TECHNISCHE UNIVERSITÄT BERLIN

# QuestionPy
# Ein frei programmierbarer Fragetyp.

Martin Gauk, Jan Britz | innoCampus | MoodleMoot DACH 2025 | 4. September 2025

innoCampus

# Einleitung

- innoCampus
  - unterstützt Studierende und Lehrende bei Organisation und Durchführung von Lehrveranstaltungen und Klausuren
  - betreibt seit 2005 Moodle an der TU Berlin
  - stellt Verschiedene Werkzeuge zur digitalen Kommunikation bereit (Matrix, Zoom und BBB)

- QuestionPy
  - Förderung durch Land Berlin
  - Projektmitarbeiter: Jan Britz, Maximilian Haye, Mirko Dietrich und Martin Gauk

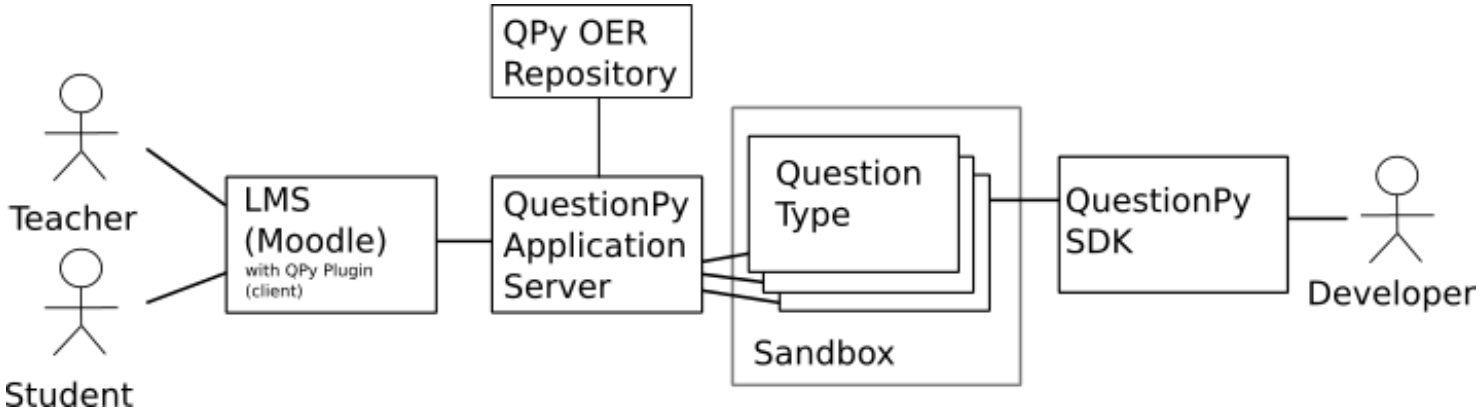
# Motivation – Welche Möglichkeiten gibt es bereits?

- Wünsche der Lehrenden nach mehr Fragemöglichkeiten
- STACK und CodeRunner ermöglichen bereits Umsetzung vieler Fragen
  - Fokussierung auf mathematische bzw. Programmier-Fragestellungen
  - noch nicht generisch genug
- einzelne Lehrende behelfen sich mit JavaScript
  - hakelige Eingabe von JS in den WYSIWYG-Feldern

# Motivation – Was wollen wir?

- Frei programmierbarer Fragetyp mit kompletter Logik zur Erzeugung des Fragetextes, Eingabefelder, der Bewertung, des Feedbacks, Musterlösung, etc.
- Können auch von den Lehrenden selbst programmiert werden
- Ähnlich zu SCORM, QTI und teilweise LTI: offline Inhalte erarbeiten und in Moodle laden
- Fragetypen können in Form von QPy-Paketen über öffentliche Repositories bereitgestellt werden
- Lehrende viel weniger abhängig von LMS-Administration
- LMS-Administration kann nicht beliebig viele Moodle-Plugins installieren
  - Wartbarkeit / Zeitaufwand
  - Sicherheit

# Aufbau – Übersicht



# Aufbau – Paket – Question-Options

- Optionen einer Frage können über Python-Klassen und -Funktionen definiert werden
- In `FormModel`-Klasse können Felder definiert werden
- Erlaubt auch Moodle-typische „hide-if“- und „disable-if“-Bedingungen
- Komplexere Validierungen mit Pydantics `field_validator`-Decorator möglich

# Aufbau – Paket – Question-Options

```python
from pydantic import field_validator, ValidationInfo

from questionpy.form import FormModel, checkbox, is_not_checked, text_area, text_input


class MyModel(FormModel):
    question = text_input(label="Question", required=True)

    with_knowledge = checkbox(
        left_label="Custom Knowledge",
        right_label=None,
        help="If selected, your custom knowledge will be used to score the answer.",
    )

    knowledge = text_area(label="Knowledge", hide_if=[is_not_checked("with_knowledge")])

    @field_validator("knowledge", mode="after")
    @classmethod
    def context_required(cls, value: str | None, validation_info: ValidationInfo):
        if validation_info.data["with_knowledge"] and not value:
            raise ValueError("Knowledge is required when 'Custom Knowledge' is checked.")
        return value
```

# Aufbau – Paket – Question-Attempt

- In `Attempt`-Klasse steckt die gesamte Logik eines Attempts
- Jinja2 als Templating-Engine
- CSS- und JS-Einbindung
  - Über `build_hooks` kann z.B. auch TS verwendet werden
- Score eines Attempts kann in `_compute_score` zurückgegeben werden
- Über verschiedene Scoring-Errors kann z.B. angegeben werden, ob eine Antwort manuell gescored werden muss

# Aufbau – Paket – Question-Attempt

```python
from questionpy import Attempt, BaseScoringState, Question, ResponseNotScorableError

from .form import MyModel


class MyAttempt(Attempt):
    scoring_state_class = MyScoringState

    def _init_attempt(self) -> None:
        self.call_js("main.js", "init")
        self.use_css("styles.css")

    def _compute_score(self) -> float:
        if not self.response or "answer" not in self.response:
            msg = "'answer' is missing"
            raise ResponseNotScorableError(msg)

        return 1 if self.response["answer"] == 42 else 0

    @property
    def formulation(self) -> str:
        return self.jinja2.get_template("formulation.xhtml.j2").render()


class MyQuestion(Question):
    attempt_class = MyAttempt

    options: MyModel
```

# Aufbau – Paket – Ordnerstruktur

```
./
├── python/
│   ├── namespace/
│   │   └── short_name/
│   │       └── ... (Python-Code)
├── templates/
│   └── ... (Jinja2-Templates)
├── js/
├── css/
├── assets/
│   ├── logo.{svg,png,jpg}
│   └── ... (Etwaige andere öffentliche Dateien)
├── locale/
└── qpy_config.yml
```

# Aufbau – SDK

```
Usage: questionpy-sdk [OPTIONS] COMMAND [ARGS]…

Options:
    -n, --no-interaction  Do not ask any interactive question.
    -v, --verbose         Use log level DEBUG.
    -h, --help            Show this message and exit.

Commands:
    create   Create new package.
    i18n     Manage translations for a package.
    package  Build package from directory SOURCE.
    repo     Repository commands.
    run      Run a package.
```

# Aufbau – Plugin

## Admin

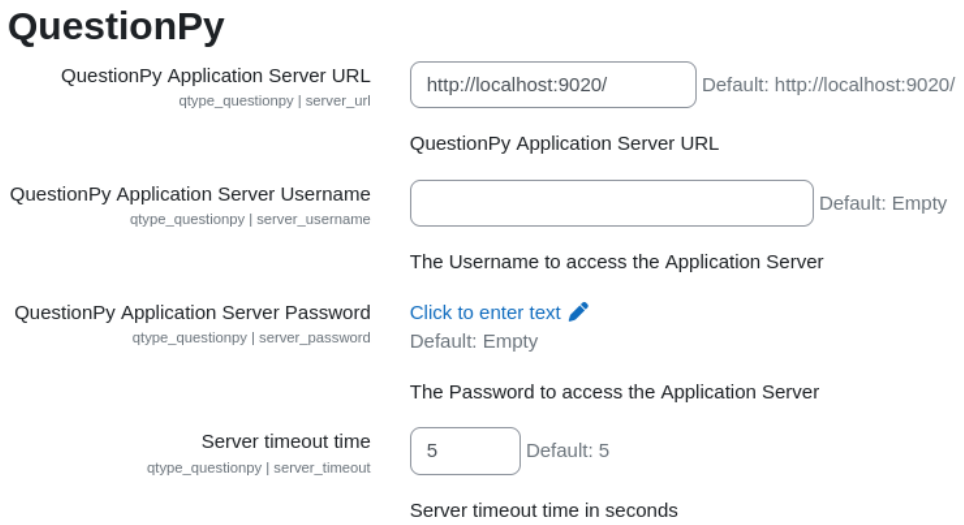
- Authentifizierung und Status-Abfrage
- Laden und Entfernen von Paketen
- Jegliche Fehler werden geloggt und sind einsehbar

## Lehrende

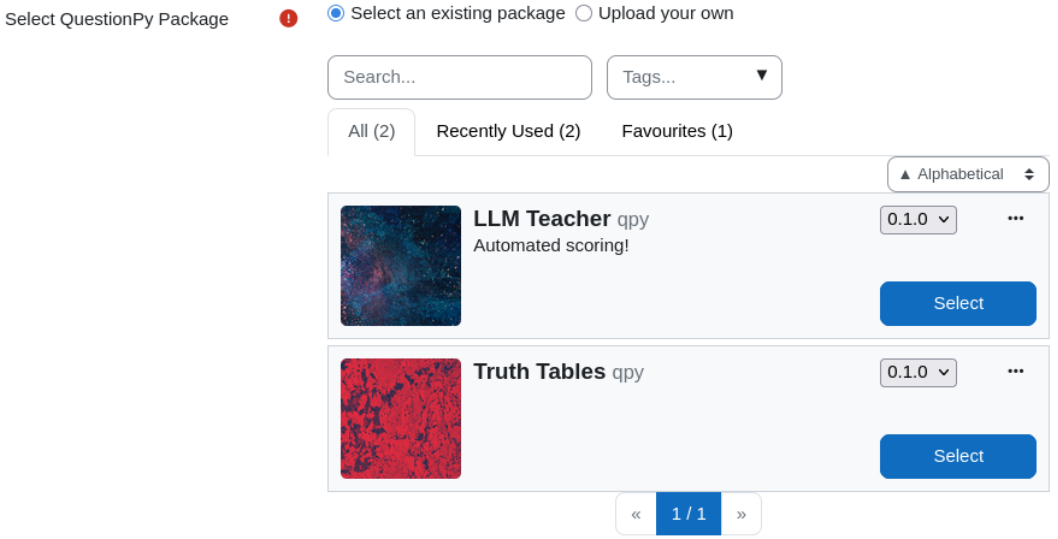
- Paketauswahl über Suchcontainer
- Verwendung eigener Pakete über Upload
- Anzeigen aller Schritte eines Attempts

# Aufbau – Plugin

## Admin-Panel




## Suchcontainer



# Aufbau - Server

- Server kann über `config.yml` konfiguriert werden
- Pakete aus mehreren Quellen bündeln
  - Lokal
  - Repository
  - LMS
- Ausführung der Pakete in wiederverwendbaren Prozessen
- Kontrolle von Paket-Berechtigungen

# Aktueller Stand

- Produktiver Einsatz ab dem Wintersemester
- Im Rahmen eines Programmier-Praktikums wurden Pakete erstellt
  - SQL-Fragetyp

# Ausblick – Welche Features sollen noch kommen?

- Ausführung in Containern
- Erweiterung der Worker-Berechtigungen
- Migration zwischen Paket-Versionen
- Moodle-Admin-Seite zum Pakete verwalten

# Ausblick – Wie geht die Entwicklung weiter?

- Jeder bei der Entwicklung helfen, da OSS
- Feature-Requests sind willkommen

# Showcase

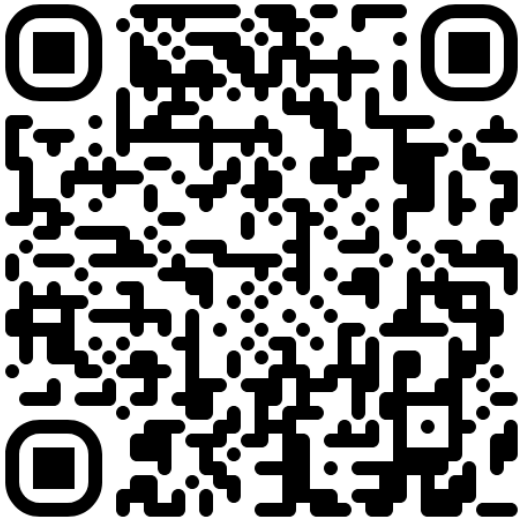
- Truthtables-Fragetyp
- JupyterLite-Fragetyp
- Subplugin für mod_assign (Prototyp um eine QPy-Frage in der Aufgaben-Aktivität inkl. Gruppenmodus einzubinden)

# Diskussions- und Frage-Runde

Hat noch jemand Fragen, Anregungen oder Kommentare?

## Links

Projektseite



GitHub