Spring MVC Validasyon İşlemleri

harezmi bilişim çözümleri

Spring MVC ve Validasyon

- Controller bean'larında validasyonun aktive olması için ApplicationContext'de **<mvc:annotation-driven/>** elemanına ihtiyaç vardır

```
<beans>

    <mvc:annotation-driven/>

</beans>
```

Classpath'deki mevcut JSR-303/JSR-349 kütüphaneleri otomatik olarak tespit edilip JSR Validator provider, spring managed bean olarak konfigüre edilmektedir

Spring MVC ve Validasyon

- Spring MVC'de validasyon işlemleri **Validator** arayüzü üzerinden gerçekleştirilir

```
public interface org.springframework.validation.Validator {
    boolean supports(Class<?> clazz);
    void validate(Object target, Errors errors);
}
```

- Form model nesnelerini **data binding sırasında** validate etmek için kullanılır
- Validator sınıfları yazılarak **WebDataBinder** nesnesine tanıtılır

Spring MVC ve Validasyon

```
import org.springframework.validation.Validator;

public class OwnerValidator implements Validator {
    @Override
    public boolean supports(Class<?> clazz) {
        return Owner.class.isAssignableFrom(clazz);
    }
    @Override
    public void validate(Object target, Errors errors) {
        Owner owner = (Owner) target;
        if (StringUtils.isEmpty(owner.getFirstName())) {
            errors.rejectValue("firstName", "errors.required.firstName");
        }

        if (StringUtils.isEmpty(owner.getLastName())) {
            errors.rejectValue("lastName", "errors.required.firstName");
        }

        errors.pushNestedPath("address");
        if(StringUtils.isEmpty(owner.getAddress().getPhone())) {
            errors.rejectValue("phone", "errors.required.phone");
        }
        errors.popNestedPath();
    }
}
```

Spring MVC ve Validasyon

```
@Controller
public class PetclinicController {

    @InitBinder
    protected void initBinder(WebDataBinder binder) {
        binder.setValidator(new OwnerValidator());
    }

}
```

Handler metot çağrılmadan önce @InitBinder ile işaretlenmiş metot invoke edilerek WebDataBinder'ın özelleştirilmesi sağlanır

DataBinder nesnesine sadece bu controller'da kullanılmak üzere spesifik bir Validator nesnesi tanımlanabilir

```
@Controller
public class PetclinicController {

    @InitBinder
    protected void initBinder(WebDataBinder binder) {
        binder.addValidators(new OwnerValidator());
    }

}
```

Ya da **<mvc:annotation-driven/>** ile tanımlanan global validator'a ilave lokal validator'lar da eklenebilir

Spring MVC ve Validasyon

- Form data binding sırasında meydana gelen validasyon hataları **Errors** nesnesi içerisinde toplanır
- Daha sonra bu hatalara Spring MVC handler metot içerisinde **BindingResult** nesnesi ve sayfa içerisinde **<spring:bind/>** tag'i vasıtası ile erişilebilir

Spring MVC Controller'ları ve Validasyon

- @**Controller** metotlarındaki **input parametreler** ve **return değerleri** otomatik olarak da validate edilebilir
- Handler metot invokasyonu sırasında meydana gelen **validasyon hataları** da yine Controller içerisinde @**ExceptionHandler** ile ele alınabilir

Spring MVC Controller'ları ve Validasyon

Eğer metot parametresinin validasyonunda herhangi bir sorun olursa MethodArgumentNotValidException fırlatılır

```
@RestController
public class PetClinicRestController {

    @Autowired
    private PetClinicService petClinicService;

    @RequestMapping(value="/owner",method=RequestMethod.POST)
    @ResponseStatus(code=HttpStatus.CREATED)
    public Long createOwner(@RequestBody @Valid Owner owner) {
        petClinicService.create(owner);
        return owner.getId();
    }

    @ExceptionHandler
    public void handle(MethodArgumentNotValidException exception,
                       HttpServletResponse response) {
        response.setStatus(HttpStatus.PRECONDITION_REQUIRED.value());
    }
}
```

Spring MVC Controller'ları ve Validasyon

JSR-303 validasyon kısıtları da denetlenebilir. Bunun için sınıf düzeyinde @Validated anotasyonu kullanılmalıdır. Hata durumunda ConstraintViolationException fırlatılır

```
@RestController
@Validated
public class PetClinicRestController {

    @Autowired
    private PetClinicService petClinicService;


    @RequestMapping(value="/owners/{email}",method=RequestMethod.GET)
    public Owner findOwnerByEmail(@PathVariable @Email String email) {
        return petClinicService.findOwner(email);
    }

    @ExceptionHandler
    public void handle(ConstraintViolationException exception,
                                        HttpServletResponse response) {
        response.setStatus(HttpStatus.PRECONDITION_REQUIRED.value());
    }
}
```

Spring MVC Controller'ları ve Validasyon

Validasyon hataları **BindingResult** nesnesinde toplanır. BindingResult kontrol edilerek iş akışı yönlendirilebilir.

```
@Controller
public class PetClinicRestController {

    @RequestMapping(value="/owner",method=RequestMethod.POST)
    @ResponseStatus(code=HttpStatus.CREATED)
    public void create(@RequestBody @Valid Owner owner,
                       BindingResult bindingResult,
                       HttpServletResponse response)
                    throws Exception {
        if(bindingResult.hasErrors()) {
            //...
            response.setHeader("Location", "/ownerCreateFailed");
        } else {
            //...
            response.setHeader("Location", "/ownerCreateSuccess");
        }
    }
}
```

BindingResult, validate edilen Model argümanından hemen sonra gelmelidir

Spring MVC ve Custom Global Validator

- İstenirse IoC container içerisinde tanımlanan **custom bir Validator** implemantasyonu da **global validator** nesnesi olarak tanıtılabilir

```
<beans>

    <mvc:annotation-driven validator="validator"/>

    <bean id="validator"
              class="com.javaegitimleri.web.GlobalValidatorImpl"/>

</beans>
```

İletişim

- Harezmi Bilişim Çözümleri
- Kurumsal Java Eğitimleri
- http://www.java-egitimleri.com
- info@java-egitimleri.com