@RestController

- **Purpose:**
  - Combines `@Controller` and `@ResponseBody`.
  - Used for creating RESTful web services.
- **Usage:**
  - Applied to classes to indicate that all methods return data suitable for RESTful services.

@Service

- **Purpose:**
  - Marks a class as a service class in the business layer.
  - Typically used to hold business logic.
- **Usage:**
  - Applied to service classes.

@Repository

- **Purpose:**
  - Indicates that a class is a Data Access Object (DAO).
  - Converts checked exceptions into Spring's DataAccessException.
- **Usage:**
  - Applied to classes at the persistence layer.

@Entity

- Annotates a Java class to be an entity, representing a table in a relational database.
- Used in conjunction with the Jakarta Persistence API to map entities to database tables.

@Table

- Specifies the details of the table that will be used to persist the entity.
- Allows customization of the table name, schema, and other attributes.

@Column(unique=true, nullable=false)

- Annotates a field to specify the column mapping for a persistent property.
- `unique=true` ensures that the values in the column are unique.
- `nullable=false` specifies that the column cannot have a null value.

@Id

- Marks a field as the primary key of the entity.
- Indicates that the annotated field will be used as the primary key in the corresponding database table.

@GeneratedValue

- Specifies the strategy for generation of primary key values.
- Used in conjunction with the @Id annotation to define how primary key values should be generated.

@NotNull

- An annotation from the Bean Validation API.
- Marks a field as not nullable, ensuring that the corresponding database column cannot contain null values.

Spring Annotations

@RestControllerAdvice

- A specialization of the @ControllerAdvice annotation.
- Used in Spring MVC to define global controllers advice.
- Centralizes exception handling for all controllers in the application.

@ExceptionHandler

- Indicates a method that handles exceptions thrown by specific controller methods.
- Used in conjunction with @RestControllerAdvice to provide a centralized exception handling mechanism.

ResponseEntity

- Represents the entire HTTP response in Spring applications.
- Used to customize the response status, headers, and body.
- Provides flexibility in constructing and returning HTTP responses from controller methods.