Spring MVC的自定义数据转换器

使用Spring MVC进行开发时，请求中的数据会自动封装到业务方法的形参中，此工作是由HandlerAdapter组件来完成的。

HTTP请求中的所有参数都是String类型的，如果业务方法中的参数是String或者int类型，HandlerAdapter可以自动完成数据类型的转换，但是如果参数是其他特殊的数据类型，比如Date类型，HandlerAdapter是不会将String类型转为Date类型的，我们手动来编写自定义的数据转换器来完成特定的需求。需要使用Converter接口来协助Spring MVC完成数据类型的转换。

- 创建DateConverter转换器，实现org.springframework.core.convert.Converter接口。

```
package com.southwind.converter;

import org.springframework.core.convert.converter.Converter;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateConverter implements Converter<String,Date> {

    private String pattern;

    public DateConverter(String pattern){
        this.pattern = pattern;
    }

    @Override
    public Date convert(String source) {
        SimpleDateFormat simpleDateFormat = new
SimpleDateFormat(this.pattern);
        Date date = null;
        try {
            date = simpleDateFormat.parse(source);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        return date;
    }
}
```

- springmvc.xml中配置conversionService bean。

```
<bean id="conversionService"
class="org.springframework.context.support.ConversionServiceFactoryBean">
    <property name="converters">
        <list>
            <bean class="com.southwind.converter.DateConverter">
                <constructor-arg type="java.lang.String" value="yyyy-MM-dd">
</constructor-arg>
            </bean>
        </list>
    </property>
</bean>

<mvc:annotation-driven conversion-service="conversionService">
</mvc:annotation-driven>
```

bean的类名称必须是org.springframework.context.support.ConversionServiceFactoryBean.

bean必须包含一个converters属性，它将列出在应用程序中用到的所有定制Converter，将我们自定义的DateConverter添加到converters中，通过有参构造函数来创建DateConverter的实例化对象。

同时annotation-driver元素的conversion-service属性必须是上面配置好的bean的id。

```
@Controller
public class TestHandler {

    @RequestMapping("/dateConverterTest")
    public String index(Date date){
        System.out.println(date);
        return "index";
    }

}
```

除了Date类型的转换，还可以自定义数据格式，比如注册一个Student，前端页面按照“id-name-age”的形式输入String类型的数据，通过转换器可以将该String类型的数据直接转为Student对象。

- 创建Student实体类。

```
package com.southwind.entity;

public class Student {
    private int id;
    private String name;
    private int age;

    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;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }
}
```

- 创建addStudent.jsp

```
<%--
  Created by IntelliJ IDEA.
  User: southwind
  Date: 2019-01-21
  Time: 20:48
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <form action="studentConveterTest" method="post">
        学生信息:<input type="text" name="student"/><span style="font-size:
13px">(id-name-age)</span><br/>
        <input type="submit" value="提交"/>
    </form>
</body>
</html>
```

- 业务方法

```
@RequestMapping("/studentConveterTest")
@ResponseBody
public Student studentConverter(Student student){
    return student;
}
```

- 创建StudentConverter转换器。

```
package com.southwind.converter;

import com.southwind.entity.Student;
import org.springframework.core.convert.converter.Converter;

public class StudentConverter implements Converter<String, Student> {

    @Override
    public Student convert(String source) {
        String[] args = source.split("-");
        Student student = new Student();
        student.setId(Integer.parseInt(args[0]));
        student.setName(args[1]);
        student.setAge(Integer.parseInt(args[2]));
        return student;
    }
}
```

- springmvc.xml中配置StudentConverter转换器。

```
<bean id="conversionService"
class="org.springframework.context.support.ConversionServiceFactoryBean">
    <property name="converters">
        <list>
            <!-- 日期转换器 -->
            <bean class="com.southwind.converter.DateConverter">
                <constructor-arg type="java.lang.String" value="yyyy-MM-dd">
</constructor-arg>
            </bean>
            <!-- Student转换器 -->
            <bean class="com.southwind.converter.StudentConverter"></bean>
        </list>
    </property>
</bean>

<mvc:annotation-driven conversion-service="conversionService">
    <mvc:message-converters>
        <bean
class="org.springframework.http.converter.StringHttpMessageConverter">
```

```
            <property name="supportedMediaTypes"
value="text/html;charset=UTF-8"></property>
        </bean>
        <bean
class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter4">
</bean>
    </mvc:message-converters>
</mvc:annotation-driven>
```

Spring MVC对RESTful架构的支持

REST（RepresEntational State Transfer）资源表现层状态转化，是目前流行的一种互联网软件架构，结构清晰，符合标准，易于理解，便于扩展。

资源 Resources

网络中的一个实体，或者说是网络上的一个具体信息，可以是一段文本、一张图片、一首歌曲、一种服务，总之就是一个具体的存在，可以用一个URI（统一资源定位符）指向它，每种资源对应一个特定的URI。要获取这个资源，访问它的URI就可以，URI即每个资源独一无二的标识。

表现层 Representation

把资源具体呈现出来的形式叫做它的表现层，比如文本可以用txt格式表现，也可以用html格式、xml格式、json格式。

状态转化 State Transfer

每发出一个请求，就代表了客户端和服务端的一次交互过程，HTTP协议是一个无状态协议，即所有的状态都保存在服务端。因此，如果客户端想要操作服务端，必须通过某种手段，让服务端发生“状态转化”，而这种转化是建立在表现层之上的，也就是“表现层状态转化”。

特点

- URL更加简洁，将参数通过URL传到服务端。

  传统方式：http://localhost:8080/query?id=1

  REST:http://localhost:8080/query/1
- 有利于不同系统之间的资源共享，只需要遵守一定的规范，不需要做其他配置就可以实现资源共享。

RESTful具体来讲就是四种表现形式，HTTP协议中的四种请求类型分别表示四种基本操作。

- GET 用来获取资源
- POST 用来创建资源
- PUT 用来修改资源
- DELETE 用来删除资源

form表单只支持GET和POST，不支持PUT、DELETE，如何解决这一问题?

通过添加HiddenHttpMethodFilter过滤器，可以讲POST转为PUT或者DELETE。

过滤器的实现原理大致如下：检测请求参数重是否包含_method，如果包含则获取该参数的值，然后判断是哪种操作后继续传递。

- form表单添加隐藏域标签，name=_method,value=PUT/DELETE。

```
<%--
  Created by IntelliJ IDEA.
  User: southwind
  Date: 2019-01-21
  Time: 21:45
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <form action="httpPut" method="post">
        <input type="hidden" name="_method" value="PUT"/>
        <input type="submit" value="修改"/>
    </form>
    <hr/>
    <form action="httpDelete" method="post">
        <input type="hidden" name="_method" value="DELETE"/>
        <input type="submit" value="删除"/>
    </form>
</body>
</html>
```

- web.xml中配置HiddenHttpMethodFilter。

```
<filter>
    <filter-name>HiddenHttpMethodFilter</filter-name>
    <filter-
class>org.springframework.web.filter.HiddenHttpMethodFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>HiddenHttpMethodFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

- 业务方法

```
@PutMapping("/httpPut")
@ResponseBody
public String httpPut(){
    return "PUT";
}

@DeleteMapping("/httpDelete")
@ResponseBody
public String httpDelete(){
    return "DELETE";
}
```

- 传参

```
@PutMapping("/httpPut/{id}/{name}")
@ResponseBody
public String httpPut(@PathVariable("id") int id,@PathVariable("name")
String name){
    return "id:"+id+"-name:"+name;
}
```