Web Apps

ALWAYS IMPORT

```
<%@ taglib prefix="form" uri="http://www.springframework.org/tags/form
" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
```

```
<spring:url value="/css/main.css" var="urlCss"/>
<link rel="STYLESHEET" href="${urlCss}" type="text/css" />

<c:url var="logoutUrl" value="/logout"></c:url>
<form action="${logoutUrl}" method="post">

<spring:url value="/contacts/" var="contactUrl" />
<a href="${contactUrl}${contact.id}">${contact.firstname}</a>
```

Resource Handler (Webconfig)

```
@Bean
public MessageSource messageSource() {
    ResourceBundleSource messageSource = new ResourceBundleMessageSour
ce();
    messageSource.setBasename("resources/converter");
    return messageSource;
}
```

Message Source (Webconfig)

```
@Override
public void addResourceHandlers(ResourceHandlerRegistry registry) {
    registry.addResourceHandler("/css/**").addResourceLocations("resou
rces/css/");
 registry.addResourceHandler("/scripts/**").addResourceLocations("reso
urces/scripts/");
}
```

Internationalization

```
<head>
    <spring:message code="date.format.pattern" var="dateFormatPattern"
/>
    <spring:message code="label_contact_first_name" var="labelContactF
irstName"/>
</head>
<p>
    Today is <fmt:formatDate value="${today}" pattern="${dateFormatPat
tern}" />
    ${labelContactFirstName}
</p>
```

Tags

```
<%@ attribute name="date" required="true" type="java.time.LocalDate" %
>
<%@ attribute name="pattern" required="false" type="java.lang.String"
%>

<c:if test="${empty pattern}">
    <c:set var="pattern" value="yyyy/MM/dd"/>
</c:if>

<fmt:parseDate value="${date}" pattern="yyyy-MM-dd" var="parsedDate" t
ype="date"/>
<fmt:formatDate value="${parsedDate}" type="date" pattern="${pattern}"
/>
```

```
<%@ taglib prefix="tags" tagdir="/WEB-INF/tags" %>

<body>
    <tags:localDate date="${contact.birthDate}" pattern="${dateFormatP
attern}"/>
</body>
```

Chapter 1: Servlet/JSP

JSP: Java Server Pages

- **Scriptlets** (<% and %>)
    - <% out.print(new java.util.Date()); %>
- **Comments** (<%-- and --%>)
- **Expressions** (<%= and %>)
    - <% =new java.util.Date() %>
- **Declaration** (<%! and %>)
- **Directive**: (<%@ and %>)
    - <%@ pageimport="java.util.Date, domein.Rotator" %>

JSTL: JSP Standard Tag Library

Foreach

```
<c:forEach items="${colorArray}" var="color" varStatus="colorCounter">
    ${colorCounter.count} ${color}<br>
</c:forEach>
```

If

```
<c:if test="${ userType == 'student'}"> <!-- test="${not empty myStrin
g}" -->
    <jsp:includepage="inputComments.jspf"/>
</c:if>
```

Choose, when, otherwise

```
<c:choose>
    <c:when test="${ userType == 'student'}">
        <jsp:include page="inputComments.jspf"/>
    </c:when>
    <c:when test="${ userType == ‘professor'}" >
        Professor
    </c:when>
    <c:otherwise>
        Person
    </c:otherwise>
</c:choose>
```

Import (content from outside the container)

```
<body>
    Horse: <c:import url="http://www.wickedlysmart.com/skyler/horse.js
p"/>
</body>
```

Forward

```
<jsp:forward page="BeerServlet"></jsp:forward>
```

Include (only pages form the current web app)

```
<html>
    <body>
        <jsp:include page = "header.jsp"/>

        <actualContent></actualContent>

        <jsp:include page = "footer.jsp"/>
    </body>
</html>
```

Chapter 2: Spring Basis

Annotations

Annotations (ComponentScan niet vergeten)

```
public class StartUp {

    public static void main(String... args) {
        ApplicationContext ctx = new AnnotationConfigApplicationContex
t(Cfg.class);

        // calculate refers to CalculateSpring @Service
        CalculateSpring bean = context.getBean("calculate", CalculateS
pring.class);
        bean.execute(args);
    }
}
```

```
@ComponentScan(basePackages = {"domain", "spring_wiring"}) @Configurat
ion
public class Cfg { }
```

```
@Service("calculate")
public class CalculateSpring {

    private Operation ops;
    private ResultWriter writer;

    @Qualifier("add") // @Qualifier("multiply")
    @Autowired
    public void setOps(Operation ops) { this.ops = ops; }
    public Operation getOps() { return ops; }

    @Autowired
    public void setWriter(ResultWriter writer) { this.writer = writer;
 }
    public ResultWriter getWriter() { return writer; }

    public void execute(String[] numbers) {
        long op1 = Long.parseLong(numbers[0]);
        long op2 = Long.parseLong(numbers[1]);
        writer.showResult("The result of " + op1 + ops.getName() + op2
 + " is "
            + ops.operate(op1, op2) + "!");
    }
}
```

```
@Service("add") // @Service("multiply") public class OperationMultiply
 implements Op...
public class OperationAdd implements Operation {

    @Override
    public long operate(long op1, long op2) { return op1 + op2; }

    @Override
    public String getName() { return " plus "; }
}
```

Aspect Oriented Programming

Idol (before, after, after-returning, after-throwing, around)

```
public class SpringIdolAop {

    public static void main(String[] args) {
        ApplicationContext context = new AnnotationConfigApplicationCo
ntext(Cfg.class);

        Performer performer = context.getBean("alicia", Performer.clas
s);
        performer.perform();
    }
}
```

```
@Configuration @EnableAspectJAutoProxy
public class Cfg {

    @Bean public Audience audience() { return new Audience(); }
    @Bean public Singer alicia() { return new Singer("Alicia Keys", "N
o One"); }

    @Bean
    public CriticismEngine criticismEngine() {
        CriticismEngine engine = new CriticismEngine();
        String[] criticismPool = new String[3];
        criticismPool[0] = "I'm not being rude, but that was appalling
.";
        engine.setCriticismPool(criticismPool);
        return engine;
    }
}
```

```
@Aspect
public class Audience {

    @Pointcut("execution(* *.perform(..))")
    public void performance() { }

    @Before("performance()")
    public void takeSeats() {
        System.out.println("The audience is taking their seats.");
    }

    @Before("performance()")
    public void turnOffCellPhones() {
        System.out.println("The audience is turning off their cellphon
es");
    }

    @AfterReturning("performance()")
    public void applaud() {
        System.out.println("CLAP CLAP CLAP CLAP CLAP");
    }

    @AfterThrowing("performance()", throwing = "ex")
    public void demandRefund() {
        System.out.printf("This exception has occurred: %s", ex);
    }
}
```

```
@Aspect
public class CriticismEngine {

    private String[] criticismPool;
    private SecureRandom random = new SecureRandom();

    public void setCriticismPool(String[] p) { this.criticismPool = p;
 }

    @Pointcut("execution(* *.perform(..))")
    public void performance() { }

    @After("performance()")
    public void criticism() {
        int index = random.nextInt(criticismPool.length);
        System.out.println(criticismPool[index]);
    }
}
```

```
public class Singer implements Performer {

    private String name, song;

    public void setSong(String song) { this.song = song; }

    public Singer(String name, String song) {
        this.name = name; this.song = song;
    }

    @Override
    public void perform() throws PerformanceException {
        if (song == null)
            throw new PerformanceException(name + " doesn't have a son
g to sing.");

        System.out.println(name + " is singing " + song);
    }
}
```

Magician

```
public interface MindReader {
    void interceptThoughts(String thoughts);
    String getThoughts();
}
```

```
@Aspect
public class Magician implements MindReader {

    private String thoughts;
    @Override public String getThoughts() { return thoughts; }

    @Pointcut("execution(* domain.Thinker.thinkOfSomething(String)) &&
 args(thoughts)")
    public void thinking(String thoughts) { }

    @Override
    @Before("thinking(thoughts)")
    public void interceptThoughts(String thoughts) {
        System.out.println("Intercepting volunteer's thoughts " + thou
ghts);
        this.thoughts = thoughts;
    }
}
```

```
public interface Thinker {
    void thinkOfSomething(String thoughts);
}
```

```
public class Volunteer implements Thinker {

    private String thoughts;
    public String getThoughts() { return thoughts; }

    @Override
    public void thinkOfSomething(String thoughts) { this.thoughts = th
oughts; }
}
```

Chapter 3: Spring Web MVC

Exercise Framework

```
<!-- USERFORM -->
<body>
    <form:form method="POST" action="user" modelAttribute="user">
        <table>
            <tr>
                <td>Subscribe to newsletter? :</td>
                <!-- RECIEVENEWSLETTER IS PROPERTY IN MODELATTRIBUTE U
SER -->
                <td><form:checkbox path="receiveNewsletter" /></td>
            </tr>
            <tr>
                <td>Favourite Web Frameworks :</td>
                <!-- FAVFRAMEWORK IS PROPERTY IN MODELATTRIBUTE USER -
->
                <td><form:checkboxes items="${webFrameworkList}" path=
"favFramework" />
            </tr>
            <tr>
                <td><input type="submit" value="Submit" /></td>
            </tr>
        </table>
    </form:form>
</body>
```

```
<!-- RESULTVIEW -->
<body>
    Receive Newsletter : ${user.receiveNewsletter} <br>

    Favourite Web Frameworks :
    <c:forEach items="${user.favFramework}" var="currentUser">
        <br>${currentUser}
    </c:forEach>
</body>
```

```
@Controller
public class UserController {

    @RequestMapping( value = {"/user"}, method = RequestMethod.GET)
    public String showHomePage(Model model) {
        List<String> listFrameworks = (new FrameworkBean().getWebFrame
workList());
        User user = new User();
        user.setFavFramework(new String[]{listFrameworks.get(0)});
        model.addAttribute("user", user);
        model.addAttribute("webFrameworkList", listFrameworks);
        return "userForm";
    }

    @RequestMapping( value = {"/user"}, method = RequestMethod.POST)
    // @ModelAttribute comes from USERFORM.JSP
    public String onSubmit(@ModelAttribute User user, Model model) {
        model.addAttribute("user", user);
        return "resultView";
    }
}
```

```
public class FrameworkBean {

    private final List<String> items;
    public List<String> getWebFrameworkList() { return webFrameworkLis
t; }

    public FrameworkBean() {
        items = new ArrayList<>(Arrays.asList(new String[]{"Spring", "
str", "JSF"}));
    }
}
```

```
public class User {

    private boolean receiveNewsletter = true;
    private String[] favFramework;

    // GETTERS AND SETTERS
}
```

Maven and Spring Boot

Dropdown list

```
<form:form method="POST" action="beer" modelAttribute="beerCommand" >
    Choose color:
    <form:select path="colorSelected" multiple="false">
        <form:options items="${colorsList}"/>
    </form:select>
    <button type="submit">CLICK</button>
</form:form>
```

List

```
<body>
    Got beer color ${beerColor}
    <ul>
        <c:forEach items="${brands}" var="brand">
            <li>${brand}</li>
        </c:forEach>
    </ul>
</body>
```

Exercise bank (GetMapping and postMapping = MAVEN)

```
<!-- FORM -->
<body>
    <fieldset>
        <legend>Bank Account Balance</legend>
        <form:form method="POST" action="bank" modelAttribute="bankCus
tomer">
            <p> Bank user ID:
                <form:input path="id" size = "20"/>&amp;nbsp;
            </p>
            <input type="submit" value="Show Balance">
            <br>
            <form:errors path="id" cssClass="error" />
        </form:form>
    </fieldset>
</body>
```

```
<!-- RESULT -->
<body>
    <img src="${urlImg}" alt="balance"/>
    <ul>
        <li>First name: ${bankCustomer.firstname}</li>
        <li>Last name: ${bankCustomer.lastname}</li>
        <li>ID: ${bankCustomer.id}</li>
        <li>Balance:  euro
            <!-- FORMATTED BALANCE, USES GETTER INSTEAD OF PROPERTY AC
CESS -->
            <spring:bind path="bankCustomer.balance">
                ${status.value}
            </spring:bind>
        </li>
        <li>
            <!-- FORMATTED BALANCE, USES GETTER INSTEAD OF PROPERTY AC
CESS -->
            <spring:bind path="bankCustomer.balanceNoSign">
                ${status.value}
            </spring:bind>
        </li>
    </ul>
</body>
```

```
@Controller @RequestMapping("/bank")
public class BankController {

    @Autowired private BankCustomerLookup bankCustomerLookup;

    @GetMapping
    public String showHomePage(Model model) {
        model.addAttribute("bankCustomer", new BankCustomer());
        return "form";
    }

    @PostMapping
    public String processForm(@Valid BankCustomer cus, BindingResult r
es, Model mod) {
        if (res.hasErrors())
            return "form";

        BankCustomer bankCustomer = bankCustomerLookup.getCustomer(cus
.getId());
        mod.addAttribute("bankCustomer", bankCustomer);

        if (bankCustomer == null)
            return "unknownCustomer";

        if (bankCustomer.getBalance() < 0)
            return "negativeBalance";

        return "result";
    }
}
```

```
public class BankCustomer {

    @Pattern(regexp = "^[1-9]\\d{2}$", message = "must be between 100
and 999")
    private String id;

    private String firstname, lastname;

    @NumberFormat(style = NumberFormat.Style.CURRENCY)
    private double balance;

    public BankCustomer(String id, String f, String l, double b) {
        this.id = id; this.firstname = f; this.lastname = l; this.bala
nce = b;
    }

    // GETTERS AND SETTERS

    @NumberFormat(style = NumberFormat.Style.CURRENCY)
    public double getBalanceNoSign() {
        return Math.abs(balance);
    }
}
```

Format / Validation

String

```
@NotEmpty (message = "Password must not be blank.")
@Size(min = 1, max = 10, message = "Password must between 1 to 10 Char
acters.")
@Email
@Pattern(regexp = "^[a-zA-Z]+", message = "Does not match pattern")
```

Number

```
@NotNull
@Min(1)
@Max(110)
@DecimalMin(value = "20.50", message = "must be smaller than or equal
to 20.50")
@DecimalMax(value = "20.50", message = "must be greater than or equal
to 20.50")
@Range (min = 10, max = 90)
```

Example

```
// ALWAYS USE SPRING:BIND TO FORMAT THESE
@NumberFormat(pattern="#,##0.00")
private BigDecimal balance = new BigDecimal("20003000.2599");

@NumberFormat(style=Style.PERCENT)
private double percent = 0.25;

@DateTimeFormat(pattern="dd/MM/yyyy")
private Date currentDate = new Date(
```

```
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags"%>
<spring:bind path="account.balance"> ${status.value} </spring:bind>
<spring:bind path="account.percent"> ${status.value} </spring:bind>
<spring:bind path="account.currentDate"> ${status.value} </spring:bind>
```

Custom Annotation and Validator

```
<form:form method="POST" action="registration" modelAttribute="registr
ation">
    <!-- SHOW ALL ERROR MESSAGES -->
    <form:errors path="*" cssClass="error"/>
    <p>
        <label>User Name:</label>
        <form:input path="userName" size = "20"/> 
        <!-- SHOW ONLY ONE ERROR MESSAGE -->
        <form:errors path="userName" cssClass="error"/>
    </p>
    <p>
        <label>Password:</label>
        <form:password path="password" size = "20"/> 
        <form:errors path="password" cssClass="error"/>
    </p>
    <p>
        <label>Confirm Password:</label>
        <form:password path="confirmPassword" size = "20"/> 
        <form:errors path="confirmPassword" cssClass="error"/>
    </p>
    <p>
        <label>Email:</label>
        <form:input path="email" size = "20"/> 
        <form:errors path="email" cssClass="error"/>
    </p>
    <p>
        <input type="submit" value="OK" />
    </p>
</form:form>
```

```
@Controller @RequestMapping("/registration")
public class RegistrationController {

    @Autowired
    private RegistrationValidation registrationValidation;

    @RequestMapping(method = RequestMethod.GET)
    public String showHomePage(Model model) {
        model.addAttribute("registration", new Registration());
        return "registrationForm";
    }

    @RequestMapping(method = RequestMethod.POST)
    public String processRegistration(@Valid Registration reg, Binding
Result result) {
        registrationValidation.validate(reg, result);

        if (result.hasErrors())
            return "registrationForm";

        reg.setConfirmPassword(null);
        reg.setPassword(null);
        return "registrationSuccess";
    }
}
```

```
@Constraint(validatedBy = EmailValidator.class)
@Target({METHOD, FIELD})
@Retention(RUNTIME)
public @interface ValidEmail {
    String message() default "you must include a valid email";
    Class<?>[] groups() default {};
    Class<? extends Payload>[] payload() default {};
}
```

```
public class EmailValidator implements ConstraintValidator<ValidEmail,
 String> {

    @Override
    public void initialize(ValidEmail constraintAnnotation) { }

    @Override
    public boolean isValid(String value, ConstraintValidatorContext co
ntext) {
        return (value.contains("@"));
    }
}
```

```
public class RegistrationValidation implements Validator {

    @Override
    public boolean supports(Class<?> type) {
        return Registration.class.isAssignableFrom(type);
    }

    @Override
    public void validate(Object target, Errors errors) {
        Registration registration = (Registration) target;

        String userName = registration.getUserName();
        if (userName.length() < 4 || userName.length() > 15) {
            errors.rejectValue("userName", "lengthOfUser.registration.
userName",
                "username must be between 4 and 15 characters long.");
        }
        if (!(registration.getPassword()).equals(registration.getConfi
rmPassword())) {
            errors.rejectValue("password", "matchingPassword.registrat
ion.password",
                "password does not match the confirm password.");
        }
    }
}
```

```
// OTHER EXAMPLE, BUT NOT PART OF THIS EXERCISE
public class AccountValidation implements Validator {

    @Override
    public boolean supports(Class<?> type) {
        return Account.class.isAssignableFrom(type);
    }

    // VALIDATE IF PERCENTAGE IS EVEN
    @Override
    public void validate(Object target, Errors errors) {
        Account account = (Account) target;
        double percentage = account.getPercent();

        if ((int) (percentage * 100) % 2 != 0) {
            errors.rejectValue("percent", "", "The percentage must be
even");
        }
    }
}
```

```
public class Registration {

    @NumberFormat(pattern = "#,##0.00")
    @NotNull
    @Min(value = 10000, message = "Balance moet minstens 100000 zijn")
    private BigDecimal balance = new BigDecimal("20003000.2599");

    @NumberFormat(style = NumberFormat.Style.PERCENT)
    @DecimalMin(value = "0", message = "must be greater than or equals
 to 0%")
    @DecimalMax(value = "60", message = "must be les than or equals to
 60%")
    private double percent = 0.25;

    @Pattern(regexp = "^[a-zA-Z]+", message = "username must be alpha
with no spaces")
    private String userName;

    @Size(min = 4, max = 20)
    private String password;

    @NotEmpty
    private String confirmPassword;

    @ValidEmail // @NotNull @Email
    private String email;

   // GETTERS AND SETTERS
}
```

Error Messages and i18n

Validation Messages For Properties In Classes Should Be Placed In Default Package/Validationmessages.Properties

Error Messages And Translations Should Be Placed In Resources/Converter.Properties

Variables

```
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="spring" uri="http://www.springframework.org/tags" %
>

<head>
    <spring:message code="welcome.message" var="labelWelcome" />
    <spring:message code="date.format.pattern" var="dateFormatPattern"
/>
</head>
<body>
    <h2> ${labelWelcome}</h2>
    <br> Today: <fmt:formatDate value="${today}" pattern="${dateFormat
Pattern}" />
    <br> Locale : ${pageContext.response.locale}
</body>
```

```
@Bean public MessageSource messageSource() {
    ResourceBundleMessageSource messageSource = new ResourceBundleMess
ageSource();
    messageSource.setBasename("resources/messages");
    return messageSource;
}
```

Default values

```
@Controller
public class AboutController {

    // VALUE COMES FROM RESOURCES/CONVERTER.PROPERTIES
    @Value("#{ messageSource.getMessage('admin.email',null,'en')}")
    private String email;

    @RequestMapping("/about")
    public String courtReservation(Model model) {
        model.addAttribute("email", email);
        return "about";
    }
}
```

Example

```
<!-- HTML SAME AS CUSTOM ANNOTATION AND VALIDATOR -->
```

```
@Configuration @EnableWebMvc @ComponentScan("controller")
public class WebConfig extends WebMvcConfigurerAdapter {

    @Bean
    public RegistrationValidation registrationValidation() {
        return new RegistrationValidation();
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry)
{
        registry.addResourceHandler("/css/**").addResourceLocations("r
esources/css/");
    }

    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver resolver = new InternalResourceVi
ewResolver();
        resolver.setPrefix("/WEB-INF/jsp/"); resolver.setSuffix(".jsp"
);
        return resolver;
    }

    @Bean
    public MessageSource messageSource() {
        ResourceBundleMessageSource messageSource = new ResourceBundle
MessageSource();
        messageSource.setBasename("resources/converter");
        return messageSource;
    }
}
```

```
public class RegistrationValidation implements Validator {

    @Override
    public boolean supports(Class<?> type) {
        return Registration.class.isAssignableFrom(type);
    }

    @Override
    public void validate(Object target, Errors errors) {
        Registration registration = (Registration) target;

        // MATCHINGPASSWORD.REGISTRATION.PASSWORD COMES FROM RESOURCEB
UNDLE
        if (!(registration.getPassword()).equals(registration.getConfi
rmPassword())) {
            errors.rejectValue("password",
            "matchingPassword.registration.password",
            "Password and Confirm Password Not match.");
        }
    }
}
```

```
public class Registration {

    // UNDER DEFAULT PACKAGE/VALIDATIONMESSAGES.PROPERTIES AND NOT DEF
INED
    // AS BEAN IN WEBCONFIG
    @Pattern(regexp = "^[a-zA-Z]+", message = "{validation.userName.Pa
ttern.message}")
    private String userName;

    @NotEmpty
    @Size(min = 4, max = 20, message = "{validation.Size.message}")
    private String password;

    @NotEmpty
    private String confirmPassword;

    // GETTERS AND SETTERS
}
```

```
@Controller @RequestMapping("/registration")
public class RegistrationController {

    @Autowired private RegistrationValidation registrationValidation;

    @Autowired private MessageSource messageSource;

    @RequestMapping(method = RequestMethod.GET)
    public String showRegistration(Model model) {
        model.addAttribute("registration", new Registration());
        return "registrationForm";
    }

    @RequestMapping(method = RequestMethod.POST)
    public String processRegistration(@Valid Registration registration
, BindingResult result, Model model, Locale locale) {

        registrationValidation.validate(registration, result);

        if (result.hasErrors()) {
            model.addAttribute("message", new Message("error",
                                messageSource.getMessage("contact_save_f
ail", new Object[]{},locale)));
            return "registrationForm";
        }

        registration.setConfirmPassword(null);
        registration.setPassword(null);
        return "registrationSuccess";
    }
}
```

```
// VALIDATIONMESSAGES.PROPERTIES
validation.userName.Pattern.message=my validationmessage Pattern
validation.Size.message=my validationmessage Size
```

```
// CONVERTER.PROPERTIES
NotEmpty=my message notEmpty
Size=my message size
contact_save_fail=Failed saving contact
matchingPassword.registration.password=my message passwoord
```

Multiple Row

```
<!-- EDIT PAGE -->
<head>
    <!-- TRANSLATIONS COME FROM RESOURCES/CONVERTER.PROPERTIES -->
    <spring:message code="label_contact_first_name" var="labelContactF
irstName"/>
    <spring:message code="label_contact_last_name" var="labelContactLa
stName"/>
    <spring:message code="label_contact_birth_date" var="labelContactB
irthDate"/>
    <spring:message code="label_contact_description" var="labelContact
Description"/>
</head>
<form:form modelAttribute="contact" method="post" >
    <c:if test="${not empty message}">
        ${message.message}
    </c:if>

    <!-- DONT FORGET TO SEND THE ID AS WELL -->
    <form:hidden path="id" />
    <p>
        <label>${labelContactFirstName}</label>
        <form:input path="firstname" size = "20"/>
        <form:errors path="firstname" cssClass="error"/>
    </p>
    <p>
        <label>${labelContactLastName}</label>
        <form:input path="lastname" size = "20"/>
        <form:errors path="lastname" cssClass="error"/>
    </p>
    <p>
        <label>${labelContactBirthDate}</label>
        <form:input path="birthDate" size = "20"/>
        <form:errors path="birthDate" cssClass="error"/>
    </p>
    <p>
        <label>${labelContactDescription}</label>
        <form:textarea cols="60" rows="8" path="description" id="conta
ctDescription"/>
        <form:errors path="description" cssClass="error"/>
    </p>

    <form:hidden path="version" />
    <input type="submit" align="center" value="Save"/> 
    <input type="reset" align="center" value="Reset"/>
</form:form>
```

```
<!-- LIST PAGE -->
<spring:url value="/contacts/" var="contactUrl" />

<table width="50%">
    <tr>
        <th>Name</th>
        <th>Lastname</th>
        <th>Description</th>
        <th>BirthDay</th>
    </tr>
    <c:forEach items="${contactList}" var="contact" varStatus="status"
>
        <tr>
            <td>
                <!-- REFERENCE TO DETAIL FORM -->
                <a href="${contactUrl}${contact.id}">${contact.firstna
me}</a>
            </td>
            <td>${contact.lastname}</td>
            <td>${contact.description}</td>
        </tr>
    </c:forEach>
</table>
```

```
@Configuration @EnableWebMvc @ComponentScan("controller")
public class WebConfig extends WebMvcConfigurerAdapter {

    @Bean
    public ContactService contactService() {
        return new ContactServiceImpl();
    }

    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver resolver = new InternalResourceVi
ewResolver();
        resolver.setPrefix("/WEB-INF/jsp/"); resolver.setSuffix(".jsp"
);
        return resolver;
    }

    @Bean
    public MessageSource messageSource() {
        ResourceBundleMessageSource messageSource = new ResourceBundle
MessageSource();
        messageSource.setBasename("resources/converter");
        return messageSource;
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry re) {
       re.addResourceHandler("/css/**").addResourceLocations("resource
s/css/");
       re.addResourceHandler("/scripts/**").addResourceLocations("reso
urces/scripts/");
    }
}
```

```
@RequestMapping("/contacts") @Controller
public class ContactController {

    @Autowired private MessageSource messageSource;
    @Autowired private ContactService contactService;

    @RequestMapping(value = "list", method = RequestMethod.GET)
    public String listContacts(Model model) {
        model.addAttribute("contactList", contactService.findAll());
        return "contacts/list";
    }

    @RequestMapping(value = "{id}", method = RequestMethod.GET)
    public String show(@PathVariable(value = "id") Long id, Model mode
l) {
        model.addAttribute("contact", contactService.findById(id));
        return "contacts/show";
    }

    @RequestMapping(value = "edit/{id}", method = RequestMethod.GET)
    public String updateForm(@PathVariable(value = "id") long id, Mode
l model) {
        model.addAttribute("contact", contactService.findById(id));
        return "contacts/edit";
    }

    @RequestMapping(value = "edit/{id}", method = RequestMethod.POST)
    public String update(@PathVariable(value = "id") long id, @Valid C
ontact contact, BindingResult result, Model model, Locale locale) {

        if (result.hasErrors()) {
            model.addAttribute("message", new Message("error",
            messageSource.getMessage("contact_save_fail", new Object[]
{}, locale)));
            return "contacts/edit";
        }

        contact.setId(id);
        contactService.save(contact);

        model.addAttribute("contactList", contactService.findAll());
        return "contacts/list";
    }
}
```

```
public class ContactServiceImpl implements ContactService {

    private static final List<Contact> list = new ArrayList<>();

    @Override
    public List<Contact> findAll() {
        return list;
    }

    @Override
    public Contact findById(Long id) {
        return list.stream().
            filter(c -> id.compareTo(c.getId()) == 0).findFirst().orEl
se(null);
    }

    @Override
    public Contact save(Contact contact) {
        ListIterator<Contact> it = list.listIterator();
        while (it.hasNext()) {
            Contact c = it.next();
            if (contact.getId().compareTo(c.getId()) == 0) {
                it.set(contact);
            }
        }
        return contact;
    }
}
```

Security

```
<!-- WELCOME PAGE -->
<body>
    <h1>Message : ${message}</h1>
    <h1>Username : ${username}</h1>

    <c:url var="logoutUrl" value="/logout"/>
    <form action="${logoutUrl}" method="post">
        <input type="submit" value="Log out" />
        <!-- DO NOT FORGET TO SEND CSRF TOKEN AS WELL -->
        <input type="hidden" name="${_csrf.parameterName}" value="${_c
srf.token}"/>
    </form>
</body>
```

```
<!-- LOGIN PAGE -->
<body onload='document.loginForm.username.focus();'>
    <c:if test="${not empty error}">
        <div class="error">${error}</div>
    </c:if>
    <c:if test="${not empty msg}">
        <div class="msg">${msg}</div>
    </c:if>
    <form name='loginForm' action="<c:url value='/login' />" method='P
OST'>
        <table>
            <tr>
                <td>User: <input type='text' name='username' value=''>
</td>
            </tr>
            <tr>
                <td>Password: <input type='password' name='password' /
></td>
            </tr>
            <tr>
                <td><input name="submit" type="submit" value="submit"
/></td>
            </tr>
        </table>
        <!-- DO NOT FORGET TO SEND CSRF TOKEN AS WELL -->
        <input type="hidden" name="${_csrf.parameterName}" value="${_c
srf.token}"/>
    </form>
</body>
```

```
@Configuration @EnableWebMvc @ComponentScan("controller")
@Import({SecurityConfig.class})
public class WebConfig extends WebMvcConfigurerAdapter {

    @Override
    public void addViewControllers(ViewControllerRegistry registry) {
        registry.addViewController("/403").setViewName("403");
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry)
{
        registry.addResourceHandler("/css/**").addResourceLocations("r
esources/css/");
    }

    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver resolver = new InternalResourceVi
ewResolver();
        resolver.setPrefix("/WEB-INF/jsp/");
        resolver.setSuffix(".jsp");
        return resolver;
    }

    @Bean(name = "dataSource")
    public DriverManagerDataSource dataSource() {
        DriverManagerDataSource source = new DriverManagerDataSource()
;
        source.setDriverClassName("com.mysql.jdbc.Driver");
        source.setUrl("jdbc:mysql://localhost:3306/travel?"

                      "zeroDateTimeBehavior=convertToNull");
        source.setUsername("root");
        source.setPassword("root");
        return driverManagerDataSource;
    }
}
```

```
@Configuration @EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    DataSource dataSource;

    @Autowired
    public void configAuthentication(AuthenticationManagerBuilder auth
```

```
) {
        // IN MEMORY
        // enable in memory based authentication with a user named "us
er" and "admin"
        auth.inMemoryAuthentication().withUser("user").password("user"
).roles("USER")
            .and().withUser("admin").password("admin").roles("USER", "
ADMIN");

        // JPA
        auth.jdbcAuthentication().dataSource(dataSource)
            .usersByUsernameQuery(
                "select username, password, enabled from users where u
sername=?")
            .authoritiesByUsernameQuery(
                "select username, role from user_roles where username=
?");
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.formLogin().defaultSuccessUrl("/welcome");

        http.authorizeRequests()
            .antMatchers("/welcome*").access("hasRole('ROLE_USER')")
            .anyRequest().permitAll()
            .and()
            .formLogin().loginPage("/login")
            .usernameParameter("username").passwordParameter("password
")
            .and().exceptionHandling().accessDeniedPage("/403").and().
csrf();

        // OTHER EXAMPLE
        http.authorizeRequests().antMatchers("/403*").permitAll()
            .antMatchers("/*").hasRole("ADMIN");

        http.formLogin().defaultSuccessUrl("/admin").loginPage("/login
").permitAll()
            .and().exceptionHandling().accessDeniedPage("/403").and().
csrf();

        http.logout().permitAll();
    }
}
```

```
public class SpringSecurityInitializer extends
    AbstractSecurityWebApplicationInitializer { }
```

```
@Controller
public class LoginController {

    // PRINCIPAL IS THE CURRENT LOGGED IN USER
    @RequestMapping(value = "/welcome", method = RequestMethod.GET)
    public String printWelcome(Model model, Principal principal) {
        model.addAttribute("username", principal.getName());
        model.addAttribute("message", "Spring Security Custom Form exa
mple");
        return "hello";
    }

    @RequestMapping(value = "/login")
    public String login(@RequestParam(value = "error", required = fals
e) String error,
    @RequestParam(value = "logout", required = false) String logout, M
odel model) {

        if (error != null) {
            model.addAttribute("error", "Invalid username and password
!");

        if (logout != null) {
            model.addAttribute("msg", "You've been logged out successf
ully.");

        return "login";
    }
}
```

Chapter 4: Spring en JPA

Excercises

```
@Configuration @EnableTransactionManagement
public class PersistenceJPAConfig {

    @Bean
    public LocalContainerEntityManagerFactoryBean entityManagerFactory
() {
        LocalContainerEntityManagerFactoryBean em = new
            LocalContainerEntityManagerFactoryBean();
        em.setDataSource(dataSource());
        em.setPackagesToScan(new String[]{"domain"});

        JpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter
();
        em.setJpaVendorAdapter(vendorAdapter);
        em.setJpaProperties(additionalProperties());
        return em;
    }

    @Bean
    public DataSource dataSource() {
        DriverManagerDataSource dataSource = new DriverManagerDataSour
ce();
        dataSource.setDriverClassName("com.mysql.jdbc.Driver");
        dataSource.setUrl("jdbc:mysql://localhost:3306/travel");
        dataSource.setUsername("root");
        dataSource.setPassword("root");
        return dataSource;
    }

    @Bean
    public PlatformTransactionManager transactionManager(EntityManager
Factory emf) {
        JpaTransactionManager transactionManager = new JpaTransactionM
anager();
        transactionManager.setEntityManagerFactory(emf);
        return transactionManager;
    }

    @Bean
    public PersistenceExceptionTranslationPostProcessor exceptionTrans
lation() {
```

```
        return new PersistenceExceptionTranslationPostProcessor();
    }

    private Properties additionalProperties() {
        Properties properties = new Properties();
        //properties.setProperty("hibernate.hbm2ddl.auto", "create");
        properties.setProperty("hibernate.hbm2ddl.auto", "validate");
        return properties;
    }
}
```

```
@Configuration @EnableWebMvc @ComponentScan("controller")
@Import({PersistenceJPAConfig.class})
public class WebConfig extends WebMvcConfigurerAdapter {

    @Bean
    public BankCustomerDao bankCustomerDao() {
        return new JpaBankCustomerDao();
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry)
{
        registry.addResourceHandler("/css/**").addResourceLocations("r
esources/css/");
        registry.addResourceHandler("/img/**").addResourceLocations("r
esources/img/");
    }

    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver resolver = new InternalResourceVi
ewResolver();
        resolver.setPrefix("/WEB-INF/jsp/");
        resolver.setSuffix(".jsp");
        return resolver;
    }
}
```

```
@Controller @RequestMapping("/bank")
public class BankController {

    @Autowired
    private BankCustomerDao bankCustomerDao;

    @RequestMapping(method = RequestMethod.GET)
    public String showHomePage(Model model) {
        model.addAttribute("bankCustomerList", bankCustomerDao.findAll
());
        model.addAttribute("balList", bankCustomerDao.getBankCustomers
ByBalance(0));
        model.addAttribute("bankCustomer", new BankCustomer());
        return "form";
    }

    @RequestMapping(method = RequestMethod.POST)
    public String onSubmit(@Valid BankCustomer cust, BindingResult res
, Model model) {
        if (res.hasErrors()) {
            model.addAttribute("bankCustomerList", bankCustomerDao.fin
dAll());
            model.addAttribute("baList", bankCustomerDao.getBankCustom
ersByBalance(0));
            return "form";
        }

        BankCustomer currentCustomer = bankCustomerDao.get(cust.getId(
));
        model.addAttribute("customer", currentCustomer);

        if (currentCustomer == null)
            return "unknownCustomer";

        if (currentCustomer.getBalance() < 0)
            return "negativeBalance";

        return "balance";
    }
}
```

```
@Entity
@NamedQueries({
    @NamedQuery(name = "BankCustomer.getBankCustomersByBalance",
    query = "SELECT b FROM BankCustomer b WHERE b.balance >= :bankCust
omerBalance")
})
public class BankCustomer implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id @GeneratedValue(strategy = GenerationType.AUTO)
    @Min(1) @Max(999) @NotNull
    private Long id;

    private String firstname, lastname;

    @NumberFormat(style = NumberFormat.Style.CURRENCY)
    private double balance;

    @NumberFormat(style = NumberFormat.Style.CURRENCY)
    public double getBalanceNoSign() {
        return Math.abs(balance);
    }

    public BankCustomer() { }

    // GETTERS, SETTERS AND DEFAULT OVERRIDES
}
```

```
public interface BankCustomerDao extends GenericDao<BankCustomer> {

    public List<BankCustomer> getBankCustomersByBalance(double balance
);
}
```

```
public interface GenericDao<T> {

    public List<T> findAll();
    public T update(T object);
    public T get(Long id);
    public void delete(T object);
    public void insert(T object);
    public boolean exists(Long id);
}
```

```
@Transactional
public class GenericDaoJpa<T> implements GenericDao<T> {
```

```
    private Class<T> type;
    protected EntityManager em;

    public GenericDaoJpa(Class<T> type) {
        super();
        this.type = type;
    }

    @PersistenceContext
    public void setEntityManager(EntityManager em) {
        this.em = em;
    }

    @Transactional(readOnly = true)
    public T get(Long id) {
        T entity = this.em.find(this.type, id);
        return entity;
    }

    @Transactional(readOnly = true)
    public List<T> findAll() {
        return this.em.createQuery(
        "select entity from " + this.type.getName() + " entity").getRe
sultList();
    }

    @Override
    public void insert(T object) {
        em.persist(object);
    }

    @Override
    public void delete(T object) {
        em.remove(em.merge(object));
    }

    @Transactional(readOnly = true)
    @Override
    public boolean exists(Long id) {
        T entity = this.em.find(this.type, id);
        return entity != null;
    }

    @Override
    public T update(T object) {
        return em.merge(object);
    }
```

```
}
```

```
@Repository("bankCustomerDao")
public class JpaBankCustomerDao extends GenericDaoJpa<BankCustomer>
    implements BankCustomerDao {

    public JpaBankCustomerDao() {
        super(BankCustomer.class);
    }

    @Override
    @Transactional(readOnly = true)
    public List<BankCustomer> getBankCustomersByBalance(double balance
) {
        TypedQuery<BankCustomer> queryBankCustomer
            = em.createNamedQuery(
            "BankCustomer.getBankCustomersByBalance",
            BankCustomer.class);
        queryBankCustomer.setParameter(
            "bankCustomerBalance", balance);
        return queryBankCustomer.getResultList();
    }
}
```

Chapter 5: Spring en Webservices

SOAP

Server

```
@WebService(serviceName = "EquationGenerator")
public class EquationGenerator {

    @WebMethod(operationName = "generateEquation")
    public Integer[]  generateEquation(@WebParam(name = "operation") S
tring operation,
                                       @WebParam(name = "difficulty")
int difficulty) {

        int minimum = (int) Math.pow(10, difficulty - 1);
        int maximum = (int) Math.pow(10, difficulty);

        SecureRandom randomObject = new SecureRandom();
        Equation equation = new Equation(
             randomObject.nextInt(maximum - minimum) + minimum,
             randomObject.nextInt(maximum - minimum) + minimum, operat
ion);

        Integer[] result = new Integer[3];
        result[0] = equation.getLeftOperand();
        result[1] = equation.getRightOperand();
        result[2] = equation.getReturnValue();
        return result;
    }
}
```

Client

```
@Configuration @EnableWebMvc @ComponentScan("controller")
@Import({WebServiceConfig.class})
public class WebConfig extends WebMvcConfigurerAdapter {

    @Bean
    public HugeIntegerSubtractValidator hugeIntegerSubtractValidator()
 {
        return new HugeIntegerSubtractValidator();
    }

    @Override
    public void addResourceHandlers(ResourceHandlerRegistry registry)
{
        registry.addResourceHandler("/css/**").addResourceLocations("r
esources/css/");
    }

    @Bean
    public ViewResolver viewResolver() {
        InternalResourceViewResolver resolver = new InternalResourceVi
ewResolver();
        resolver.setPrefix("/WEB-INF/jsp/");
        resolver.setSuffix(".jsp");
        return resolver;
    }
}
```

```
@Configuration @EnableWs
public class WebServiceConfig {

    @Bean
    public JaxWsPortProxyFactoryBean hugeIntegerBean() throws Malforme
dURLException {
        JaxWsPortProxyFactoryBean proxy = new JaxWsPortProxyFactoryBea
n();
        // URL van de backend WSDL
        proxy.setWsdlDocumentUrl(new
URL("http://localhost:8080/webservice_HugeInteger/HugeInteger?WSDL"));

        //this is specified in the WSDL
        proxy.setServiceName("HugeInteger"); // 'service' zoeken in WS
DL XML
        proxy.setPortName("HugeIntegerPort"); // 'port' zoeken in WSDL
 XML
        proxy.setNamespaceUri("http://service/"); // 'TargetNameSpace'
 in WSDL XML

        // SET THE API CLASS
        proxy.setServiceInterface(IHugeInteger.class);
        return proxy;
    }
}
```

```
public class HugeIntegerCommand {

    @Pattern(regexp="^[0-9]+", message="{hugeIntegerCommand.number1.Pa
ttern.message}")
    private String number1;

    @Pattern(regexp="^[0-9]+", message="{hugeIntegerCommand.number2.Pa
ttern.message}")
    private String number2;

    private String operation;

    // GETTERS AND SETTERS
}
```

```
@Controller @RequestMapping("/hugeInteger")
public class HugeIntegerController {

    private HugeIntegerService service;

    @Autowired
    private HugeIntegerSubtractValidator hugeIntegerSubtractValidator;

    @RequestMapping(method = RequestMethod.GET)
    public String showHugeInteger(Model model) {
        HugeIntegerCommand hugeIntegerCommand = new HugeIntegerCommand
();
        model.addAttribute("hugeIntegerCommand", hugeIntegerCommand);
        hugeIntegerService = new HugeIntegerService();
        model.addAttribute("operationList", hugeIntegerService.getOper
ationList());
        return "hugeInteger";

    }

    @RequestMapping(method = RequestMethod.POST)
    public String resultHugeInteger(@Valid HugeIntegerCommand hugeInte
gerCommand, BindingResult result, Model model) {

        hugeIntegerSubtractValidator.validate(hugeIntegerCommand, resu
lt);

        if (result.hasErrors()) {
            model.addAttribute("operationList", service.getOperationLi
st());
            return "hugeInteger";
        }

        model.addAttribute("result",
                service.calculateHugeIntegers(hugeIntegerCommand.getNu
mber1(),
                hugeIntegerCommand.getNumber2(), hugeIntegerCommand.ge
tOperation()));

        model.addAttribute("operationList", hugeIntegerService.getOper
ationList());
        return "hugeInteger";

    }
}
```

```
public class HugeIntegerService extends SpringBeanAutowiringSupport {

    @Autowired
    private IHugeInteger hugeIntegerBean;

    private final List<String> operationList;

    public HugeIntegerService() {
        operationList = new ArrayList<>(Arrays.asList(new String[]{"+"
, "-"}));
    }

    public String calculateHugeIntegers(String first, String sec, Stri
ng operation) {
       switch (operation) {
            case "+":
                return hugeIntegerBean.add(first, sec);
            case "-":
                return hugeIntegerBean.subtract(first, sec);
        }
        return "";
    }

    public List<String> getOperationList() {
        return operationList;
    }
}
```

```
@WebService
public interface IHugeInteger {

    public String add(@WebParam(name = "first") String first,
                      @WebParam(name = "second") String second);

    public String subtract(@WebParam(name = "first") String first,
                           @WebParam(name = "second") String second);
}
```

REST

Server

```
@Configuration @EnableWebMvc @ComponentScan("controller")
public class WebConfig extends WebMvcConfigurerAdapter {

    @Bean
    public ViewResolver viewResolver() {
        return new ContentNegotiatingViewResolver();
    }

    @Override
    public void configureContentNegotiation(ContentNegotiationConfigur
er configurer) {
        configurer.defaultContentType(MediaType.APPLICATION_JSON);
        configurer.mediaType("json", MediaType.APPLICATION_JSON);
        configurer.mediaType("xml", MediaType.APPLICATION_XML);
    }
}
```

```
@RestController @RequestMapping("/fruit")
public class FruitController {

    @Autowired
    private IFruitService fruitService;

    @RequestMapping(value = "/{id}", method = RequestMethod.GET)
    public Fruit getFruitDetail(@PathVariable("id") int fruitId) {
        return fruitService.getFruitDetail(fruitId);
    }
}
```

```
public interface IFruitService {

    public Fruit getFruitDetail(int id);
}
```

```
public class FruitService implements IFruitService {

    private final Map<Integer, Fruit> fruitMap = new HashMap<Fruit>()
{{
        put(1, new Fruit(1, "orange", 10));
        put(2, new Fruit(2, "strawberry", 20));
    }};

    @Override
    public Fruit getFruitDetail(int id) {
        return fruitMap.get(id);
    }
}
```

```
@XmlRootElement(name = "Fruit")
public class Fruit {

    private int id;
    private String name;
    private int quality;

    public Fruit(int id, String name, int quality) {
        this.id = id; this.name = name; this.quality = quality;
    }

    public String getName() { return name; }
    @XmlElement(name="Name")
    public void setName(String name) { this.name = name; }

    public int getQuality() { return quality; }
    @XmlElement(name="Quality")
    public void setQuality(int quality) { this.quality = quality; }

    public int getId() { return id; }
    @XmlElement(name="Id")
    public void setId(int id) { this.id = id; }
}
```

Other example

```
@Configuration @EnableWebMvc @ComponentScan("controller")
public class WebConfig extends WebMvcConfigurerAdapter {

    @Bean
    public ViewResolver viewResolver() {
        return new ContentNegotiatingViewResolver();
    }

    @Override
    public void configureContentNegotiation(ContentNegotiationConfigur
er configurer) {
        configurer.defaultContentType(MediaType.APPLICATION_JSON);
    }
}
```

```
public interface EmpRestURIConstants {
    String DUMMY_EMP = "/rest/emp/dummy";
    String GET_EMP = "/rest/emp/{id}";
    String GET_ALL_EMP = "/rest/emps";
    String CREATE_EMP = "/rest/emp/create";
    String DELETE_EMP = "/rest/emp/delete/{id}";
}
```

```
@RestController
public class EmployeeController {

    Map<Integer, Employee> empData = new HashMap<>();

    @RequestMapping(value = DUMMY_EMP, method = RequestMethod.GET)
    public Employee getDummyEmployee() {
        Employee emp = new Employee(); emp.setId(9999); emp.setName("D
ummy");
        emp.setCreatedDate(new Date());
        empData.put(9999, emp);
        return emp;
    }

    @RequestMapping(value = GET_EMP, method = RequestMethod.GET)
    public Employee getEmployee(@PathVariable("id") int empId) {
        return empData.get(empId);
    }

    @RequestMapping(value = GET_ALL_EMP, method = RequestMethod.GET)
    public List<Employee> getAllEmployees() {
        List<Employee> emps = new ArrayList<>();
        Set<Integer> empIdKeys = empData.keySet();
        for (Integer i : empIdKeys) {
            emps.add(empData.get(i));
        }
        return emps;
    }

    @RequestMapping(value = CREATE_EMP, method = RequestMethod.POST)
    public Employee createEmployee(Employee emp) {
        emp.setCreatedDate(new Date());
        empData.put(emp.getId(), emp);
        return emp;
    }

    @RequestMapping(value = DELETE_EMP, method = RequestMethod.PUT)
    public Employee deleteEmployee(@PathVariable("id") int empId) {
        Employee emp = empData.get(empId);
        empData.remove(empId);
        return emp;
    }
}
```

```
public class Employee implements Serializable {

    private static final long serialVersionUID = 1L;
    private int id;
    private String name;
    private Date createdDate;

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    @JsonSerialize(using = DateSerializer.class)
    public Date getCreatedDate() { return createdDate; }
    public void setCreatedDate(Date createdDate) { this.createdDate =
createdDate; }
}
```