Serializers allow complex data such as querysets or model instances to be converted to native Python datatypes that can then be easily rendered into JSON, XML or other content types. When the resulting series of bits is reread according to the serialization format, it can be used to create a semantically identical clone of the original object. Serializers also provide deserialization, allowing parsed data to be converted back into complex types.

ModelSerializer

The ModelSerializer class provides a shortcut that lets you automatically create a Serializer class with fields that correspond to the Model fields. The ModelSerializer class is the same as a regular Serializer class, except that it will automatically generate a set of fields for you, based on the model.

JSON

JSON (JavaScript Object Notation)is a text-based format and is the unofficial king of the web as far as object serialization goes. Use the JSON library to serialize and deserialize the simple and complex objects graphs and see what happens. The interface is almost identical to the pickle interface. You have dump(), dumps(), load(), and loads() functions.

Binary Serialization

Binary serialization allows single objects or complex models to be converted to binary streams, which may be stored in files or transported to other systems. Python’s NumPy array can be used to serialize and deserialize data to and from byte representation.

XML Serialization

XML Serialization is the process of serializing a Python Object to the form of XML or from an XML to Python.

**import** xml.etree.ElementTree **as** ET