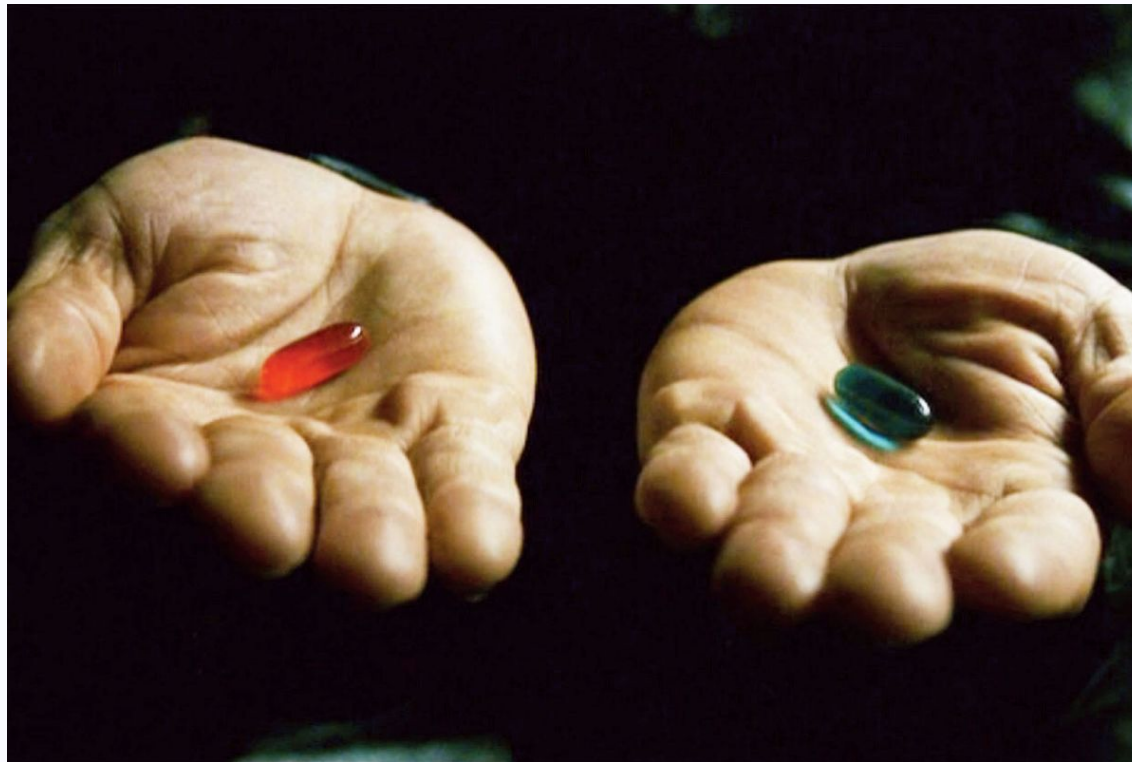


You have a choice



Remixed from material by [Ali Seyhun Saral](#) & [Philipp Chapkovski](#)

Installing oTree

- What do you need?
 - Python3 installation
 - A virtual environment (optional but **STRONGLY** recommended)

Virtual Environment

A directory that contains

- Python installation
- a number of additional packages.

Virtual environments are useful for “isolated” python developments with different packages & dependencies.

Especially important if you want to run experiments in the DICE Lab



Virtual Environment 1
myvenv1

python 3.7
otree 2.5
numpy 1.18

windows

c:\ot\myvenv1

macos/linux

/home/seyhun/ot/myvenv1



Virtual Environment 2
myvenv2

python 3.6
otree 2.3
numpy 1.0

windows

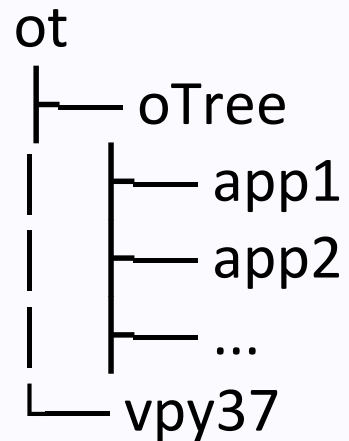
c:\old\ot\myvenv1

macos/linux

/home/seyhun/old/myvenv1

oTree Installation (a recommendation)

We would like to have this structure:



- **ot**: Parent folder for oTree project and virtualenv
- **oTree**: oTree project folder. That contains necessary files and “apps”
- **vpy37**: virtual environment. A copy of python and necessary packages (as well as otree-base)

oTree Installation

- **Create a folder for oTree**

`mkdir ot`

- **Go to the folder**

`cd ot`

- **Create the virtual environment**

`python3.7 -m venv vpy37` (vpy37 is a name I gave for “virtualenv python”)

- **Activate the virtual environment**

`source vpy37/bin/activate` (linux/macOS)

`vpy37\Scripts\activate` (windows)

`(vpy37) ~/ot>`

- **Install oTree**

`pip3 install -U "otree<5"`

Creating oTree project and app

- Create an oTree project

`otree startproject oTree`

- Go to oTree folder

`cd oTree`

- Check if oTree is running properly

`otree devserver`

- Go to <http://localhost:8000>

- Stop the server with Ctrl-C

- Create an app

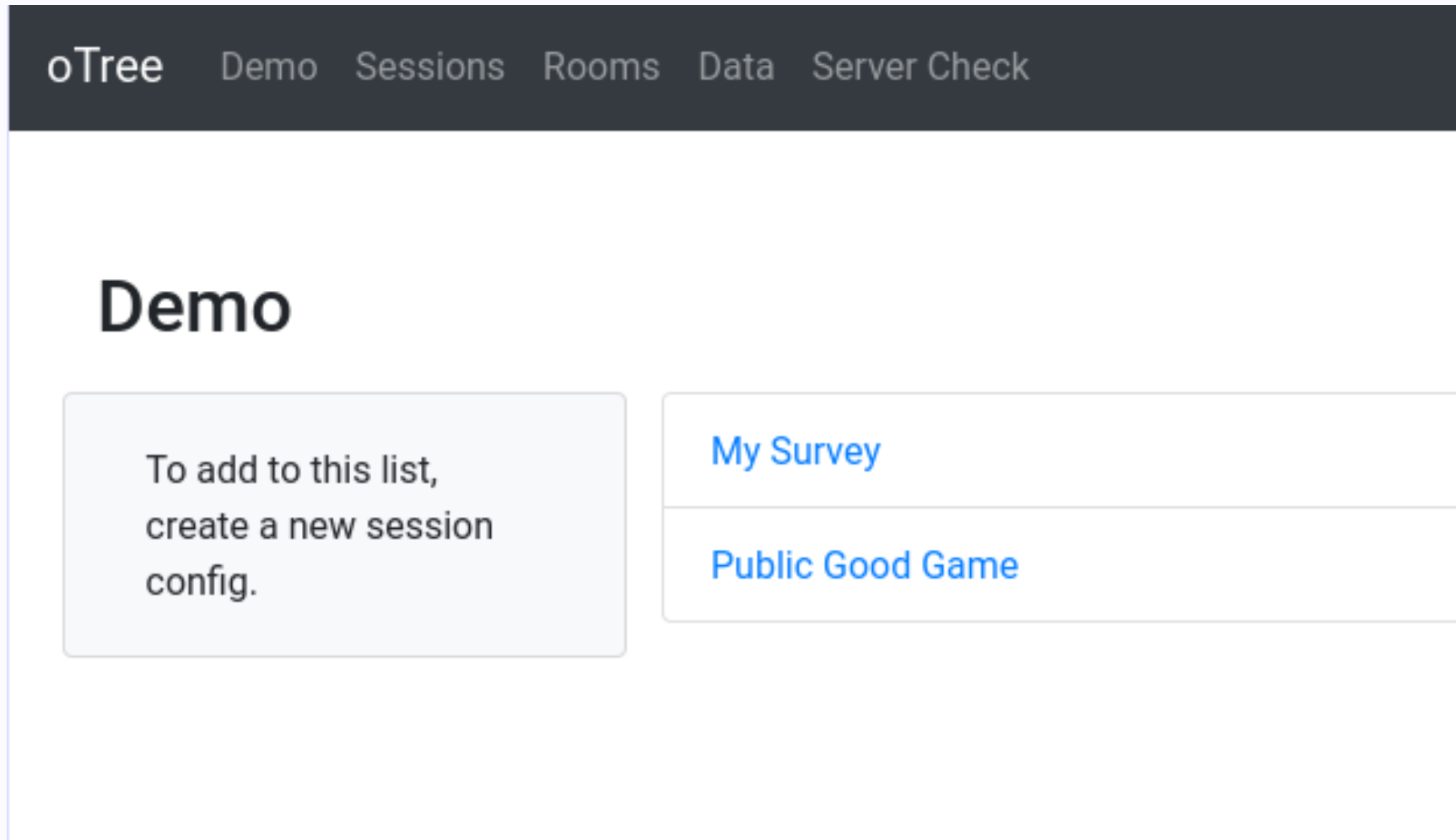
`otree startapp my_survey`

Command Line Mini Cheatsheet

| | |
|---|--|
| print working directory | cd (Windows) pwd (Linux/ MacOS) |
| list directories in the current directory | dir (Windows) ls (Linux/MacOS) |
| go to directory | cd FOLDERPATH (absolute or relative) cd c:\ot\oTree cd ot |
| create directory | mkdir FOLDERNAME |
| activate virtualenv | venvpath\Scripts\activate.bat (windows) source VENVPATH/bin/activate (linux, macos) |
| deactivate virtualenv | deactivate |
| run Otree server (developing) | otree devserver |
| create app | otree startapp APPNAME |

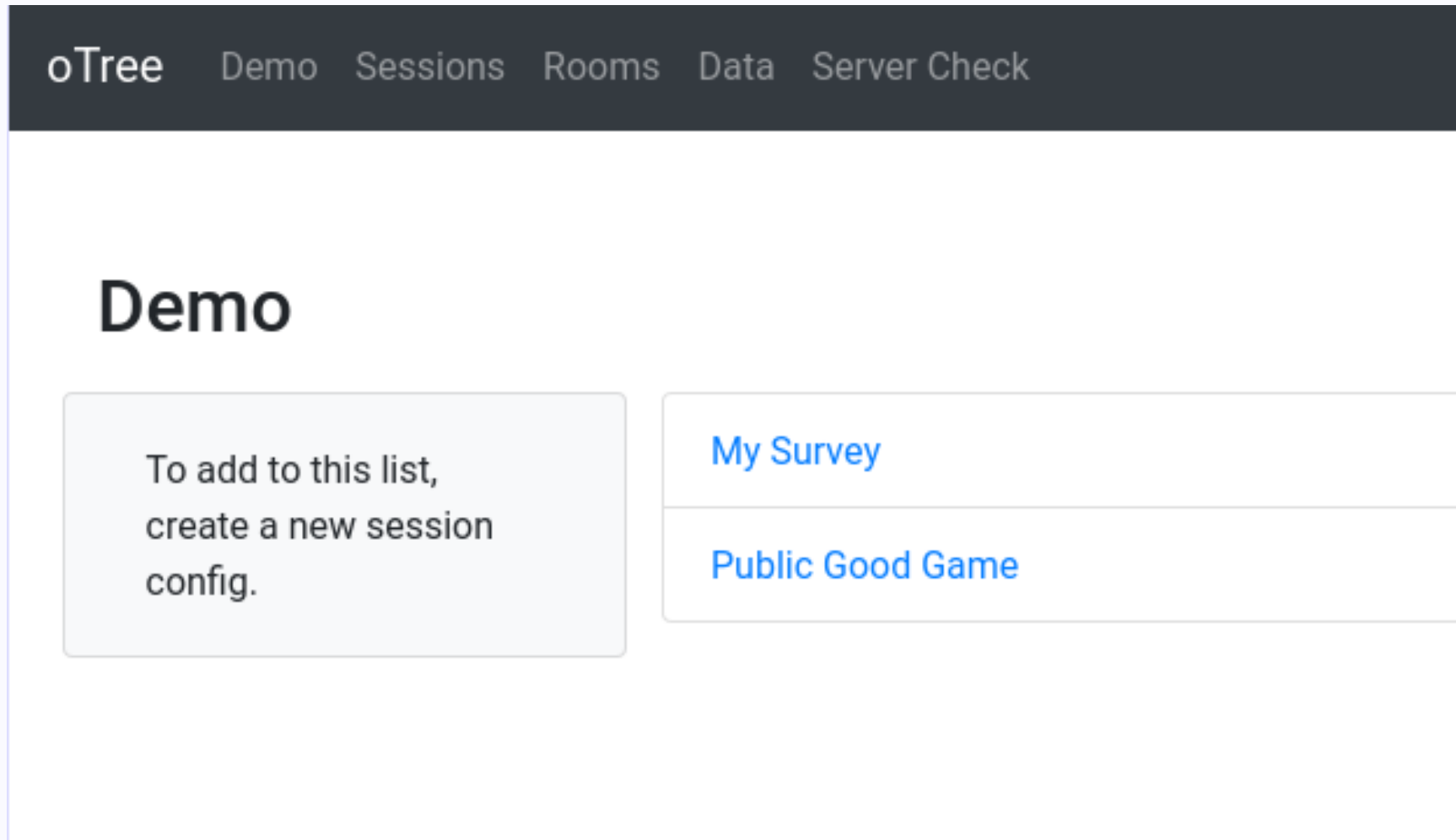
oTree Admin Interface

- <http://localhost:8000>



oTree Admin Interface

- Demo: A quick way to test experiments



oTree Admin Interface

- Session: To create and manage sessions

Create a new session

Session config:

My Survey

Number of participants

Must be a multiple of 1

Create

Configure session

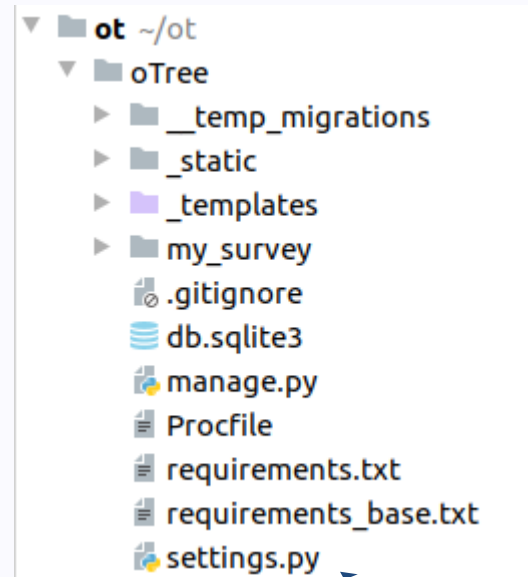
You can make more properties configurable by adding them to your session config in `settings.py`.

| | |
|-------------------------------|----------------|
| participation_fee | <div>0.0</div> |
| real_world_currency_per_point | <div>1.0</div> |

App sequence

| | |
|-----------|----------------------|
| my_survey | Your app description |
|-----------|----------------------|

oTree File Structure

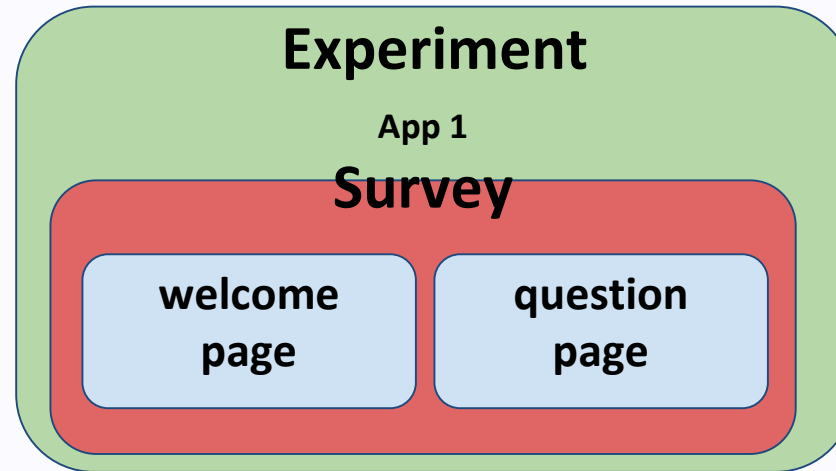


we access settings of oTree here

What is an app in oTree?

App is a basic unit of an experiment. An app consists one or more pages of an experiment.

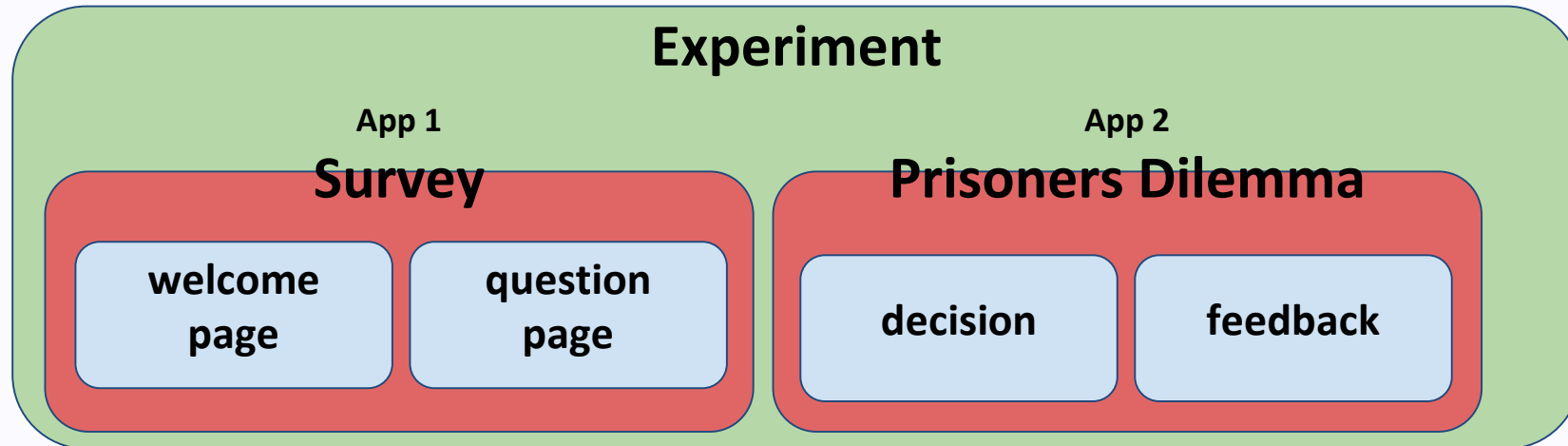
Each experiment should consist at least an app.



What is an app in oTree?

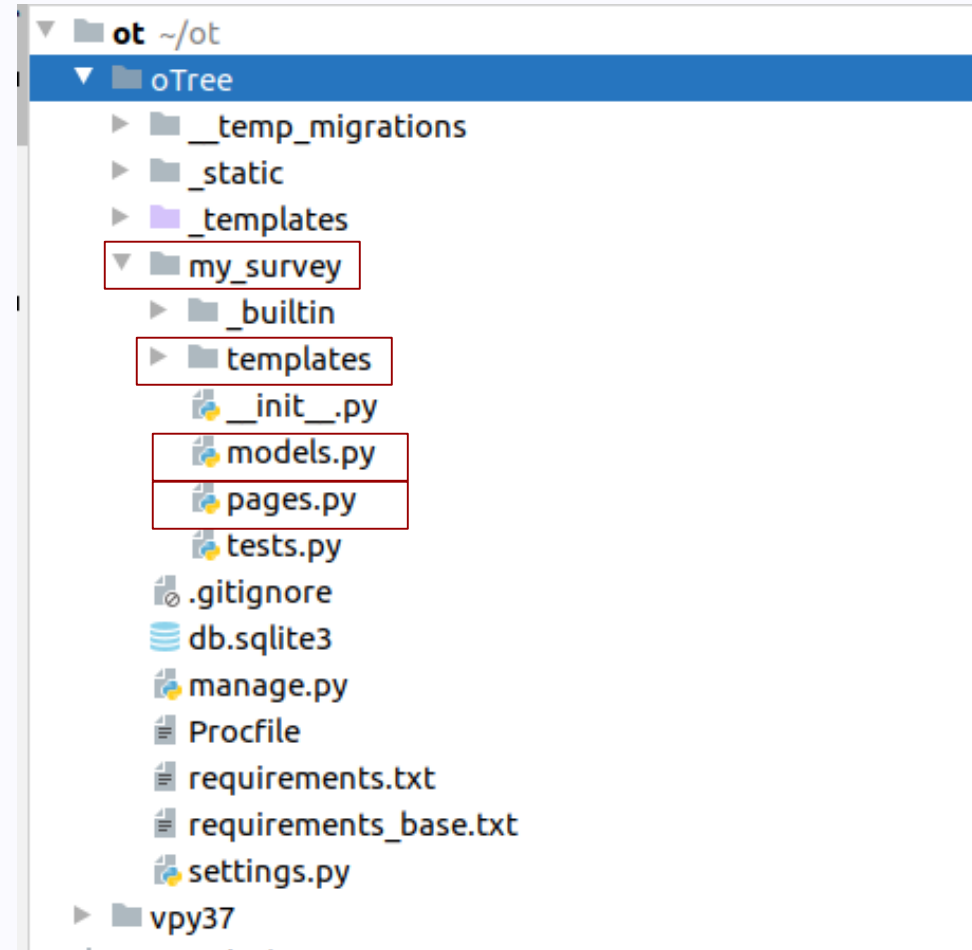
App is a basic unit of an experiment. An app consists one or more pages of an experiment.

Each experiment should consist at least an app.



Create & register an app

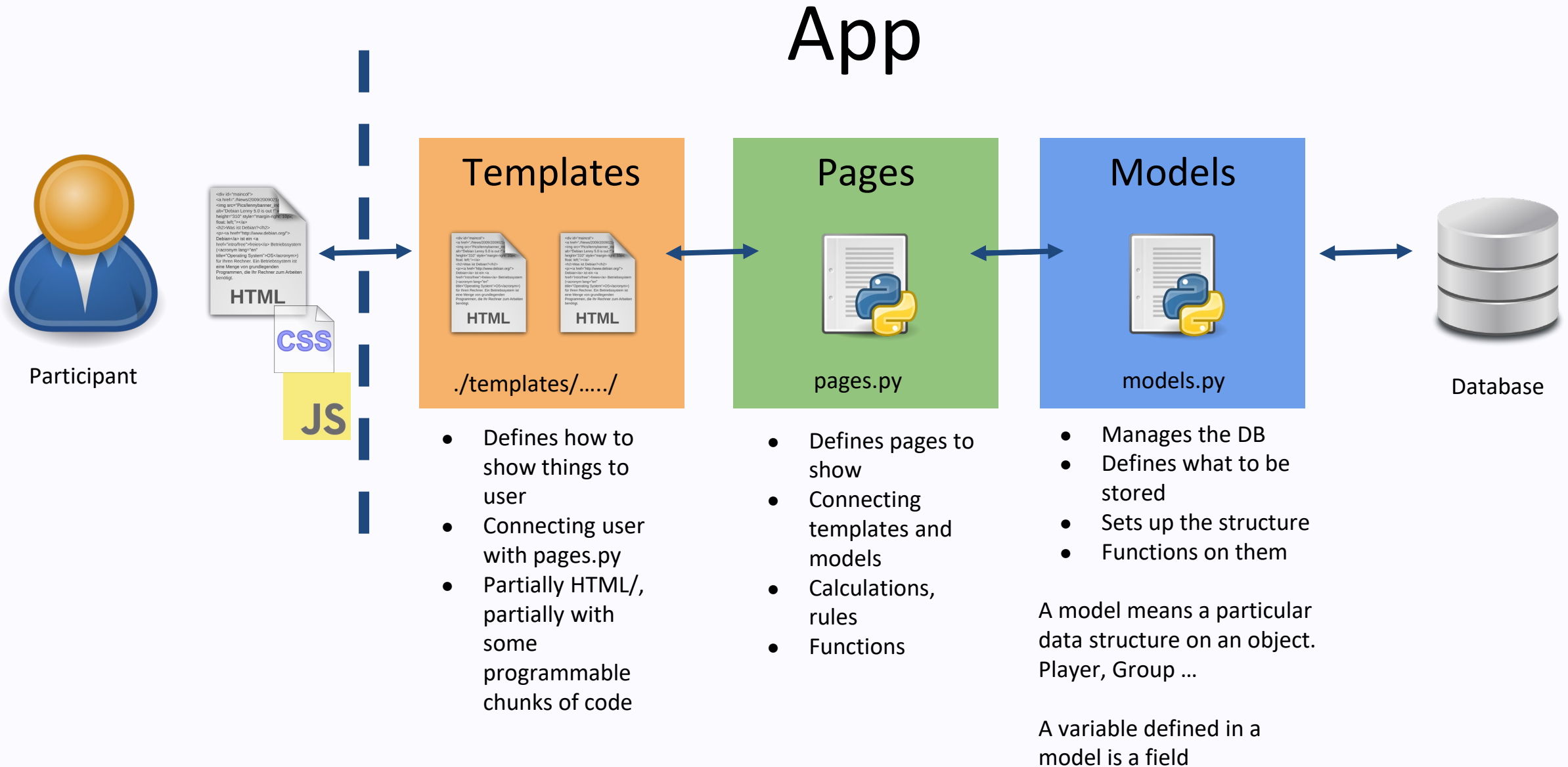
- Go to oTree project folder
cd ot/oTree
- Create the app
otree startapp my_survey



App should be added to settings.py

```
SESSION_CONFIGS = [  
    dict(  
        name='survey',  
        display_name='My Survey',  
        num_demo_participants=1,  
        app_sequence=['my_survey']  
    ),  
    dict(  
        name='pgoods',  
        display_name='Public Good Game',  
        num_demo_participants=3,  
        app_sequence=['pgoods', 'my_survey']  
    ),  
]
```

oTree (django) architecture



A simple survey

Task: Asking users their name, age and if they are student or not.

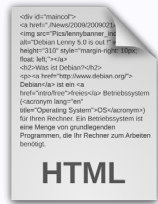
Page 0: Welcome Page

Page 1: Show the form to fill and get the data

Page 2: Show the data



Participant



HTML

Templates



./templates/...../

Templates:

Welcome
Ask
Results

Each template is a
separate file matches the
page class

Pages



pages.py

Pages:

Welcome
Ask
Results

Each page is a class

Models



models.py

Player model:

name
age
is_student

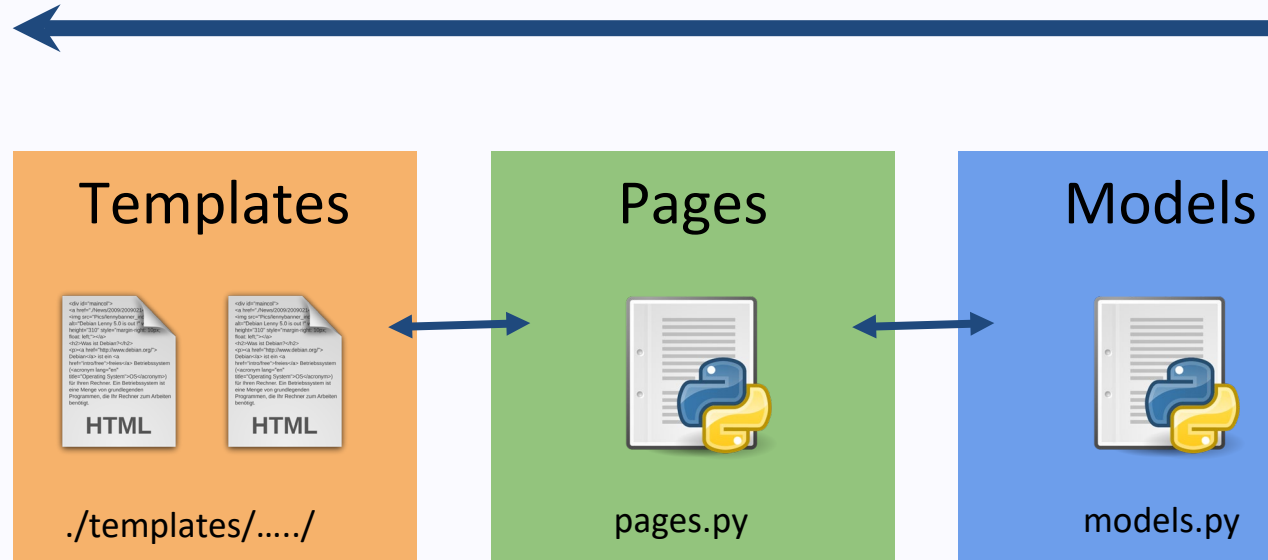
Each model is a class (Player)

Each variable is a field(column)



Database

Adding a new page



Models

Models



models.py

- Manages the DB
- Defines what to be stored
- Sets up the structure
- Functions on them
- A model means a particular data structure on an object.
Player, Group ...
- They should inherit from base models (BasePlayer, BaseGroup ...)

A variable defined in a model is a field.

```
class Player(BasePlayer):  
    somevariable = models.StringField()
```

Fields

- Fields create the data structure to record the data. They can be thought as “columns” in a spreadsheet.
- They are placed in Player or Group class in models.py
- You can create a field for form input, or to save the data without an explicit input
- They are created from built-in Models object by defining them in a Model (Group, Player)

| Field name | What for? | Example |
|-----------------|---------------------------------------|--|
| StringField | Short text, Categories | department = models.StringField() |
| IntegerField | Integers (whole numbers) | age = models.IntegerField() |
| FloatField | Decimals | percentage = models.FloatField() |
| BooleanField | True or False | is_dictator = models.BooleanField() |
| CurrencyField | Numbers in currency (or point) format | earned_stage1 = models.CurrencyField() |
| LongStringField | Long text | diary_entry = models.LongStringField() |

Building plan - Survey

Models



models.py

Manages Data Structure

Player class

- name: Text (StringField)
- age: Number (IntegerField)
- is_student: True/False (BooleanField)

```
class Player(BasePlayer):  
    name = models.StringField()  
    age = models.IntegerField()  
    is_student = models.BooleanField()
```

Pages

Pages



pages.py

Manages “backend” of the pages

Page class: Welcome page

Page class : Ask page

- Set up the forms name, age, is_student

Page class : Results page

```
class WelcomePage(Page):  
    pass
```

```
class Ask(Page):  
    pass
```

```
class Results(Page):  
    pass
```

Pages - Forms

Pages



pages.py

Manages “backend” of the pages

Page class: Welcome page

Page class : Ask page

- Set up the forms name, age, is_student

Page class : Results page

- oTree handles the form generation internally by two variables in a Page class
 - **form_model** : tell which model you use from your models.py
 - **form_fields** : tell which fields you want the input from

Pages - Forms

Pages



pages.py

Manages “backend” of the pages

Page class: Welcome page

Page class : Ask page

- Set up the forms name, age, is_student

Page class : Results page

```
class WelcomePage(Page):  
    pass
```

```
class Ask(Page):  
    form_model = 'player'  
    form_fields = ['name', 'age', 'is_student']
```

```
class Results(Page):  
    pass
```

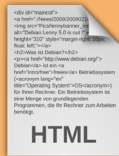
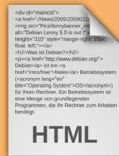
```
page_sequence = [WelcomePage, Ask, Results]
```


Methods of an oTree Page

- BEFORE page is shown:
 - `is_displayed()` (should return True if page is to be shown)
 - `vars_for_template()` (should return variables in a form of dictionary)
- AFTER page is shown:
 - `before_next_page()` (Everything you want to compute/do after the participant has made choices)

Templates

Templates



Manages “frontend” of the pages

- **WelcomePage.html**
- **Ask.html**
- **Results.html**
Show variables.

Adding a Welcome page - Step III - Add the template

my_survey/templates/my_survey/WelcomePage.html

{% extends "global/Page.html" %}
{% load otree static %}

Load otree default styling
Load static files
(Do not alter here)

{% block title %}
Welcome to the experiment
{% endblock %}

Place for the title of the page

between {% block title %} and {% endblock %}

{% block content %}
Please click next to continue
{% next_button %}

Place for everything else
between {% block content %} and {% endblock %}

{% endblock %}

{% next_button %} creates the next button

Templates in oTree

- Generally extend global/Page.html
 - Title block and content block
- Django template language
 - `{% stuff %}` or `{{ stuff }}`
 - Also if-conditions and other constructs possible
 - `{% if xyz %}`
 - `{% else %}`
 - `{% endif %}`
- Everything else is html
 - `<h1> BIG TITLE </h1>`
 - ` DICE`

Adding a Page - Step III - Add the templates

my_survey/templates/my_survey/Ask.html

```
{% extends "global/Page.html" %}  
{% load otree static %}
```

```
{% block title %}  
    Please answer the following questions  
{% endblock %}
```

```
{% block content %}  
    {% formfields %}  
    {% next_button %}
```

```
{% endblock %}
```

Adding a Page- Step III - Field form(s) on a template

```
{% fieldform player.my_field label= 'Enter something' %}
```

OR:

```
{% for field in form %}  
    {% formfield field %}  
{% endfor %}
```

OR:

```
{% formfields %}
```

OR:

```
{% formfield 'my_field' %}
```

Adding a Page- Step III - Add the templates

my_survey/templates/my_survey/Results.html

{% extends "global/Page.html" %}
{% load otree static %}

{% block title %}
 Thank you
{% endblock %}

{% block content %}
Thank you very much for your participation.

 {% next_button %}
{% endblock %}

Adding a Welcome page - Step III - Add the templates

my_survey/templates/my_survey/Results.html

```
{% extends "global/Page.html" %}
{% load otree static %}
```

```
{% block title %}
    Thank you
{% endblock %}
```

```
{% block content %}
Thank you very much for your participation {{player.name }}.
```

```
    {% next_button %}
{% endblock %}
```


Adding a Page - Step III - Templates: using variables

- In a template you can access variables defined in `vars_for_template` of the specific page:

`{{ var }}`

You can also use lists and dictionaries

- You can also access any variable in Constants, Player, Group...

`{{ player.variablename }}`
`{{ group.variablename }}`
`{{ Constants.variablename }}`

Enhancing Our Fields

- **Adding labels**

- `models.StringField(label = "What is your name")`

- **Minimum, maximum age**

- `models.IntegerField(min = 18, max = 117)`

- **Empty allowed**

- `models.IntegerField(blank = True)`

- **Default**

- `models.IntegerField(initial = 20)`

- **Multiple choices**

`department = models.StringField(choices = ["Economics", "Law"])`

Dynamic formfield validation

```
class Player(BasePlayer):
```

```
    fruit = models.StringField()
```

```
    def fruit_choices(self):
```

```
        import random
```

```
        choices = ['apple', 'kiwi', 'mango']
```

```
        random.shuffle(choices)
```

```
        return choices
```

Similar things with other functions:

- `{field_name}_max()`
- `{field_name}_min()`
- `{field_name}_error_message()`
- Validating multiple fields together

Validating multiple fields

```
class Ask(Page):  
    form_model = 'player'  
    form_fields = ['name', 'age', 'is_student', 'department']  
  
    def error_message(self, values):  
        print("The values I get are", values)  
        if values['is_student'] and not values['department']:  
            return "But you said that you are student. Do you mind sharing your department with us?"
```

Show the app in the folder

Lets run the App