

## Contents

Einrichten einer Moodle-Instanz .....	2
Installation von Plugins.....	6
Code Plugin.....	7
version.php.....	7
question.php .....	8
questiontype.php .....	8
edit_code_form.php.....	8
renderer.php.....	8
DB .....	8
Lokalisierung.....	10
Workflow: Parameter hinzufügen .....	10
Monaco-Editor.....	11
Language Server .....	12
Links.....	12

## Einrichten einer Moodle-Instanz

Um Moodle zu hosten ist ein Web- und Datenbankserver notwendig. Da Moodle auf PHP basiert ist ein simpler Webserver wie Apache ausreichend. Als Datenbank ist MySQL oder MariaDB zu empfehlen. Ein einfacher Weg eine Entwicklungsumgebung aufzusetzen ist XAMPP: <https://www.apachefriends.org/>

Moodle verlangt folgende Änderungen in der php.ini

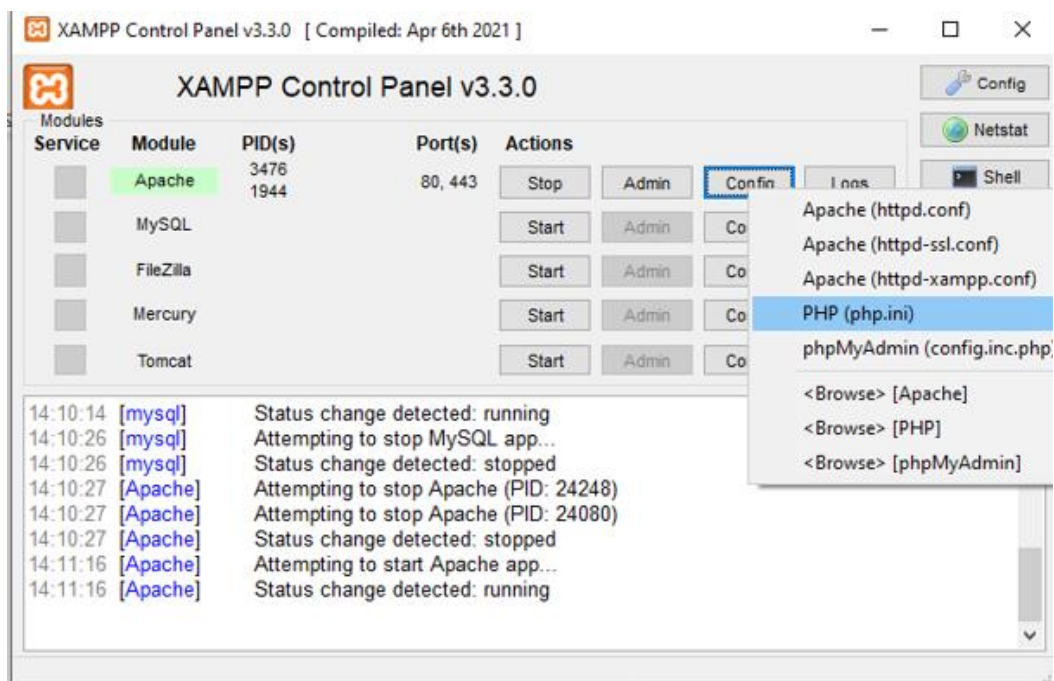
```
max_input_vars = 50000

max_execution_time = 120

post_max_size = 256M

upload_max_filesize = 256M
```

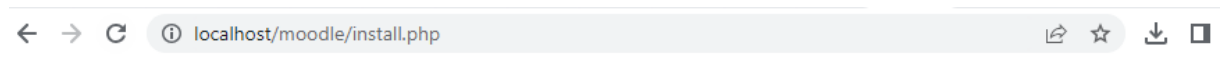
Die php.ini kann unter XAMPP wie folgt geöffnet werden:



Nach der Installation von XAMPP muss das Moodle-Repository geklont werden (<https://github.com/moodle/moodle>). Ein git clone Kommando, welches eine Stable-Branch klonet, ist: `git clone --branch MOODLE_401_STABLE git://git.moodle.org/moodle.git`

Für eine Stabile Entwicklungsumgebung ist die kombination aus Moodle\_401 und XAMPP 7.4.33 empfohlen (<https://sourceforge.net/projects/xampp/files/XAMPP%20Windows/7.4.33/>).

Der Moodle-Ordner ist entweder in den Ordner xampp/htdocs zu legen, oder ein Link ist zwischen dem Moodle-Ordner und dem htdocs-Ordner zu erstellen. Nach dem Starten des Apache- und des DB-Servers kann die Moodle-Installation mit dem Besuchen von localhost/moodle begonnen werden.



**Deprecated:** Using \${var} in strings is deprecated, use {\${var}} instead in C:\xampp\htdocs\moodle\lib\phpminimumversionlib.php

**Deprecated:** Using \${var} in strings is deprecated, use {\${var}} instead in C:\xampp\htdocs\moodle\lib\phpminimumversionlib.php

## Installation

### Language

#### Choose a language

Please choose a language for the installation. This language will also be used as the default language for the site, though it may be changed later.

Language

English (en)

Next »



XAMPP verwendet MariaDB. Unter XAMPP sieht die Datenbankeinrichtung so aus:

# Installation

## Database

### Database settings

#### MariaDB (native/mariadb)

The database is where most of the Moodle settings and data are stored and must be configured here.

The database name, username, and password are required fields; table prefix is optional.

The database name may contain only alphanumeric characters, dollar (\$) and underscore (\_).

If the database currently does not exist, and the user you specify has permission, Moodle will attempt to create a new database with the correct permissions and settings.

This driver is not compatible with legacy MyISAM engine.

Database host	<input type="text" value="localhost"/>
Database name	<input type="text" value="moodle"/>
Database user	<input type="text" value="root"/>
Database password	<input type="password"/>
Tables prefix	<input type="text" value="mdl_"/>
Database port	<input type="text" value="3306"/>

[< Previous](#) [Next >](#)

Etwaige Probleme werden auf der folgenden Seite angezeigt. Wenn Plugins fehlen, sind die entsprechenden Zeilen in der `php.ini` auszukommentieren.

## Installation - Moodle 4.1.4+ (Build: 20230616)

### Moodle 4.1.4+ (Build: 20230616)

For information about this version of Moodle, please see the online [Release Notes](#)

#### Server checks

Name	Information	Report	Plugin	Status
php_extension	sodium	should be installed and enabled for best results <a href="#">↗</a>		
php_setting	opcache.enable	PHP setting should be changed. <a href="#">↗</a> PHP opcode caching improves performance and lowers memory requirements, OPcache extension is recommended and fully supported.		
unicode		must be installed and enabled <a href="#">↗</a>		
database	mariadb (5.5.5-10.4.27-MariaDB)	version 10.4 is required and you are running 10.4.27 <a href="#">↗</a>		
php		version 7.4.0 is required and you are running 7.4.33 <a href="#">↗</a>		
pcrunicode		should be installed and enabled for best results <a href="#">↗</a>		
php_extension	iconv	must be installed and enabled <a href="#">↗</a>		
php_extension	mbstring	must be installed and enabled <a href="#">↗</a>		
php_extension	curl	must be installed and enabled <a href="#">↗</a>		

Danach führt Moodle die Erstellung und Initialisierung der Datenbank durch, dass kann einige Minuten dauern. Danach sind einige abschließende Einstellungen vorzunehmen.

## Installation

On this page you should configure your main administrator account which will have complete control over the site. Make sure you give it a secure username and password as well as a valid email address. You can create more admin accounts later on.

Expand all

### ▼ General

Username

?

admin

Choose an authentication method

?

Manual accounts

The password must have at least 8 characters, at least 1 digit(s), at least 1 lower case letter(s), at least 1 upper case letter(s), at least 1 special character(s) such as as \*, -, or #

New password

!

?

.....

☐ Force password change 

?

First name

!

Admin

Last name

!

User

Email address

!

p61580@fhooe.at

Email visibility

?

Visible to everyone

City/town

Select a country

Select a country...

Timezone

Server timezone (Europe/Berlin)

## Installation

## New settings - Site home settings

Full site name  
fullname

Short name for site (eg single word)  
shortname

Site home summary  
summary

↕

A ▼

B

I

☰

☷

☰

☷

🔗

🔄

😊

🖼️

H-P

🌐

⚙️

This summary can be displayed on the site home using the course/site summary block.

Danach ist Moodle installiert und kann verwendet werden.


test
Home
Dashboard
My courses
Site administration

🔔
🗨️
AU
Edit mode

Welcome, Admin! 🙌

Timeline

Next 7 days
Sort by dates
Search by activity type or name



No in-progress courses

## Installation von Plugins

Plugins können mittels „Site administration“ -> „Plugins“ -> „Install plugins“ installiert werden. Dabei werden sie gezippt in das Feld gezogen. Moodle platziert dann automatisch das Plugin im entsprechenden Ordner.

### Plugin installer

Install plugins from the Moodle plugins directory ?

#### ▼ Install plugin from ZIP file ?

ZIP package



Choose a file...



You can drag and drop files here to add them.

Accepted file types:

Archive (ZIP) .zip

[Show more](#)

## Code Plugin

Die Grundlagen der Plugin-Entwicklung sind nicht teil dieses Dokuments. Tutorials und „Kurse“ sind in unter Moodle-Academy zu finden. Für dieses Plugin sind die Moodle-Developer Kurse am relevantesten. <https://moodle.academy/>



Das Moodle-Plugin besteht folgenden Dateien:

- version.php
- question.php
- questiontype.php
- edit\_code\_form.php
- renderer.php
- DB-Dateien
- Lokalisierung

### version.php

In dieser Datei wird die Version des Plugins, im Format YYYYMMDDXX angeben. Moodle berücksichtigt erst Änderungen in der Datenbank, wenn die Versionsnummer erhöht wird.

### question.php

Die Datei question.php definiert die Instanz einer Frage. Dabei werden die verschiedenen fragenspezifischen Konfigurationsparameter als Variablen definiert.

### questiontype.php

Hier werden Metainformationen über den Typ gespeichert. Dabei kann für bestimmte Parameter definiert werden welche Werte sie annehmen können. Es kann auch für Parameter ein Default-Wert gesetzt werden. Weiters sind hier die Funktionen zum Speichern einer Frageninstanz und zum Erstellen einer solchen Instanz.

### edit\_code\_form.php

Diese Datei beschreibt die Eingabemaske der definierten Parameter einer Frage. Dabei wird mittels der Moodle-Forms-API ein Formular erstellt.

<https://moodledev.io/docs/apis/subsystems/form>

### renderer.php

Hier wird die Frage während eines Tests dargestellt. Dabei werden die relevanten Parameter in unsichtbaren p-Tags geschrieben. Die Initialisierung des Monaco-Editors geschieht mit der Ausgabe eines script-Tags. Dieses script-Tag zeigt auf die Datei app.js im Dist-Ordner des Plugins.

### DB

In Moodle sind die Tabellen eines Plugins in der Datei install.xml definiert. Entwickler müssen diese Datei nicht per Hand verändern, sondern können Moodle-Dev-Tools benutzen:

Site administration → XMLDB editor



## testmoodle

Search

General Users Courses Grades Plugins Appearance Server Reports Development

### Main view

[Reserved words] [Doc] [Reconcile XMLDB files] [Check indexes] [Check defaults] [Check integers] [Check foreign keys]

admin/tool/admin_presets/db	[Create] [Load] [Edit] [Save] [Doc] [XML] [Revert] [Unload] [Delete]
admin/tool/analytics/db	[Create] [Load] [Edit] [Save] [Doc] [XML] [Revert] [Unload] [Delete]
admin/tool/availabilityconditions/db	[Create] [Load] [Edit] [Save] [Doc] [XML] [Revert] [Unload] [Delete]
admin/tool/behat/db	[Create] [Load] [Edit] [Save] [Doc] [XML] [Revert] [Unload] [Delete]
admin/tool/brickfield/db	[Create] [Load] [Edit] [Save] [Doc] [XML] [Revert] [Unload] [Delete]
admin/tool/capability/db	[Create] [Load] [Edit] [Save] [Doc] [XML] [Revert] [Unload] [Delete]
admin/tool/cohortroles/db	[Create] [Load] [Edit] [Save] [Doc] [XML] [Revert] [Unload] [Delete]
admin/tool/componentlibrarv/db	[Create] [Load] [Edit] [Save] [Doc] [XML] [Revert] [Unload] [Delete]

Danach das Plugin suchen. In diesem Fall ist es: question/type/code

question/type/calculatedsimple/db	[Create] [Load] [Edit] [Save] [Doc] [XML] [Revert] [Unload] [Delete]
question/type/code/db	[Create] [Load] [Edit] [Save] [Doc] [XML] [Revert] [Unload] [Delete]
question/type/ddimageortext/db	[Create] [Load] [Edit] [Save] [Doc] [XML] [Revert] [Unload] [Delete]

Danach: Load + Edit

## testmoodle

Search

General Users Courses Grades Plugins Appearance Server Reports Development

### Edit XML file

Path: question/type/code/db

Version: 20230417

Comment: XMLDB file for Moodle question/type/code

Change

[View original] [View edited] [New table] [New table from MySQL] [View SQL code] [View PHP code] [Back to main]

### Tables

qtype\_code\_options [Edit] [Delete] [XML]

Nun könne die Tabellen des Plugins bearbeitet werden. Nachdem die Änderungen in install.xml gespeichert wurden, muss die Datei upgrade.php erweitert werden.

Unter dem Punkt „View PHP code“ kann dieser Upgrade-Code generiert werden. Dieser muss dann nur noch in die Datei kopiert werden. Damit Moodle diese Änderung erkennt, muss die Versionsnummer in der Datei `version.php` angehoben werden.

**Table: qtype\_code\_options**

Select action:  Select field/key/index:

```
if ($oldversion < XXXXXXXXXX) {

    // Define field id to be added to qtype_code_options.
    $table = new xmldb_table('qtype_code_options');
    $field = new xmldb_field('id', XMLDB_TYPE_INTEGER, '10', null, XMLDB_NOTNULL,
XMLDB_SEQUENCE, null, null);

    // Conditionally launch add field id.
    if (!$dbman->field_exists($table, $field)) {
        $dbman->add_field($table, $field);
    }

    // Code savepoint reached.
    upgrade_plugin_savepoint(true, XXXXXXXXXX, 'qtype', 'code');
}
```

## Lokalisierung

In Moodle-Code kann mittels `get_string('ID', 'qtype_code')` auf lokalisierte Strings zugegriffen werden. Diese Strings befinden sich in der `qtype_code.php` Datei. Für jede unterstützte Sprache gibt es eine solche Datei. Der Inhalt einer dieser Dateien sieht so aus: `$string['answertext'] = 'Answer text';`

## Workflow: Parameter hinzufügen

Um einen Parameter zu der Code-Frage hinzuzufügen, dabei müssen folgende Änderungen vorgenommen werden:

- `question.php`

In der Datei `question.php` muss der gewünschte Parameter als Variable hinzugefügt werden.

- `questiontype.php`

In `questiontype.php` müssen die Funktionen `save_question_options` und `initialise_question_instance` erweitert werden. Die Erweiterung ist in der Funktion `save_question_options` als `$options->newVar = initialValue` und `$options->newVar = $formdata->newVar` durchzuführen. Die Funktion `initialise_question_instance` ist mit

`$question->newVar = $questiondata->options->newVar` zu erweitern.

- `edit_code_form.php`

Hier sind die Funktionen `definition_inner` und `data_preprocessing` zu erweitern. In `definition_inner` ist, mithilfe der Form-API, die Eingabemaske zu erweitern. In `data_preprocessing` ist `$question->newVar = $question->options->newVar` hinzuzufügen.

- `install.xml`

Diese Datei wird mittels des XMLDB-Editors erweitert.

- `upgrade.php`

Diese Datei muss mit dem PHP-Code vom XMLDB-Editor erweitert werden.

- `renderer.php`

Hier werden die HTML-Tags für den Monaco-Editor generiert. In der Funktion `formulation_and_controls` kann mittels `$qa-> get_question()->newVar` auf den neuen Parameter zugegriffen werden.

- `version.php`

Wie schon erwähnt, muss die Version erhöht werden damit Moodle die Datenbank aktualisiert.

## Monaco-Editor

Der Monaco-Editor ist mithilfe einer separaten Datei eingebunden. Um einen neuen LS einzubinden, muss folgendes IF-Konstrukt erweitert werden:

```
if (language == 'java') {
    var ext = '.java';
} else if (language == 'csharp') {
    var ext = '.cs';
} else if (language == 'pascal') {
    var ext = '.pas';
} else if (language == 'python') {
    var ext = '.py';
} else if (language == 'c') {
    var ext = '.c';
} else if (language == 'cpp') {
    var ext = '.cpp'
}
```

Die Parameter der Frage werden wie folgt ausgelesen:

```
var mID = document.getElementById('mId').textContent;
```

Wenn der Code geändert wurde, muss mittels dem Command `npx webpack` der Code neu verpackt werden. Dabei entsteht der Ordner `dist`. Dieser Ordner ist im Root-Verzeichnis (`code/`) des Plugins abzulegen. Der alte Ordner ist vorher zu löschen. `webpack` ist bereits im `node_modules` Ordner vorhanden und muss nicht installiert werden. Falls nicht vorhanden, muss `npm` installiert werden.

## Language Server

Die LS werden mittels Docker realisiert. Dabei hostet ein Docker einen Node-Server. Das Moodle-Plugin verbindet sich mit diesem Server. Der Server startet danach den jeweiligen LS. Die Kommunikation über das LSP erfolgt mittels eines Websocket mit Port 3001. Der Inhalt des Editors wird mittels eines HTTP-POST übertragen. Für die Implementierung eines neuen LS sind folgende Schritte notwendig:

1. Erweitern der `index.js` Datei um die entsprechende Dateiendung
2. Erstellen eines Docker-Containers welcher die Voraussetzungen des LS mitbringt
3. Anpassen der `server.mjs` Datei, um den LS zu starten.

Der Ordner `node_modules`, welcher in allen vier Dockerordnern zu finden ist, wird vom Server benötigt und ist in die Docker-Images aufzunehmen. Da unter Linux-Dockern die, von Tools wie `apt` installierten, Node-Versionen veraltet sind, muss das Tool `nvm` verwendet werden. Eine mögliche Installation sieht so aus:

```
RUN curl -o- https://raw.githubusercontent.com/nvm-  
sh/nvm/v0.39.0/install.sh | bash  
ENV NVM_DIR="/root/.nvm"  
ENV SH_VERSION="1.38.0"  
RUN . "$NVM_DIR/nvm.sh" && nvm install 18.16.0  
RUN . "$NVM_DIR/nvm.sh" && nvm alias default stable  
  
# Add NVM binaries to the system PATH  
ENV PATH="/root/.nvm/versions/node/v18.16.0/bin:${PATH}"
```

Die bestehenden Docker sind wie folgt zu Starten:

```
docker run -d -p 3001:3001 -p 3002:3002 <name>
```

## Links

- <https://moodle.academy/>
- <https://moodledev.io/docs/apis/subsystems/form>
- [https://docs.moodle.org/dev/Question\\_types](https://docs.moodle.org/dev/Question_types)
- <https://github.com/TypeFox/monaco-languageclient>
- <https://github.com/microsoft/monaco-editor>

- <https://microsoft.github.io/monaco-editor/>
- <https://microsoft.github.io/language-server-protocol/implementors/servers/>
- <https://www.npmjs.com/package/webpack>