|  |  |  |  |
| --- | --- | --- | --- |
| **Annotation**  **Name** | **Description** | **Input** | **Output** |
| **@JsonIgnore** | Ignores a field during serialization/deserialization. | public class User {  public String name;  @JsonIgnore  public String password;} | {"name": "John"} (password is ignored) |
| **@JsonInclude** | Include only non-null/non-empty values in JSON. | @JsonInclude(JsonInclude.Include.NON\_NULL)  public class Product {  public String name;  public String description;  } | If description = null, it'll be excluded from the JSON. |
| **@JsonFormat** | **:** Format dates or other types during serialization. | public class Event {  @JsonFormat(pattern = "yyyy-MM-dd")  public LocalDate eventDate;  } | {"eventDate":"2025-07-29"} |
| **@JsonAlias** | Accepts multiple names for a field during deserialization. | public class Car {  @JsonAlias({"car\_name", "vehicleName"})  public String name;  } | {"car\_name": "Toyota"} or {"vehicleName": "Toyota"} |
| **@JsonAnyGetter / @JsonAnySetter** | Handle unknown properties dynamically. | public class Config {  private Map<String, String> settings = new HashMap<>();  @JsonAnySetter  public void set(String key, String value) {  settings.put(key, value);  }  @JsonAnyGetter  public Map<String, String> getSettings() {  return settings;  }  } | Accepts arbitrary key-value pairs from JSON. |
| **@JsonEnumDefaultValue** | Assign default enum if value doesn’t match any enum. | public enum Status {  ACTIVE,  INACTIVE,  @JsonEnumDefaultValue  UNKNOWN} | If input is invalid, it’ll use UNKNOWN. |
| **@JsonIgnoreProperties** | Ignore specified fields from serialization/deserialization. | @JsonIgnoreProperties({"age", "email"})  public class Person {  public String name;  public int age;  public String email;  } | return only "name" |
| **@JsonIncludeProperties** | Only include specified fields during serialization. | @JsonIncludeProperties({"name", "email"})  public class Person {  public String name;  public int age;  public String email;  } | ignore the field age |
| **@JsonFilter** | Apply filtering dynamically using filter ID. | @JsonFilter("userFilter")  public class User {  public String username;  public String password;  } | SimpleBeanPropertyFilter in ObjectMapper. |
| **@JsonIdentityInfo** | Handle cyclic references using object IDs. | @JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "id")  public class Employee {  public int id;  public String name;  public Department department;  } |  |
| **@JsonIdentityReference** | **:** Serialize the referenced object as an ID instead of whole object | public class Employee {  @JsonIdentityReference(alwaysAsId = true)  public Department department;  } |  |
| **@JsonManagedReference / @JsonBackReference** | Manage parent-child bidirectional relationships.(two way mapping) | public class User {  public String name;    @JsonManagedReference  public List<Address> addresses;  }  public class Address {  public String city;  @JsonBackReference  public User user;  } |  |