**Django REST Framework - Summary: Serializers vs Views**

**1. Role of the Serializer**

* Converts data **from JSON to Python objects (deserialization)** and **from Python objects to JSON (serialization)**.
* Validates incoming data according to the model’s field definitions.
* Can save validated data as a model instance using .save().
* Can define **custom fields** not present in the model:
  + For computed output (e.g., summary = expression + result).
  + For write-only input (e.g., confirm\_password).
  + These fields are **not saved in the database**, but help with processing and response.
* Keeps data clean, structured, and secure for API use.

**2. Role of the View**

* Receives HTTP requests (GET, POST, PUT, DELETE).
* Orchestrates the flow:
  + Receives incoming data (request.data).
  + Instantiates the serializer.
  + Calls is\_valid() to validate.
  + If valid, calls serializer.save() to persist data.
  + Returns a serialized response (serializer.data).
* Best place for **business logic**, like computations (e.g., evaluate expression before saving).
* Delegates data formatting and validation to the serializer.

**3. Serialization vs Deserialization in the View**

* **Deserialization**: When the view **creates a serializer with incoming data**:
* serializer = MySerializer(data=request.data)
  + DRF treats this as input.
  + You then call serializer.is\_valid() and serializer.save().
* **Serialization**: When the view **creates a serializer with an instance**:
* serializer = MySerializer(instance=my\_model)
  + DRF converts the model instance to JSON for response.
  + You return serializer.data to the client.

**Quick Tip**:

* data=... → means you’re **deserializing** input from the client.
* instance=... → means you’re **serializing** a model to send out as JSON.

**Conclusion**: Serializers and Views work hand-in-hand. The **View handles the request/response**, while the **Serializer ensures the data is valid, structured, and safely processed**. Serializers can be extended with custom logic and fields — making them powerful helpers in your API workflow.