1. @Controller : It indicates that the class is responsible for handling incoming HTTP requests.

```java
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Component
public @interface Controller {
    @AliasFor(
            annotation = Component.class
    )
    String value() default "";
}
```

2. @RestController :

   RestController = Controller + ResponseBody

3. @ResponseBody:
   - Denotes that return value of the controller method should be serialized to HTTP response body.
   - If we do not provide ResponseBody, Spring will consider response as name for the view and tries to resolve and render it (in case we are using the @Controller annotation)

4. @RequestMapping
   1) Value, path (both are same)
   2) Method
   3) Consumes, produces
   4) @Mapping
   5) @Reflective({ControllerMappingReflectiveProcessor.class})

@RequestMapping (path = "/fetchUser", method = RequestMethod.GET, consumes = "application/json", produces = "application/json")

```java
@Target({ElementType.TYPE, ElementType.METHOD})
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Mapping
@Reflective({ControllerMappingReflectiveProcessor.class})
public @interface RequestMapping {
    String name() default "";

    @AliasFor("path")
    String[] value() default {};

    @AliasFor("value")
    String[] path() default {};

    RequestMethod[] method() default {};

    String[] params() default {};

    String[] headers() default {};

    String[] consumes() default {};

    String[] produces() default {};
}
```

5. @RequestParam: Used to bind, request parameter to controller method parameter.

http://localhost:8080/api/fetchUser?firstName=SHRAYANSH&lastName=JAIN&age=32

```java
@RestController
@RequestMapping(value = "/api/")
public class SampleController {

    @GetMapping(path = "/fetchUser")
    public String getUserDetails(@RequestParam (name = "firstName") String firstName,
                                 @RequestParam (name = "lastName", required = false) String lastName,
                                 @RequestParam(name = "age") int age) {
        return "fetching and returning user details based on first name = " + firstName + " , lastName = " + lastName + " and age is = " + age;
    }
}
```

The framework automatically performs type conversion from the request parameter's string representation to the specified type.

1. Primitive types: Such as int, long, float, double, boolean, etc.
2. Wrapper classes: Such as Integer, Long, Float, Double, Boolean, etc.
3. String: Request parameters are inherently treated as strings only.
4. Enums: You can bind request parameters to enum types.
5. Custom object types: We can do it using a registered PropertyEditor.

How to used PropertyEditor?

```java
@RestController
@RequestMapping(value = "/api/")
public class SampleController {

    @InitBinder
    protected void initBinder(DataBinder binder) {

        binder.registerCustomEditor(String.class, field: "firstName", new FirstNamePropertyEditor());
    }

    @GetMapping(path = "/fetchUser")
    public String getUserDetails(@RequestParam (name = "firstName") String firstName,
                                 @RequestParam (name = "lastName", required = false) String lastName,
                                 @RequestParam(name = "age") int age) {
        return "fetching and returning user details based on first name = " + firstName + " , lastName = " + lastName + " and age is = " + age;
    }
}
```

```java
public class FirstNamePropertyEditor extends PropertyEditorSupport {
    @Override
    public void setAsText(String text) throws IllegalArgumentException {
        setValue(text.trim().toLowerCase());
    }
}
```

6. @PathVariable: Used to extract values from the path of the URL and help to bind it to controller method parameter.

```java
@RestController
@RequestMapping(value = "/api/")
public class SampleController {

    @GetMapping(path = "/fetchUser/{firstName}")
    public String getUserDetails(@PathVariable(value = "firstName") String firstName) {
        return "fetching and returning user details based on first name = " + firstName;
    }
}
```

7. @RequestBody: Bind the body of HTTP request (typically JSON) to controller method parameter (java object).

```java
@RestController
@RequestMapping(value = "/api/")
public class SampleController {


    @PostMapping(path = "/saveUser")
    public String getUserDetails(@RequestBody User user) {
        return "User created " + user.username + ":" + user.email;
    }
}
```

```java
public class User {

    @JsonProperty ("user_name")
    String username;
    String email;

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

```
curl --location --request POST 'http://localhost:8080/api/saveUser' \
--header 'Content-Type: application/json' \
--data-raw '{
   "user_name": "Shrayansh",
   "email": "sjxyztest@gmail.com"
}'
```

8. ResponseEntity : It represents the entire HTTP response.

Header, status, response body etc.

```java
@RestController
@RequestMapping(value = "/api/")
public class SampleController {


    @GetMapping(path = "/fetchUser")
    public ResponseEntity<String> getUserDetails(@RequestParam(value = "firstName") String firstName) {
        String output = "fetched User details of " + firstName;
        return ResponseEntity.status(HttpStatus.OK).body(output);
    }
}
```