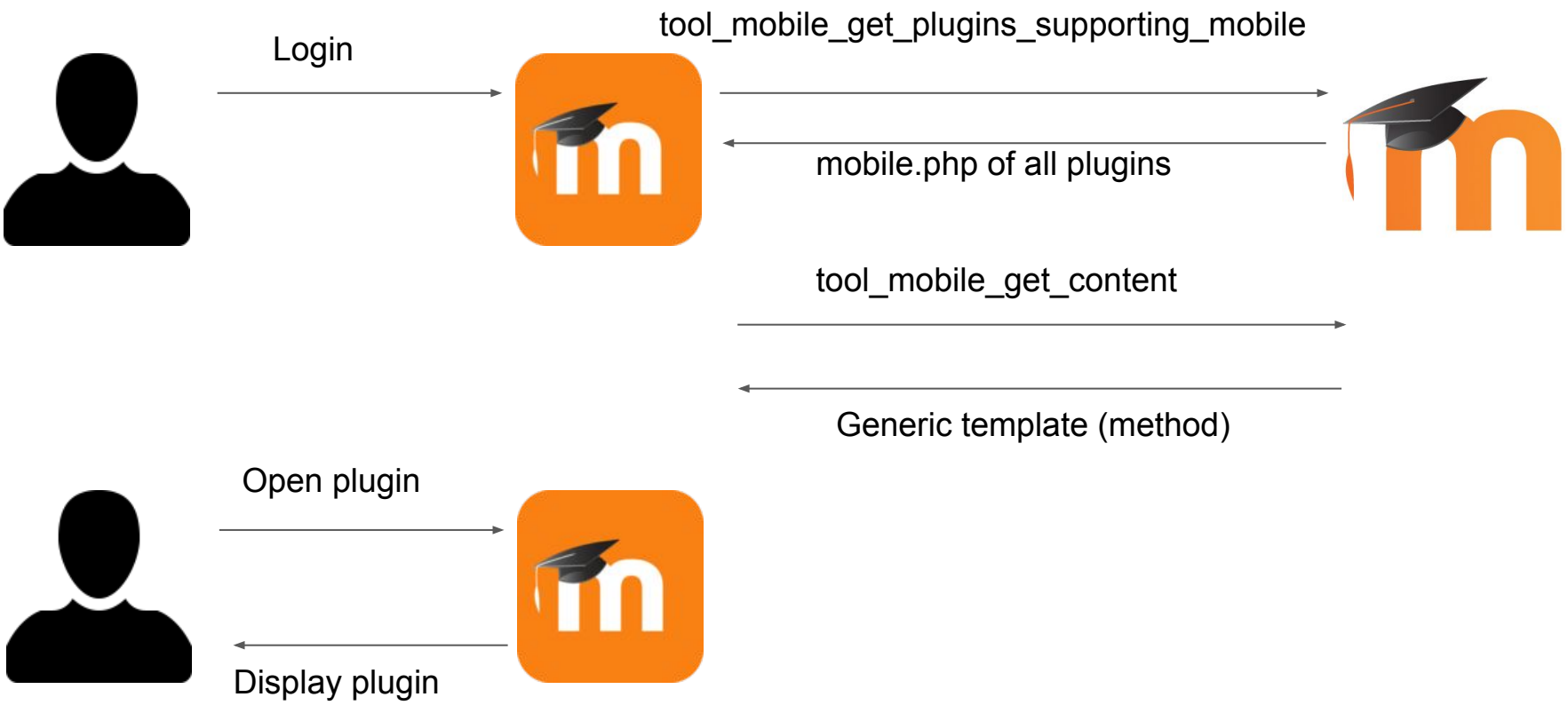
2. A generic template is generated and downloaded on login, and then it uses JS data supplied by the app to build the view every time it's needed. E.g.: question types, user profile fields, etc.

3. Pure JS plugins.

Templates downloaded when requested



Templates built with JS



Step by step example

Certificate module

- Simple activity module that displays the certificate issued for the current user along with the list of the dates of previously issued certificates.
- It stores in the course log that the user viewed a certificate.
- This module also works offline: when the user downloads the course or activity, the data is pre-fetched and can be viewed offline.

Step by step example

- Step 1. Creating the db/mobile.php file
- Step 2. Creating the main function
- Step 3. Creating the template for the main function
- Step 4. Adding an additional page
- Step 5. Plugin webservices, if included

https://docs.moodle.org/dev/Mobile_support_for_plugins#Step_by_step_example

Mobile.php explained

- In the Step by Step section we learned about some of the existing options for handlers configuration.

- But there are more options that are common and others delegate specific.

- All the options are documented here:

https://docs.moodle.org/dev/Mobile_support_for_plugins#Mobile.php_supported_options

Delegates explained

- A plugin (or a link to it) can be shown in several places.
- The app has one delegate per each place a plugin can be shown, so each plugin can register itself to any set of delegates and be displayed/linked in several places in the app.
- Some advanced delegates require JavaScript to be supported.
- Depending on the type of plugin being implemented you will have to use different delegates (and more than one at the same time)

https://docs.moodle.org/dev/Mobile_support_for_plugins#Delegates

Components and directives

- A component (represented as an HTML tag) is used to add custom elements to the app. Example of components are: ion-list, ion-item, core-search-box
- A directive (represented as an HTML attribute) allows you to extend a piece of HTML with additional information or functionality. Example of directives are: core-auto-focus, *ngIf, *ngFor
- The Mobile app uses Angular, Ionic and custom components and directives.

Components and directives

- Angular directives: https://angular.io/api?type=directive
- Ionic components: https://ionicframework.com/docs/
- Custom components: https://docs.moodle.org/dev/Mobile_support_for_plugins#Custom_core_components_and_directives
- There is also a set of custom components and directives for supporting plugins: https://docs.moodle.org/dev/Mobile_support_for_plugins#Specific_component_and_directives_for_plugins

Advanced features

- For those advanced plugins (or plugins requiring specific functionality) there are several advanced features that will allow you to:
  - Display the plugin only if certain conditions are met.
  - Use persistent data between WS calls.
  - Access to certain content related JavaScript functions available in templates.
  - Inject custom JavaScript code to initialize the plugin.
  - Override handlers implementation via the JavaScript API (not fully supported)

https://docs.moodle.org/dev/Mobile_support_for_plugins#Advanced_features

mobile@moodle.com
@moodler