스프링시작하기

스프링 API 9장

1. JsonIgnore
2. Filter
3. 소스버전별관리하기

박명회

1.1 @JsonIgnore

예제에서는 @JsonIgnore 애너테이션을 사용하여 주민등록번호 필드를 숨겼습니다. 이제 이 필드는 JSON 직렬화 및 역직렬화 과정에서 무시되므로 클라이언트에게 전송되지 않습니다.

주의할 점은 주민등록번호와 같은 민감한 정보가 데이터베이스에 저장되는 경우, 암호화하여 저장하거나 다른 안전한 방법으로 보호되어야 합니다. 클라이언트에게 필요한 경우에만 안전한 방식으로 제공되어야 합니다.

```java
import com.fasterxml.jackson.annotation.JsonIgnore;

public class User {
    private Long id;
    private String username;
    private String email;

    @JsonIgnore
    private String 주민등록번호;

    // 생성자, getter, setter 생략
}
```

1.2 Admin User Filter

```
ml.jackson.annotation.JsonFilter;

nfo")
User {
d;
 username;
성자 등 생략
```

```
import org.springframework.beans.BeanUtils;
import org.springframework.http.converter.json.MappingJacksonValue;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import com.fasterxml.jackson.databind.ser.FilterProvider;
import com.fasterxml.jackson.databind.ser.impl.SimpleBeanPropertyFilter;
import com.fasterxml.jackson.databind.ser.impl.SimpleFilterProvider;

import kr.co.joneconsulrting.newmyrestfulservice.bean.User;

@RestController
public class UserController {

    @GetMapping("/admin-user")
    public MappingJacksonValue getAdminUser() throws Exception {
        // User 객체 생성
        User user = new User();
        user.setId(id:1);
        user.setUsername(username:"admin");
        user.setEmail(email:"admin@example.com");

        // AdminUser로 변환
        AdminUser adminUser = new AdminUser();
        BeanUtils.copyProperties(adminUser, user);

        // 필터 적용
        MappingJacksonValue mappingJacksonValue = new MappingJacksonValue(adminUser);
        FilterProvider filters = new SimpleFilterProvider().addFilter(id:"UserInfo",
                SimpleBeanPropertyFilter.filterOutAllExcept(...propertyArray:"id", "username"));
        mappingJacksonValue.setFilters(filters);

        return mappingJacksonValue;
    }
}
```

1.2 Admin User Filter

이제 "/admin-user" 엔드포인트에 대한 GET 요청을 처리하는 컨트롤러에서 @JsonFilter 어노테이션을 사용하여 필터링된 응답을 생성합니다. 필터링된 필드는 "id"와 "username"이며, 나머지 필드는 필터링됩니다.

1.4 Admin Users 만들어보기

```java
@GetMapping("/admin-users")
public MappingJacksonValue retrieveAllUsers() {
    List<User> users = service.findAll();
    List<AdminUser> adminUsers = new ArrayList<>();
    AdminUser adminUser = null;

    for (User user : users) {
        adminUser = new AdminUser();
        BeanUtils.copyProperties(user, adminUser);
        adminUsers.add(adminUser);
    }

    SimpleBeanPropertyFilter filter = SimpleBeanPropertyFilter
            .filterOutAllExcept(...propertyArray:"id", "name", "joinDate", "ssn");

    FilterProvider filters = new SimpleFilterProvider().addFilter(id:"UserInfo", filter);

    MappingJacksonValue mapping = new MappingJacksonValue(adminUsers);
    mapping.setFilters(filters);

    return mapping;
}
```

1.4 RestApi 버전관리

```java
@RestController
@RequestMapping("/api/v1")
public class ApiControllerV1 {

    @GetMapping("/hello")
    public String hello() {
        return "Hello from API version 1!";
    }

    @GetMapping("/greet")
    public String greet() {
        return "Greetings from API version 1!";
    }
}
```

```java
@RestController
@RequestMapping("/api/v2")
public class ApiControllerV2 {

    @GetMapping("/hello")
    public String hello() {
        return "Hello from API version 2!";
    }

    @GetMapping("/greet")
    public String greet() {
        return "Greetings from API version 2!";
    }

    @GetMapping("/new-feature")
    public String newFeature() {
        return "This is a new feature available in API version 2!";
    }
}
```

1.4 파라미터로 버전관리

```java
@RestController
public class UserController {

    @GetMapping(value = "/users", params = "version=v1")
    public String getUsersV1() {
        return "This is version 1 of the API";
    }

    @GetMapping(value = "/users", params = "version=v2")
    public String getUsersV2() {
        return "This is version 2 of the API";
    }
}
```

1.4 해더로 버전 관리해보기

```java
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class UserController {

    @GetMapping(value = "/users", headers = "version=v1")
    public ResponseEntity<String> getUsersV1() {
        return ResponseEntity.ok("User list from API version 1");
    }

    @GetMapping(value = "/users", headers = "version=v2")
    public ResponseEntity<String> getUsersV2() {
        return ResponseEntity.ok("User list from API version 2");
    }
}
```

1.4 해더로 버전 관리해보기

```java
@RestController
public class ApiController {

    @GetMapping("/hello")
    public ResponseEntity<String> hello(@RequestHeader HttpHeaders headers) {
        String version = headers.getFirst("API-Version");
        if (version != null && version.equals("1")) {
            return ResponseEntity.ok("Hello from API version 1!");
        } else if (version != null && version.equals("2")) {
            return ResponseEntity.ok("Hello from API version 2!");
        } else {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Invalid API
        }
    }

    @GetMapping("/greet")
    public ResponseEntity<String> greet(@RequestHeader HttpHeaders headers) {
        String version = headers.getFirst("API-Version");
        if (version != null && version.equals("1")) {
            return ResponseEntity.ok("Greetings from API version 1!");
        } else if (version != null && version.equals("2")) {
            return ResponseEntity.ok("Greetings from API version 2!");
        } else {
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Invalid API
        }
    }
}
```

고맙습니다.