需要导入serverlet-api maven

相比之前少了map这一层，直接操作request，效率更高，这个request对象是handleradapt来创建的，在我们执行这个handler之前，handleradapt会检测你这个方法是需要这么个参数就会给你创建一个

HttpServletRequest

Spring MVC 可以在业务方法中直接获取 Servlet 原生 Web 资源，只需要在方法定义时添加 HttpServletRequest 入参即可，在方法体中直接使用 request 对象。

```
@RequestMapping("/request")
public String request(HttpServletRequest request){
    User user = new User();
    user.setId(1);
    user.setName("张三");
    request.setAttribute("user",user);
    return "show";
}
```

@ModelAttribute

Spring MVC 还可以通过 @ModelAttribute 注解的方式添加业务数据，具体使用步骤如下：

- 定义一个方法，该方法用来放好要填充到业务数据中的对象。
- 给该方法添加 @ModelAttribute 注解，不是响应请求的业务方法。 是返回对象的方法

```
@RequestMapping("/modelAttribute")
public String modelAttribute(){
  return "show";
}

@ModelAttribute
public User getUser(){
  User user = new User();
  user.setId(2);
  user.setName("李四");
  return user;
}
```

@ModelAttribute 注解的作用是，当 Handler 接收到一个客户端请求之后，无论调用哪个业务方法，都会优先调用被 @ModelAttribute 注解修饰的方法，并将其返回值做为业务数据，再进入到业务方法，此时业务方法只需要返回视图信息即可，不需要返回业务数据，即使返回业务数据，也会被 @ModelAttribute 注解修饰的方法返回的数据所覆盖。 优先级最高

域对象中存值都是以 key-value 形式存储的，那么此时的 key 值是什么？默认值是业务数据对应的类的类名首字母小写之后的结果。

```
<%--
  Created by IntelliJ IDEA.
  User: southwind
  Date: 2020-02-08
```

```
  Time: 18:05
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page isELIgnored="false" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>show</h1>
    ${requestScope.user}
</body>
</html>
```

如果 getUser 没有返回值，则必须手动在该方法中填充业务数据，使用 Map 或者 Model 均可。

```
@ModelAttribute
public void getUser(Model model){
    User user = new User();
    user.setId(1);
    user.setName("张三");
    model.addAttribute("user",user);
}
```

如果同时存在两个 @ModelAttribute 注解方法。

```
@ModelAttribute
public void getUser(Model model){
    User user = new User();
    user.setId(1);
    user.setName("张三");
    model.addAttribute("user",user);
}

@ModelAttribute
public User getUser(){
    User user = new User();
    user.setId(2);
    user.setName("李四");
    return user;
}
```

直接给 Model 对象进行装载的方法优先级更高。

这里之前的是把数据绑定到request域

业务数据绑定到 session 域对象

- HttpSession

```
@RequestMapping("/session")
public String session(HttpSession session){
    User user = new User();
    user.setId(1);
    user.setName("张三");
    session.setAttribute("user",user);
    return "show";
}
```

- JSP

```
<%--
  Created by IntelliJ IDEA.
  User: southwind
  Date: 2020-02-08
  Time: 18:05
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ page isELIgnored="false" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <h1>show</h1>
    ${sessionScope.user}      从session域中取
</body>
</html>
```

@SessionAttribute

@SessionAttribute 注解不是给方法添加的，而是给类添加。

```
@SessionAttributes(value = "user")
public class ViewHandler {
  ...
}
```

只要给类添加了 @SessionAttributes 注解之后，无论类中的哪个业务方法被访问，将业务数据绑定到 request 域对象的同时，也会将业务数据绑定到 session 域对象中，也就是说 request 和 session 对象会同时存在业务数据，前提是 request 域中的 key 值需要和 @SessionAttributes 注解中的 value 值一致。

@SessionAttributes 除了可以通过 key 值绑定，也可以通过业务数据的数据类型进行绑定。

```
@Controller
@SessionAttributes(types=User.class)
public class ViewHandler {
  ...
}
```

@SessionAttributes 可以同时绑定多个业务数据。

```
@Controller
@SessionAttributes(value={"user","address"})
public class ViewHandler {
  ...
}
```

或者

```
@Controller
@SessionAttributes(types={User.class,Address.class})
public class ViewHandler {
  ...
}
```

springmvc进行web开发的时候前端页面会进行自动的封装到业务方法的参数中这项工作是由handleradapt组件进行完成的，我们知道http表单中请求参数是string等类型，这个handleradapt可以自动处理，但是如果是其他类型比如date类型，handleradapte是无法将string转成date类型需要自己写个转换器辅助完成类型转换

Spring MVC自定义数据类型转换器

1、创建 DateConverter 类，并实现 org.springframework.core.convert.converter.Converter 接口，这样它就成为了一个自定义数据类型转换器，需要指定泛型 <String,Date>，表示将 String 类型转为 Date 类型。

```
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 s) {
```

```
        SimpleDateFormat simpleDateFormat = new
SimpleDateFormat(this.pattern);
        try {
            return simpleDateFormat.parse(s);
        } catch (ParseException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

2、在 springmvc.xml 中配置 conversionService bean，这个 bean 是 org.springframework.context.support.ConversionServiceFactoryBean 的实例化对象，同时 bean 中必须包含一个 converters 属性，在其中注册所有需要使用的自定义转换器。

```
<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>
```

Student

```
package com.southwind.entity;

import lombok.Data;

@Data
public class Student {
    private Integer id;
    private String name;
    private Integer age;
}
```

StudentConverter

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

Handler

```
@RequestMapping("/student")
@ResponseBody
public Student student(Student student, HttpServletResponse response){
  response.setCharacterEncoding("UTF-8");
  return student;
}
```

如果需要将业务数据转换成 JSON，中文乱码需要在业务方法中通过设置 response 的编码格式来解决，springmvc.xml 中的 bean 不起作用，如果不需要将业务数据转成 JSON，springmvc.xml 的配置可以中文乱码的处理。

```
<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>
      <bean class="com.southwind.converter.StudentConverter"></bean>
    </list>
  </property>
</bean>

<mvc:annotation-driven conversion-service="conversionService">
  <!-- 消息转换器 -->
  <mvc:message-converters>
    <bean
class="org.springframework.http.converter.StringHttpMessageConverter">
      <property name="supportedMediaTypes" value="text/json;charset=UTF-8">
</property>
    </bean>
```

```
    <!-- fastjson -->
    <bean
class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter4">
</bean>
  </mvc:message-converters>
</mvc:annotation-driven>
```

Spring MVC与RESTful的集成

- 什么是 RESTful?

RESTful 是当前比较流行的一种互联网软件架构模型，通过统一的规范完成不同终端的数据访问和交互，REST 全称是 Representaional State Transfer（资源表现层状态转换）。

RESTful 的优点：结构清晰、有统一的标准、扩展性好。

- Resources

资源指的是网络中的某个具体文件，类型不限，可以是文本、图片、视频、音频、数据流等，是网络中真实存在的一个实体。如何获取它？可以通过统一资源定位符找到这个实体，URI，每个资源都有一个特定的 URI，通过 URI 就可以找到一个具体的资源。

- Pepresentation

资源表现层，资源的具体表现形式，例如一段文字，可以使用 TXT、HTML、XML、JSON 等不同的形式来来描述它。

- State Transfer

状态转化是指客户端和服务端之间的数据交互，因为 HTTP 请求不能传输数据的状态，所有的状态都保存在服务端，如果客户端希望访问服务端的数据，就需要使其发生状态改变，同时这种状态转化是建立在表现层，完成转换就表示资源的表现形式发生了改变。

RESTful 的概念比较抽象，特点有两个：

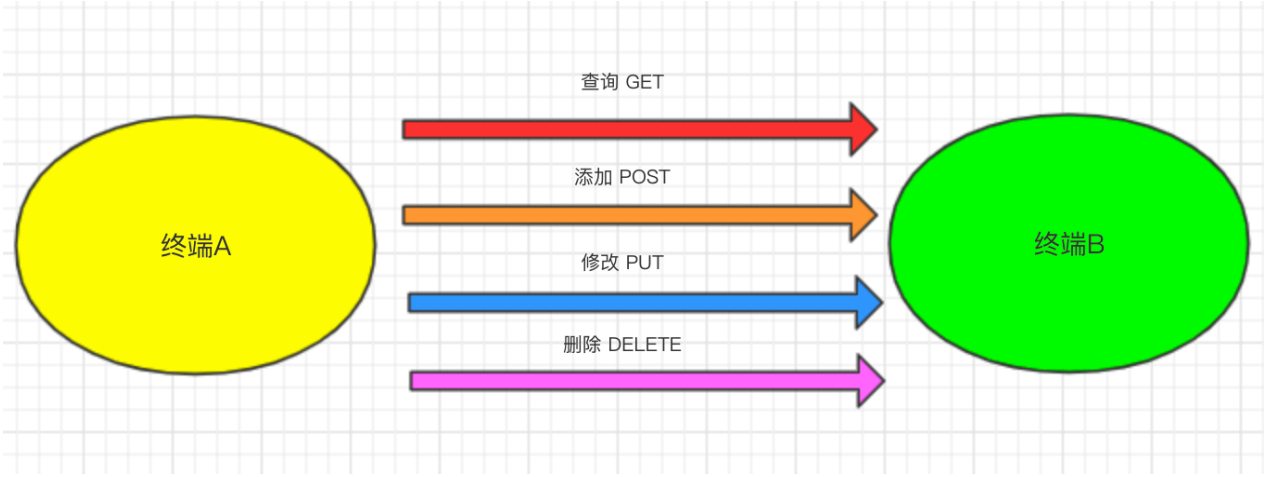
1、URL 传参更加简洁

- 传统形式 URL：http://localhost:8080/findById?id=1
- RESTful URL：http://localhost:8080/findById/1

2、完成不同终端之间的资源共享，RESTful 提供了一套规范，不同终端之间只需要遵守该规范，就可以实现数据交互。

RESTful 具体来讲就是四种表现形式，HTTP 协议中四种请求类型（GET、POST、PUT、DELETE）分别表示四种常规操作，CRUD

- 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



两个终端要完成数据交互，基于 RESTful 的方式，增删改查操作分别需要使用不同的 HTTP 请求类型来访问。