Paginacija i validacija

Paginacija

- **Aplikacije neretko sadrže više hiljada, pa i miliona instanci nekog entiteta. Neophodno je podržati da se prilikom jednog zahteva klijentske aplikacije vrati samo deo tih instanci.**
- **Ovakav način rukovanja podacima se naziva paginacija.**
- **Spring poseduje mehanizme kojima je podržana paginacija.**

Paginacija

- **Kako bi bila omogućena paginacija, potrebno je koristiti JpaPagingAndSortingRepository, ili neki od naslednika.**

- **JpaRepository nasleđuje gore pomenuti interfejs.**

Paginacija – Repozitorijum

- **Kako bismo obezbedili podršku za paginaciju, na nivou repozitorijuma, potrebno je iskoristiti metode koje primaju objekat klase koja implementira Pageable interfejs, i vraćaju objekat klase koja implementira Page interfejs:**

  Page<T> findAll(Pageable pageable)

Paginacija – Repozitorijum

- Query metode, takođe, deklarišemo tako da primaju i vraćaju prethodno pomenute objekte. Primer:

```
Page<City> findByCountryId(Long countryId, Pageable pageRequest);
```

- Isto važi I za metode sa @Query anotacijama, stavite da je jedan od parametara Pageable.

Paginacija - Servis

- **Prilikom pozivanja metoda repozitorijuma koje podržavaju paginaciju, potrebno je proslediti objekat klase koja implementira Pageable interfejs.**
- **U tu svrhu, koristimo klasu PageRequest:**

```
countryRepository.findAll(new PageRequest(BrojStranice (int), EntitetaPoStranici (int)));
```

Paginacija - Kontroler

- **Objekat koji vrate objašnjene metode jeste instanca klase koja implementira Page interfejs - PageImpl.**
- **Ova klasa enkapsulira listu preuzetih entiteta, koji se mogu dobiti pozivom metode getContent().**
- **Takođe, moguće je dobiti i ukupan broj stranica (getTotalPages()) i ukupan broj entiteta (getTotalElements()).**

Paginacija - Kontroler

- **Sve navedeno je moguće iskoristiti kako bi se klijentskoj aplikaciji preko Http header-a prosledio i ukupan broj stranica radi ispisa:**

```
//pageNum je @RequestParam
Page<Country> countries = countryService.findAll(pageNum);


HttpHeaders headers = new HttpHeaders();
headers.add("totalPages", Integer.toString(countries.getTotalPages()) );
return  new ResponseEntity<>(countries.getContent()),headers,HttpStatus.OK);
```

Paginacija - JQuery

- Sa klijentske strane je potrebno potom pokupiti header. U slučaju JQuery biblioteke:

```
$.ajax({
 type: 'GET',
 url:'url?pageNum=2',
 success: function(data, textStatus, request){
     alert(request.getResponseHeader('some_header'));
 }
 error: function (request, textStatus, errorThrown) {
     alert(request.getResponseHeader('some_header'));
 }
});
```

Validacija

- **Podatke je uvek potrebno validirati i na serverskoj strani!**
- **Spring Framework nam pruža podršku za validaciju.**

- **Kako bismo validirali elemente modela pristigle sa klijenta, potrebno je izvršiti validaciju podataka pristiglih POST i PUT Http zahtevima.**

- **Proces prevođenja tela zahteva (@RequestBody) u instancu klase modela se naziva binding.**

Validacija

- **Kako bi validacija mogla biti automatski izvršena, polja klasa modela je potrebno anotirati validacionim anotacijama.**
- **API za validaciju je odvojen od JPA API-ija!**

- **Listu predefinisanih anotacija je moguće videti na sledećoj stranici:**

  http://docs.jboss.org/hibernate/stable/validator/reference/en-US/html_single/#section-builtin-constraints

Validacija

- **Primer validacije sa definisanjem poruke:**

```
@NotBlank(message="Name cannot be empty")
@Pattern(regexp="^[A-Za-z]*$", message="Name can contain only letters")
private String name;
```

Validacija

- **Ako @RequestBody parametar funkcije kontrolera anotiramo sa @Validated (ili @Valid, @Validated je nadskup), prilikom binding-a će automatski biti odrađena validacija.**
- **Rezultate u validacije možemo pokupiti u obliku objekta klase BindingResults ili Errors.**
- **Moguće je i “ručno” inicirati validaciju!**

Validacija

- **Primer kontrolera sa validacijom:**

```
@RequestMapping(method=RequestMethod.POST, consumes="application/json")
public ResponseEntity<?> addCountry(@Validated @RequestBody Country country, Errors errors){
    if(errors.hasErrors()) {
        return new ResponseEntity<String>(errors.getAllErrors().toString(), HttpStatus.BAD_REQUEST);
    }

    Country createdCountry = countryService.save(country);
    return new ResponseEntity<Country>(createdCountry, HttpStatus.CREATED);
}
```