GeoTag-X

Project Creation Guide

https://github.com/geotagx/geotagx-project-template

I. Introduction

As you may already know, creating a GeoTag-X project requires a bit of HTML, CSS and JavaScript knowledge.
[ELABORATE]

The purpose of the builder tool is to speed up the creation process by generating the code that runs your project, letting you focus more on its content.

To build your project using the builder tool, we have added a few more requirements that you will need to specify in your project configuration. We have also added the possibility of defining a tutorial configuration that allows you to generate one or more tutorials to complement your project.

Both of the aforementioned configurations can be written in either JSON or YAML file formats although to keep things consistent, the examples in this guide use the JSON format. As a matter of fact, the examples used in this guide are taken from the sample project's JSON configurations.

II. The project configuration

`project.json/project.yaml`

The project configuration -- based on the classic PyBossa configuration -- has been extended to include a few requirements to help create a project's task presenter. In addition to the required **name**, **short_name** and **description** fields, you will need to add the following to your project configuration:

- **why**: a reminder to volunteers about the importance of their contribution to the project.
- **questionnaire**: a list of at least one or more questions asked to volunteers.

II.1. Questionnaires

The project's questionnaire is a list of entries where each is comprised of

- a unique **key*** to identify the entry
- the **type*** of question to ask
- the actual **question*** to ask
- a short **hint** about the question
- a set of **parameters** that configure the entry's behavior
- a **branch** field to determine the questionnaire's control flow

Please note that **key**, **type** and **question** are mandatory fields, while the rest are optional.

II.1.a. Keys

A key is a non-empty string that identifies each questionnaire entry. It is composed of alphanumeric characters, dashes, and/or underscores; but no whitespace.

A key is used in the following ways:

- The flow in a questionnaire is not necessarily linear. It is possible to jump from question to question based on specific criteria. To jump to a specific question, you will need to know its key.
- A help file can be created to elaborate upon a question. Its filename must be identical to the key for the the question you wish to explain.
- The result to each question is mapped to its key.

Certain keywords are reserved for internal use by the project builder and can not be used as keys. The builder tool will promptly notify you when a reserved keyword is used as a key.

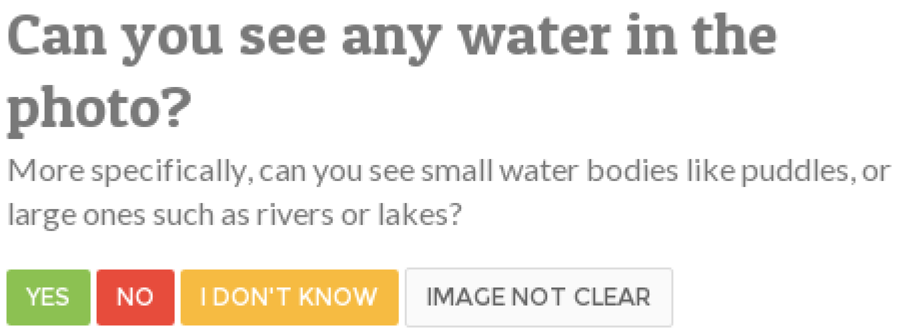
II.1.b. Help

To help volunteers answer questions as accurately as possible, you can either provide a short hint or a more elaborate explanation for questions that may be open to interpretation.

A **hint** is usually a concise way to quickly dispel ambiguity. For instance, let's take a look at the first questionnaire entry in the sample project

```
{
    "key":"isWaterVisible",
    "type":"binary",
    "question":"Can you see any water in the photo?",
    "hint":"More specifically, can you see small water bodies (...) as rivers or lakes?",
    "branch":{
        "yes":"waterColor",
        "no":"end"
    }
}
```

This entry will generate the following

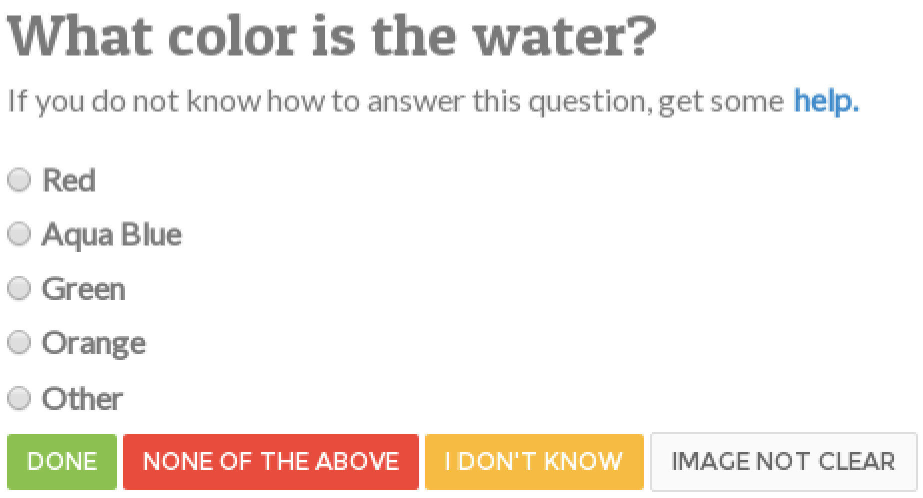


Can you see any water in the photo?

More specifically, can you see small water bodies like puddles, or large ones such as rivers or lakes?

FIGURE 1 - Question hints

Some questions may require more than a hint to get the point across in which case you will need to create an HTML file in the project's **help** directory. The help file's filename must match the key to the question you would like to explain so for instance, a question with the key `waterColor` will look for the help file named `waterColor.html` in the help directory.

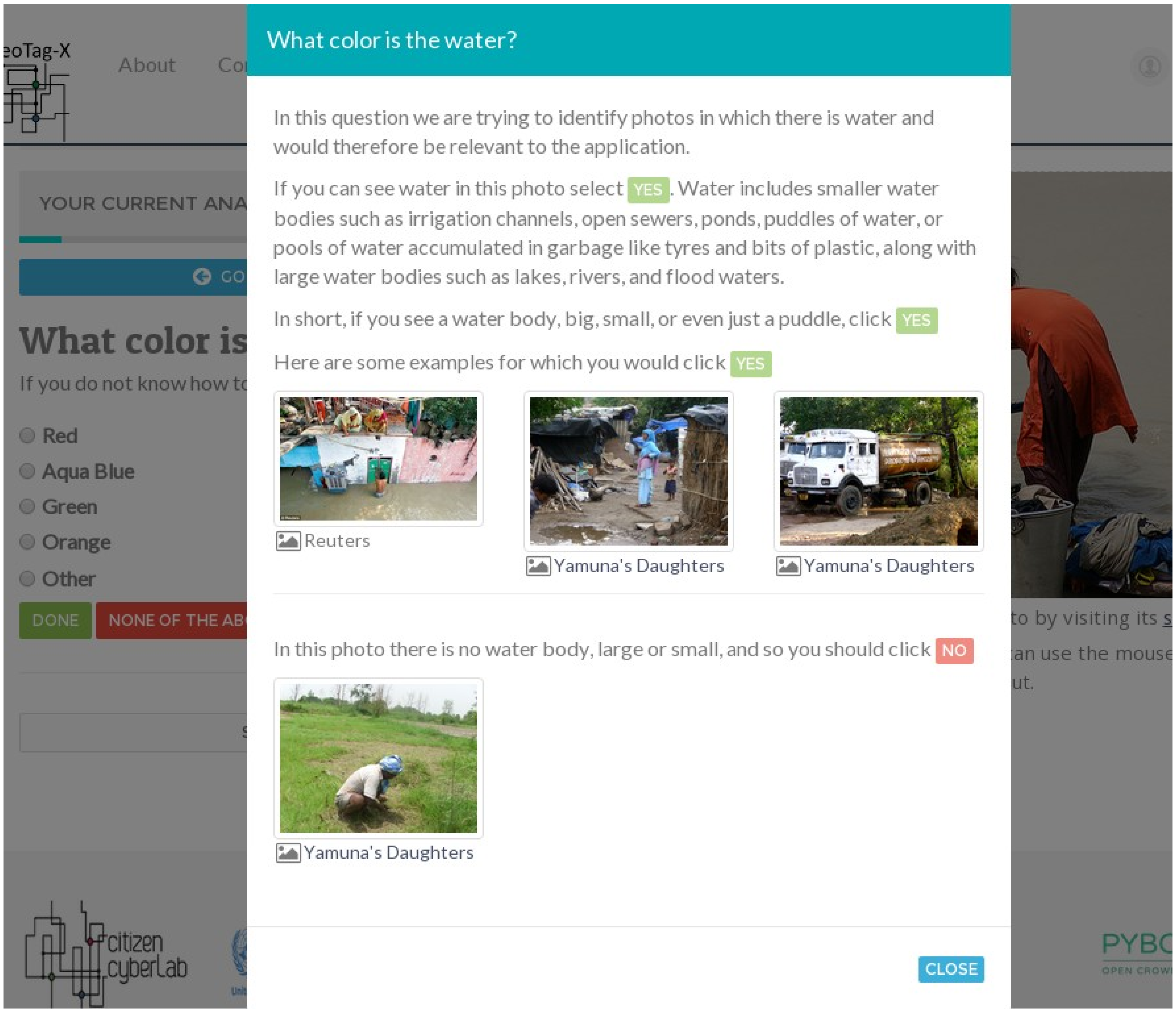


What color is the water?

If you do not know how to answer this question, get some help.

FIGURE 2 - Question help

Wait, where did the help go? Glad you asked. Since it can be as detailed as you would like it to be, it does not make sense to clutter the page so the help is presented in a modal window only when it is requested.



What color is the water?

In this question we are trying to identify photos in which there is water and would therefore be relevant to the application.

If you can see water in this photo select YES. Water includes smaller water bodies such as irrigation channels, open sewers, ponds, puddles of water, or pools of water accumulated in garbage like tyres and bits of plastic, along with large water bodies such as lakes, rivers, and flood waters.

In short, if you see a water body, big, small, or even just a puddle, click YES

Here are some examples for which you would click YES

Reuters

Yamuna's Daughters

Yamuna's Daughters

In this photo there is no water body, large or small, and so you should click NO

Yamuna's Daughters

FIGURE 3 - Help modal window

II.1.c. Input types and parameters

Binary input `type:binary`

This is the input to the most basic type of question you could ask: a yes-no question with an expected answer that is either **Yes** or **No**.

Dropdown lists `type:dropdown-list`

Dropdown lists allow a user to pick a single value from a set of options.

```
parameters:{
        prompt:string,
        size:number,
        options:array
}
```

They can be configured with the following parameters:
- The `prompt` is the always the first, unselectable option in the list. It is usually a message that invites a user to interact with the list. By default, it is set to `Please select an option`.
- The `size` sets the number of options that are simultaneously presented. It does not set the maximum number of options in the list. By default, this value is set to `1`.
- The `options` array contains `<label, value>` pairs where a `label`

Single-choice input `type:select`

```
parameters:{
        size:number,
        options:array
}
```

Check lists `type:checklist`

```
parameters:{
        options:array,
        size:number
}
```

Illustrative check lists `type:illustrative-checklist`

```
type:illustrative-checklist
parameters:{
        options:array
}
```

Short text input `type:text`

```
parameters:{
        placeholder:string,
        maxlength:number
}
```

Long text input `type:longtext`

```
parameters:{
        placeholder:string,
        maxlength:number
}
```

Number input `type:number`

```
parameters:{
        placeholder:string,
        min:number,
        max:number,
        maxlength:number
}
```

Date and time input `type:datetime`

```
parameters:{
        mindate:string,
        maxdate:string,
        mintime:string,
        maxtime:string
}
```

Date input `type:date`

```
parameters:{
        min:string,
        max:string
}
```

URL input `type:url`

```
parameters:{
        placeholder:string,
            maxlength:number
}
```

Geotagging input `type:geotagging`

```
parameters:{
            location:string
}
```

II.1.d. Control flow

By default, the questionnaire has a linear flow which means that questions are asked in the order they are defined. In some cases, it is interesting to ask a specific subset of questions based on how a volunteer progresses through your questionnaire. The simplest example I can give is the first question asked in the sample project

```
{
    "key":"isWaterVisible",
    "type":"binary",
    "question":"Can you see any water in the photo?",
    "hint":"More specifically, can you see small water bodies (...) rivers or lakes?",
    "branch":{
        "yes":"waterColor",
        "no":"end"
    }
}
```

The purpose of this yes-no question is to determine whether or not an image is relevant to the analysis. If it is, i.e. the answer to the question is `yes`, then the question with the key `waterColor` is displayed but if it is not, the questionnaire is promptly ended and a submission form is displayed.

Please note that `end` is not a question key but a reserved keyword used to explicitly end a questionnaire.

With the `branch` mechanism, you are able to dictate the questionnaire's flow. It comes in two flavors: **conditional branching** where the progress is contingent upon the answer to a question, and **absolute branching** where the progress is set in stone. The above JSON snippet is an example of conditional while the following is a good one of absolute branching

```
{
    "key":"smallPoolsOfWaterNearShelters",
    "type":"binary",
    "question":"Are these small pools of water near human shelters or a settlement?",
    "branch":"sourcesOfDrinkingWaterVisible"
}
```

The branching stipulates that whatever the outcome to this question, it is always followed by the question with the key `sourcesOfDrinkingWaterVisible`.

III. The tutorial configuration

`tutorial.json/tutorial.yaml`

Project tutorials are a great way of introducing volunteers to your project and while optional, it is highly recommended that you included at least one in your project.

The tutorial configuration is comprised of one or more tutorials where each one contains:

- an **image** to analyze during the tutorial.
- a link to the web page where the aforementioned image was found, its **source** so to speak. This page usually provides contextual information about the image that may prove to be invaluable to volunteers.
- a set of **assertions** about the image.