[Bài đọc] Converter

Spring Converter là một đối tượng được dùng để chuyển đổi kiểu dữ liệu này sang kiểu dữ liệu khác. Chẳng hạn, chúng ta có thể biểu diễn cùng một ngày theo những định dạng khác nhau, chẳng hạn như: *“December 25, 2016,” 12/25/2016,” “2016-12-25”*. Mặc định, Spring sẽ sử dụng định dạng ngày của địa phương hiện tại (current locale). Chẳng hạn, định dạng của Hoa Kỳ sẽ là *MM/dd/yyyy*. Trong trường hợp chúng ta muốn thay đổi định dạng của ngày khi liên kết một trường string với một đối tượng LocalDate thì chúng ta cần viết một converter để chuyển đổi từ string sang ngày.

*Lưu ý: Lớp java.time.LocalDate là một lớp mới được bổ sung cho Java 8, thay thế cho lớp java.util.Date. Chúng ta nên sử dụng bộ API mới về ngày tháng và thời gian, thay vì sử dụng các lớp Date và Calendar như cũ.*

Để tạo một converter, chúng ta viết một lớp Java triển khai interface *org.springframework.core.convert.converter.Converter*. Đây là một interface generic:

**public interface** Converter<S, T>

S đại diện cho kiểu dữ liệu nguồn, và T đại diện cho kiểu dữ liệu đích. Chẳng hạn, nếu muốn tạo một converter để chuyển đổi từ kiểu Long sang kiểu LocalDate thì chúng ta có thể khai báo một lớp như sau:

**public class** MyConverter **implements** Converter<Long, LocalDate> {  
  
}

Bên trong lớp này, chúng ta triển khai phương thức convert() của interface Converter.

T convert(S source)

Trong ví dụ sau, chúng ta định nghĩa một converter để có thể làm việc với bất kỳ định dạng ngày nào.

**Lớp StringToLocalDateConverter:**

package com.codegym.cms.converter;  
  
import org.springframework.core.convert.converter.Converter;  
  
import java.time.LocalDate;  
import java.time.format.DateTimeFormatter;  
import java.time.format.DateTimeParseException;  
  
public class StringToLocalDateConverter implements Converter<String, LocalDate> {  
 private String datePattern;  
  
 public StringToLocalDateConverter(String datePattern) {  
 this.datePattern = datePattern;  
 }  
  
 @Override  
 public LocalDate convert(String s) {  
 try {  
 return LocalDate.*parse*(s, DateTimeFormatter.*ofPattern*(datePattern));  
 } catch (DateTimeParseException e) {  
 // the error message will be displayed in <form:errors>  
 throw new IllegalArgumentException("invalid date format. Please use this pattern\""  
 + datePattern + "\"");  
 }  
 }  
}

Converter này sẽ chuyển đổi một dữ liệu Spring sang kiểu LocalDate theo định dạng ngày được truyền vào từ constructor.

Để sử dụng converter trong một ứng dụng Spring MVC, chúng ta cần tạo một bean conversionService trong file cấu hình của Spring MVC. Tên lớp của bean phải là org.springframework.context.support.ConversionServiceFactoryBean. Bean này cần chứa một thuộc tính converters, trong đó liệt kê tất cả các converter mà chúng ta đã định nghĩa. Đoạn mã sau minh hoạ việc đăng ký converter SringToLocalDateConverter:

**Lớp ApplicationConfig:**

@Override  
public void addFormatters(FormatterRegistry registry) {  
 StringToLocalDateConverter stringToLocalDateConverter = new  
 StringToLocalDateConverter("MM/dd/yyyy");  
 registry.addConverter(stringToLocalDateConverter);  
}

Ví dụ sau đây sẽ sử dụng *StringToLocalDateConverter* để chuyển một String thành giá trị của trường birthDate của đối tượng Employee.

**Lớp Employee:**

package com.codegym.cms.model;  
  
import java.io.Serializable;  
import java.time.LocalDate;  
  
public class Employee implements Serializable {  
 private static final long *serialVersionUID* = -908L;  
 private long id;  
 private String firstName;  
 private String lastName;  
 private LocalDate birthDate;  
 private int salaryLevel;  
  
 public Employee() {  
 }  
  
 public Employee(long id, String firstName, String lastName, LocalDate birthDate, int salaryLevel) {  
 this.id = id;  
 this.firstName = firstName;  
 this.lastName = lastName;  
 this.birthDate = birthDate;  
 this.salaryLevel = salaryLevel;  
 }  
  
 public long getId() {  
 return id;  
 }  
  
 public void setId(long id) {  
 this.id = id;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public LocalDate getBirthDate() {  
 return birthDate;  
 }  
  
 public void setBirthDate(LocalDate birthDate) {  
 this.birthDate = birthDate;  
 }  
  
 public int getSalaryLevel() {  
 return salaryLevel;  
 }  
  
 public void setSalaryLevel(int salaryLevel) {  
 this.salaryLevel = salaryLevel;  
 }  
}

**Lớp EmployeeController:**

package com.codegym.cms.controller;  
  
import com.codegym.cms.model.Employee;  
import org.springframework.stereotype.Controller;  
import org.springframework.ui.Model;  
import org.springframework.validation.BindingResult;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.ModelAttribute;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.servlet.ModelAndView;  
  
@Controller  
public class EmployeeController {  
 @GetMapping("/add-employee")  
 public ModelAndView inputEmployee() {  
 ModelAndView modelAndView = new ModelAndView("/employee/EmployeeForm");  
 modelAndView.addObject("employee", new Employee());  
 return modelAndView;  
 }  
  
 @RequestMapping(value = "/save-employee")  
 public String saveEmployee(@ModelAttribute Employee employee, BindingResult bindingResult, Model model) {  
 if (bindingResult.hasErrors()) {  
 model.addAttribute("errorMessage", bindingResult.getAllErrors());  
 return "/employee/EmployeeForm";  
 }  
  
 // save employee here...  
  
 model.addAttribute("employee", employee);  
 return "/employee/EmployeeDetail";  
 }  
}

Lớp EmployeeController có hai phương thức để xử lý request là *inputEmployee()* và *saveEmployee()*. Phương thức *inputEmployee()* hiển thị view EmployeeForm. Phương thức *saveEmployee()* lấy về đối tượng Employee được tạo ra khi form được đẩy lên. Converter StringToLocalDateConverter đã hỗ trợ việc chuyển đổi string sang LocalDate, do đó trong lớp controller, chúng ta không cần phải tự làm việc này.

Đối số BindingResult của phương thức saveEmployee() chứa tất cả các lỗi binding được sinh ra bởi Spring.  Phương thức này sử dụng BindingResult để ghi nhận tất cả các lỗi binding nếu có. Các lỗi binding này có thể được hiển thị trên form, ví dụ:

**Trang EmployeeForm.html:**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Employee Form</title>  
 <style>  
 .error{  
 color: red;  
 }  
 </style>  
  
</head>  
<body>  
<h1 class="title">Employee Form</h1>  
  
<th:block th:if="${errorMessage}">  
 <p class="error" th:text="${errorMessage}"></p>  
</th:block>  
  
  
<form th:action="@{/save-employee}" th:object="${employee}" method="post">  
 <table>  
 <tr>  
 <td>First Name:</td>  
 <td><input type="text" th:field="\*{firstName}" /></td>  
 </tr>  
 <tr>  
 <td>Last Name:</td>  
 <td><input type="text" th:field="\*{lastName}"/></td>  
 </tr>  
 <tr>  
 <td>Birth Date:</td>  
 <td>  
 <input type="text" th:field="\*{birthDate}"/>  
 </td>  
 </tr>  
 <tr>  
 <td>Salary Level:</td>  
 <td><input type="text" th:field="\*{salaryLevel}"/></td>  
 </tr>  
 <tr>  
 <td></td>  
 <td><input type="submit" value="Create Employee"></td>  
 </tr>  
 </table>  
</form>  
</body>  
</html>

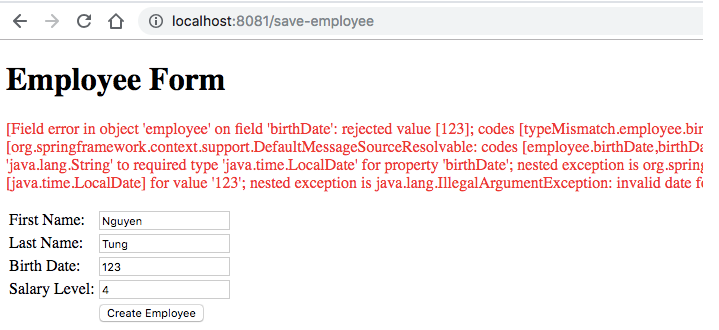
**Trang EmployeeDetail.html:**

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
 <meta charset="UTF-8">  
 <title>Employee Detail</title>  
  
</head>  
<body>  
<h1>Employee Detail </h1>  
  
<div>  
 First Name: <span th:text="${employee.firstName}"></span><br>  
 Last Name : <span th:text="${employee.lastName}"></span><br>  
 Birth Date: <span th:text="${employee.birthDate}"></span><br>  
 Salary Level: <span th:text="${employee.salaryLevel}"></span><br>  
</div>  
  
</body>  
</html>

Chúng ta có thể đi đến đường dẫn sau để thử nghiệm converter này:

http://localhost:8080/add-employee

Thử nhập vào một ngày bất hợp lệ, chúng ta sẽ được chuyển đến chính trang form Employee này kèm theo một thông báo lỗi xuất hiện trên form.



Bây giờ, nhập vào một ngày hợp lệ, chúng ta sẽ chuyển đến trang EmployeeDetail:

