SPRING – VALIDATION AND INTERNATIONALIZATION

Change the file `form.jsp` as follows:

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://www.springframework.org/tags/form" prefix="form" %>
<%@ page language="java" contentType="text/html; charset=UTF-8"
      pageEncoding="UTF-8"%>
<%@ page
      import="java.util.*, br.edu.ufabc.classspring.dao.*,
br.edu.ufabc.classspring.model.*"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/
TR/html4/loose.dtd">
<html>
      <head>
            <title>Insertion of Students</title>
      </head>
      <body>
            <form name="insertStudent" action="insert" method="POST">
            Name: <input type="text" id="name" name="name" />
            <form:errors path="student .name" cssStyle="color:red" /> <br />
            Email: <input type="text" id="email" name="email" />
            <form:errors path="student .email" cssStyle="color:red" /> <br />
            Address: <input type="text" id="address" name="address" />
            <form:errors path="student.address" cssStyle="color:red" /> <br />
            <input type="submit" value="Insert"/>
            </form>
      </body>
</html>
```

Notice that using `<form: errors...>` in the same line of the input (before `<br />`) eases to know to which field the message is related to.

In the file `StudentController.java` modify the code as highlighted below:

```
package br.edu.ufabc.classspring.controller;

import javax.validation.Valid;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.RequestMapping;
import br.edu.ufabc.classspring.dao.StudentDAO;
import br.edu.ufabc.classspring.model.Student;

@Controller
public class StudentController {
      @RequestMapping("newStudent")
      public String form() {
            return "form";
      }

      @RequestMapping("insert")
      public String insert(@Valid Student student, BindingResult result) {
            if (result.hasErrors()) {
                  return "form";
            }
            StudentDAO dao = new StudentDAO();
            dao.insert(student);
            return "added";
      }
}
```

In **Student.java** modify the code as highlighted below:

```java
package br.edu.ufabc.classspring.model;

import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;

public class Student {
      private Long id;

      @NotNull @Size(min=5)
      private String name;

      @NotNull @Size(min=5)
      private String email;

      @NotNull @Size(min=5)
      private String address;

      public Long getId() {
            return id;
      }

      public void setId(Long id) {
            this.id = id;
      }

      public String getName() {
            return name;
      }

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

      public String getAddress() {
            return address;
      }

      public void setAddress(String address) {
            this.address = address;
      }

      public String getEmail() {
            return email;
      }

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

SOME NOTES ABOUT THE FILE `Student.Java`

VALIDATION OF THE FIELDS:

If we want to use only the annotation `@NotNull`, we will have a problem because the empty input in HTML by default submits an empty string which then go through the validation. Therefore, to work around this problem, we use the annotation `@Size` together with `@NotNull`. We still can use the annotation `@NotEmpty` which returns the answer "may not be empty" resulting in the following code:

```
public class Student {
	private Long id;

	@NotEmpty
	private String name;

	@NotEmpty
	private String email;

	@NotEmpty
	private String address;
	.
	.
	.
```

VALIDATION OF EMAIL:

If we want to validate the field e-mail by verifying if it's a valid email, we can use an annotation that verifies a regular expression. See the example:

```
public class Student {
	private Long id;

	@NotEmpty
	private String name;

	@Pattern(regexp="^[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@[A-Za-z0-9]+(\\.[A-Za-
z0-9]+)*(\\.[A-Za-z]{2,})$")
	private String email;

	@NotEmpty
	private String address;
	.
	.
	.
```

Regular expressions are studied in our Formal Languages and Automata Theory Course. Our validation verifies if: (a) it there exists a word (or two words with a dot between them) before the symbol @; (b) it there exists the symbol @; (c) it there exists one word (or still two or three words separated by dots) after the symbol @. See the detailed explanation:

^	= beginning of line.
[_A-Za-z0-9-]+	= this word must be formed by the characters between the brackets and must have one or more characters (indicated by the symbol +).
(\\.[_A-Za-z0-9-]+)*	= this word is optional, which is indicated by the parentheses and by the symbol *. This word must start with a dot followed by one or more of the characters between the brackets.
@	= it must have a symbol "@".
[A-Za-z0-9]+	= this word must be formed by the characters between the brackets and must have one or more characters (indicated by the symbol +).
(\\.[A-Za-z0-9]+)*	= this word is optional, which is indicated by the parentheses including everything and followed by the symbol *. This word must start with a dot "." followed by one or more of the characters between the brackets.
(\\.[A-Za-z]{2,})	= this word must start with a dot and must be formed by the characters between the brackets, with minimum length of 2.
$	= end of line.

SPRING – INTERNATIONALIZATION

Modify the file `Spring-context.xml` as highlighted below:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:mvc="http://www.springframework.org/schema/mvc"
xsi:schemaLocation="http://www.springframework.org/schema/mvc
    http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd
    http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-3.0.xsd">
  <context:component-scan base-package="br.edu.ufabc.classspring" />
  <mvc:annotation-driven />
  <bean id="messageSource"
class="org.springframework.context.support.ReloadableResourceBundleMessageSource">
      <property name="basename" value="/WEB-INF/messages" />
  </bean>

  <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
    <property name="prefix" value="/WEB-INF/views/"/>
    <property name="suffix" value=".jsp"/>
  </bean>
</beans>
```

messages.properties

```
#override @NotEmpty default message
#override @Pattern default message
NotEmpty.student.name=Name must be filled!
NotEmpty.student.email=Email must be filled!
NotEmpty.student.address=Address must be filled!
Pattern.student.email=Email must have the format: word@word.word!
```

Modify the file `Student.java` as highlighted below:

```
package br.edu.ufabc.classspring.model;

import javax.validation.constraints.Pattern;
import org.hibernate.validator.constraints.NotEmpty;

public class Student {
	private Long id;
	@NotEmpty(message="{student.name}")
	private String name;
	@NotEmpty(message="{student.email}")
	@Pattern(regexp="^[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@[A-Za-z0-9]+(\\.[A-Za-
z0-9]+)*(\\.[A-Za-z]{2,})$",message="{student.email}")
	private String email;
	@NotEmpty(message="{student.address}")
	private String address;

	public Long getId() {
		return id;
	}

	public void setId(Long id) {
		this.id = id;
	}

	public String getName() {
		return name;
	}

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

	public String getAddress() {
		return address;
	}

	public void setAddress(String address) {
		this.address = address;
	}

	public String getEmail() {
		return email;
	}

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