

# eZ Publish 5 Field Type Tutorial

## Getting the code

The code created in this tutorial is available on GitHub: <https://github.com/ezsystems/TweetFieldTypeBundle>.

This tutorial covers the creation and development of a custom eZ Publish [Field Type](#). Field Types are the smallest building blocks of content. eZ Publish comes with about [30 native types](#) that cover most common needs (Text line, Rich text, Email, Author list, Content relation, Map location, Float, etc.)

Field Types are responsible for:

- Storing data, either using the native storage engine mechanisms, or specific means
- Validating input data
- Making the data searchable (if applicable)
- Displaying Fields of this type

Custom Field Types are a very powerful type of extension, since they allow you to hook deep into the content model.

You can find the in-depth [documentation about FieldTypes and their best practices](#). It describes how each component of a FieldType interacts with the various layers of the system, and how to implement those.

## Intended audience

This tutorial is aimed at developers who are familiar with eZ Platform and are comfortable with operating in PHP and Symfony2.

## Content of the tutorial

This tutorial will demonstrate how to create a Field Type on the example of a *Tweet* Field Type. It will:

- Accept as input the URL of a tweet (<https://twitter.com/<username>/status/<id>>)
- Fetch the tweet using the twitter oEmbed API (<https://dev.twitter.com/docs/embedded-tweets>)
- Store the tweet's embed contents and url
- Display the tweet's embedded version when displaying the field from a template



## Steps

### 1. The bundle

FieldTypes, like any other eZ Publish 5 extensions, must be provided as Symfony 2 bundles. This chapter will cover the creation and organization of this bundle.

Read more about [Creating the bundle and structuring the bundle](#).

### 2. and 3. API

This part will cover the implementation of the eZ Publish API elements required to implement a custom FieldType

Read more about [implementing the Tweet\Value class and the Tweet\Type class](#).

### 4. Converter

Storing data from any FieldType into the Legacy Storage Engine requires that your custom data is mapped to the data model.

Read more about [Implementing the Legacy Storage Engine Converter](#).

### 5. Templating

Displaying a FieldType's data is done through a twig template.

Read more about [implementing the FieldType template](#).