基于 restful 风格 spring 集成 Mybatis 的通用 mapper 以及分页助手

一、基本环境

IDE：Eclipse Java EE IDE for Web Developers Version: Neon Release (4.6.0)

JDK：1.8

Mysql：mysql-5.6.24

maven：maven3.3.9

Tomcat：tomcat9

二、项目基本搭建

1、 建立数据库

首先建立一个测试数据库 mybatis,在该数据库中建立一个数据表 tb_user,创建语句如下：

```
CREATE TABLE `tb_user` (
  `id` bigint(20) NOT NULL AUTO_INCREMENT,
  `user_name` varchar(100) DEFAULT NULL COMMENT '用户名',
  `password` varchar(100) DEFAULT NULL COMMENT '密码',
  `name` varchar(100) DEFAULT NULL COMMENT '姓名',
  `age` int(10) DEFAULT NULL COMMENT '年龄',
  `sex` tinyint(1) DEFAULT NULL COMMENT '性别，1 男性，2 女性',
  `birthday` date DEFAULT NULL COMMENT '出生日期',
  `created` datetime DEFAULT NULL COMMENT '创建时间',
  `updated` datetime DEFAULT NULL COMMENT '更新时间',
  PRIMARY KEY (`id`),
  UNIQUE KEY `username` (`user_name`)
) ENGINE=InnoDB AUTO_INCREMENT=14 DEFAULT CHARSET=utf8
```

2、 建立 maven 的父项目

建立一个 pom 工程：kang-parent，其 pom.xml 内容如下：

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<groupId>com.kang.parent</groupId>
	<artifactId>kang-parent</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>pom</packaging>

	<!-- 集中定义依赖版本号 -->
	<properties>
		<junit.version>4.10</junit.version>
		<spring.version>4.1.3.RELEASE</spring.version>
		<mybatis.version>3.2.8</mybatis.version>
		<mybatis.spring.version>1.2.2</mybatis.spring.version>
		<mybatis.paginator.version>1.2.15</mybatis.paginator.version>
		<mysql.version>5.1.32</mysql.version>
```

```
        <slf4j.version>1.6.4</slf4j.version>
        <jackson.version>2.4.2</jackson.version>
        <druid.version>1.0.9</druid.version>
        <httpclient.version>4.3.5</httpclient.version>
        <jstl.version>1.2</jstl.version>
        <servlet-api.version>2.5</servlet-api.version>
        <jsp-api.version>2.0</jsp-api.version>
        <joda-time.version>2.5</joda-time.version>
        <commons-lang3.version>3.3.2</commons-lang3.version>
        <commons-io.version>1.3.2</commons-io.version>
    </properties>

    <dependencyManagement>
        <dependencies>
            <!-- 单元测试 -->
            <dependency>
                <groupId>junit</groupId>
                <artifactId>junit</artifactId>
                <version>${junit.version}</version>
                <scope>test</scope>
            </dependency>

            <!-- Spring -->
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-beans</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-webmvc</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-jdbc</artifactId>
                <version>${spring.version}</version>
            </dependency>
            <dependency>
```

```xml
        <groupId>org.springframework</groupId>
        <artifactId>spring-aspects</artifactId>
        <version>${spring.version}</version>
</dependency>

<!-- Mybatis -->
<dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>${mybatis.version}</version>
</dependency>
<dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>${mybatis.spring.version}</version>
</dependency>

<!-- MySql -->
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>${mysql.version}</version>
</dependency>

<dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <version>${slf4j.version}</version>
</dependency>

<!-- Jackson Json 处理工具包 -->
<dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>${jackson.version}</version>
</dependency>

<!-- 连接池 -->
<dependency>
        <groupId>com.jolbox</groupId>
        <artifactId>bonecp-spring</artifactId>
        <version>0.8.0.RELEASE</version>
</dependency>
```

```xml
<!-- httpclient -->
<dependency>
	<groupId>org.apache.httpcomponents</groupId>
	<artifactId>httpclient</artifactId>
	<version>${httpclient.version}</version>
</dependency>

<!-- JSP 相关 -->
<dependency>
	<groupId>jstl</groupId>
	<artifactId>jstl</artifactId>
	<version>${jstl.version}</version>
</dependency>
<dependency>
	<groupId>javax.servlet</groupId>
	<artifactId>servlet-api</artifactId>
	<version>${servlet-api.version}</version>
	<scope>provided</scope>
</dependency>
<dependency>
	<groupId>javax.servlet</groupId>
	<artifactId>jsp-api</artifactId>
	<version>${jsp-api.version}</version>
	<scope>provided</scope>
</dependency>

<!-- 时间操作组件 -->
<dependency>
	<groupId>joda-time</groupId>
	<artifactId>joda-time</artifactId>
	<version>${joda-time.version}</version>
</dependency>

<!-- Apache 工具组件 -->
<dependency>
	<groupId>org.apache.commons</groupId>
	<artifactId>commons-lang3</artifactId>
	<version>${commons-lang3.version}</version>
</dependency>
<dependency>
	<groupId>org.apache.commons</groupId>
	<artifactId>commons-io</artifactId>
	<version>${commons-io.version}</version>
</dependency>
```

```
		</dependencies>
	</dependencyManagement>

	<build>
		<finalName>${project.artifactId}</finalName>
		<plugins>
			<!-- 资源文件拷贝插件 -->
			<plugin>
				<groupId>org.apache.maven.plugins</groupId>
				<artifactId>maven-resources-plugin</artifactId>
				<version>2.7</version>
				<configuration>
					<encoding>UTF-8</encoding>
				</configuration>
			</plugin>
			<!-- java 编译插件 -->
			<plugin>
				<groupId>org.apache.maven.plugins</groupId>
				<artifactId>maven-compiler-plugin</artifactId>
				<version>3.2</version>
				<configuration>
					<source>1.8</source>
					<target>1.8</target>
					<encoding>UTF-8</encoding>
				</configuration>
			</plugin>
		</plugins>
		<pluginManagement>
			<plugins>
				<!-- 配置 Tomcat 插件 -->
				<plugin>
					<groupId>org.apache.tomcat.maven</groupId>
					<artifactId>tomcat7-maven-plugin</artifactId>
					<version>2.2</version>
				</plugin>
			</plugins>
		</pluginManagement>
	</build>
</project>
```

3、 建立子项目

创建一个 war 工程 kang-usermanage，其 pom.xml 文件内容如下：

```
	<project                                        xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>
	<parent>
		<groupId>com.kang.parent</groupId>
		<artifactId>kang-parent</artifactId>
		<version>0.0.1-SNAPSHOT</version>
	</parent>
	<groupId>com.kang.usermanage</groupId>
	<artifactId>kang-usermanage</artifactId>
	<version>1.0.0-SNAPSHOT</version>
	<packaging>war</packaging>

	<dependencies>
		<!-- 单元测试 -->
		<dependency>
			<groupId>junit</groupId>
			<artifactId>junit</artifactId>
			<scope>test</scope>
		</dependency>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-webmvc</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-jdbc</artifactId>
		</dependency>
		<dependency>
			<groupId>org.springframework</groupId>
			<artifactId>spring-aspects</artifactId>
		</dependency>
		<!-- Mybatis -->
		<dependency>
			<groupId>org.mybatis</groupId>
			<artifactId>mybatis</artifactId>
		</dependency>
		<dependency>
			<groupId>org.mybatis</groupId>
			<artifactId>mybatis-spring</artifactId>
		</dependency>

		<dependency>
			<groupId>com.github.pagehelper</groupId>
```

```
    <artifactId>pagehelper</artifactId>
    <version>3.7.5</version>
</dependency>
<dependency>
    <groupId>com.github.jsqlparser</groupId>
    <artifactId>jsqlparser</artifactId>
    <version>0.9.1</version>
</dependency>

<!--  通用 Mapper -->
<dependency>
    <groupId>com.github.abel533</groupId>
    <artifactId>mapper</artifactId>
    <version>2.3.4</version>
</dependency>

<!-- MySql -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
</dependency>
<dependency>
    <groupId>org.slf4j</groupId>
    <artifactId>slf4j-log4j12</artifactId>
</dependency>
<!-- Jackson Json 处理工具包  -->
<dependency>
    <groupId>com.fasterxml.jackson.core</groupId>
    <artifactId>jackson-databind</artifactId>
</dependency>
<!--  连接池  -->
<dependency>
    <groupId>com.jolbox</groupId>
    <artifactId>bonecp-spring</artifactId>
</dependency>
<!-- JSP 相关  -->
<dependency>
    <groupId>jstl</groupId>
    <artifactId>jstl</artifactId>
</dependency>
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <scope>provided</scope>
```

```
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>jsp-api</artifactId>
            <scope>provided</scope>
        </dependency>
        <!-- Apache 工具组件 -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
        </dependency>
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-io</artifactId>
        </dependency>
    </dependencies>

    <build>
        <!-- 配置插件 -->
        <plugins>
            <plugin>
                <groupId>org.apache.tomcat.maven</groupId>
                <artifactId>tomcat7-maven-plugin</artifactId>
                <configuration>
                    <url>http://localhost:8081/manager/text</url>
                    <server>tomcat7</server>
                    <port>8081</port>
                    <path>/</path>
                    <username>tomcat</username>
                    <password>123456</password>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

三、配置文件详解

1、 mysql 的外部属性文件:jdbc.properties

```
jdbc.driverClassName=com.mysql.jdbc.Driver
jdbc.url=jdbc:mysql://127.0.0.1:3306/mybatis?useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries=true
jdbc.username=root
jdbc.password=root
```

2、 编写 Mybatis 的全局配置文件：mybatis/mybatis-config.xml

在这里集成了分页助手和通用 mapper。注意，分页助手的配置要放在通用 mapper

之前。

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE configuration
  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
  "http://mybatis.org/dtd/mybatis-3-config.dtd">
<configuration>

 <plugins>
      <!-- 配置分页助手 -->
      <plugin interceptor="com.github.pagehelper.PageHelper">
            <property name="dialect" value="mysql" />
            <!-- 设置为 true 时，使用 RowBounds 分页会进行 count 查询 -->
            <property name="rowBoundsWithCount" value="true" />
      </plugin>

      <!-- 通用 Mapper -->
    <plugin  interceptor="com.github.abel533.mapperhelper.MapperInterceptor">
            <!--主键自增回写方法,默认值 MYSQL,详细说明请看文档 -->
            <property name="IDENTITY" value="MYSQL" />
            <!--通用 Mapper 接口，多个通用接口用逗号隔开 -->
            <property                                   name="mappers"
value="com.github.abel533.mapper.Mapper" />
      </plugin>
 </plugins>

</configuration>
```

3、 编写 spring 整合 Mybatis 的 xml 文件。spring/applicationContext-mybatis.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
    xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
    http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd
    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
    http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util-4.0.xsd">
```

```
        <bean class="org.mybatis.spring.SqlSessionFactoryBean">
            <!-- 指定数据源 -->
            <property name="dataSource" ref="dataSource"/>
            <!-- 指定 mybatis 的全局配置文件 -->
            <property        name="configLocation"        value="classpath:mybatis/mybatis-
config.xml"/>
            <!-- 指定 mapper.xml 文件，扫描所有的文件 -->
            <property                                                name="mapperLocations"
value="classpath:mybatis/mappers/**/*.xml"/>
            <!-- 指定别名包 -->
            <property name="typeAliasesPackage" value="com.kang.mybatis.pojo"/>
        </bean>

        <!-- 定义 Mapper 接口的扫描器 -->
        <bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
            <property name="basePackage" value="com.kang.mybatis.mapper"/>
        </bean>

</beans>
```

4、 编写 spring 的事务管理配置文件 spring/applicationContext-transaction.xml

```
       <beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
        xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd
        http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
        http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util-4.0.xsd">

        <!-- 定义事务管理器 -->
        <bean id="transactionManager"

 class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
            <property name="dataSource" ref="dataSource" />
        </bean>
        <!-- 定义事务策略 -->
        <tx:advice id="txAdvice" transaction-manager="transactionManager">
            <tx:attributes>
                <!--所有以 query 开头的方法都是只读的 -->
                <tx:method name="query*" read-only="true" />
                <!--其他方法使用默认事务策略 -->
                <tx:method name="*" />
            </tx:attributes>
        </tx:advice>

        <aop:config>
            <!--pointcut 元素定义一个切入点，execution 中的第一个星号 用以匹配方
法的返回类型，
                这里星号表明匹配所有返回类型。 com.abc.dao.*.*(..)表明匹配
com.kang.mybatis.service 包下的所有类的所有
                方法 -->
            <aop:pointcut            id="myPointcut"            expression="execution(*
com.kang.mybatis.service.*.*(..))" />
            <!--将定义好的事务处理策略应用到上述的切入点 -->
            <aop:advisor advice-ref="txAdvice" pointcut-ref="myPointcut" />
        </aop:config>

</beans>
```

5、 编写 spring 关于 bean 的配置文件 spring/applicationContext.xml

```
       <beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:context="http://www.springframework.org/schema/context"
xmlns:p="http://www.springframework.org/schema/p"
        xmlns:aop="http://www.springframework.org/schema/aop"
xmlns:tx="http://www.springframework.org/schema/tx"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd
        http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.0.xsd
http://www.springframework.org/schema/tx
http://www.springframework.org/schema/tx/spring-tx-4.0.xsd
        http://www.springframework.org/schema/util
http://www.springframework.org/schema/util/spring-util-4.0.xsd">

        <!-- 使用 spring 自带的占位符替换功能 -->
        <bean
```

```
 class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">
            <!--  允许 JVM 参数覆盖  -->
            <property                                  name="systemPropertiesModeName"
value="SYSTEM_PROPERTIES_MODE_OVERRIDE" />
            <!--  忽略没有找到的资源文件  -->
            <property name="ignoreResourceNotFound" value="true" />
            <!--  配置资源文件  -->
            <property name="locations">
                  <list>
                        <value>classpath:jdbc.properties</value>
                  </list>
            </property>
      </bean>

      <!--  扫描包  -->
      <context:component-scan base-package="com.kang"/>

       <!--  定义数据源  -->
      <bean id="dataSource" class="com.jolbox.bonecp.BoneCPDataSource"
            destroy-method="close">
            <!--  数据库驱动  -->
            <property name="driverClass" value="${jdbc.driverClassName}" />
            <!--  相应驱动的 jdbcUrl -->
            <property name="jdbcUrl" value="${jdbc.url}" />
            <!--  数据库的用户名  -->
            <property name="username" value="${jdbc.username}" />
            <!--  数据库的密码  -->
            <property name="password" value="${jdbc.password}" />
            <!--  检查数据库连接池中空闲连接的间隔时间，单位是分，默认值：240，
如果要取消则设置为 0 -->
            <property name="idleConnectionTestPeriod" value="60" />
            <!--  连接池中未使用的链接最大存活时间，单位是分，默认值：60，如果
要永远存活设置为 0 -->
            <property name="idleMaxAge" value="30" />
            <!--  每个分区最大的连接数  -->
            <!--
                  判断依据：请求并发数
             -->
            <property name="maxConnectionsPerPartition" value="100" />
            <!--  每个分区最小的连接数  -->
            <property name="minConnectionsPerPartition" value="5" />
      </bean>

</beans>
```

6、 编写 springMVC 的 xml 配置文件 spring/kang-usermanage-servlet.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:p="http://www.springframework.org/schema/p"
 xmlns:context="http://www.springframework.org/schema/context"
 xmlns:mvc="http://www.springframework.org/schema/mvc"
 xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.0.xsd
            http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd
            http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-4.0.xsd">

 <!-- 定义注解驱动 -->
 <mvc:annotation-driven/>

 <!-- 定义 Controller 的扫描包 -->
 <context:component-scan base-package="com.kang.mybatis.controller"/>

 <!-- 定义试图解析器 -->
 <!--
     Example: prefix="/WEB-INF/jsp/", suffix=".jsp", viewname="test" -> "/WEB-
INF/jsp/test.jsp"
  -->
 <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
     <property name="prefix" value="/WEB-INF/views/"/>
     <property name="suffix" value=".jsp"/>
 </bean>

</beans>
```

7、 编写 web.xml 完成项目整体配置

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns="http://java.sun.com/xml/ns/javaee"
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
 id="WebApp_ID" version="2.5">
 <display-name>kang-usermanage</display-name>

 <!-- 加载 spring 的配置文件 -->
 <context-param>
     <param-name>contextConfigLocation</param-name>
```

```
            <param-value>classpath:spring/applicationContext*.xml</param-value>
      </context-param>

      <!--Spring 的 ApplicationContext  载入  -->
      <listener>
            <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
      </listener>

      <!--  编码过滤器，以 UTF8 编码，避免中文乱码  -->
      <filter>
            <filter-name>encodingFilter</filter-name>
            <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-
class>
            <init-param>
                  <param-name>encoding</param-name>
                  <param-value>UTF8</param-value>
            </init-param>
      </filter>
      <filter-mapping>
            <filter-name>encodingFilter</filter-name>
            <url-pattern>/*</url-pattern>
      </filter-mapping>

      <!--  添加 spring 对 REST 风格的支持  -->
      <filter>
            <filter-name>HttpMethodFilter</filter-name>
            <filter-
class>org.springframework.web.filter.HttpPutFormContentFilter</filter-class>
      </filter>
      <filter-mapping>
            <filter-name>HttpMethodFilter</filter-name>
            <url-pattern>/*</url-pattern>
      </filter-mapping>
      <!--  将 POST 请求转化为 DELETE 或者是 PUT  要用_method 指定真正的请求参
数  -->
      <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>


    <!-- 配置 SpringMVC 框架入口 -->
    <servlet>
        <servlet-name>kang-usermanage</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-
class>
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:spring/kang-usermanage-servlet.xml</param-
value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>


    <servlet-mapping>
        <servlet-name>kang-usermanage</servlet-name>
        <!-- 可用： *.xxx，/，/xxx/*．不可用：/* -->
        <url-pattern>/rest/*</url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
    </welcome-file-list>

</web-app>
```

8、 添加日志属性文件 log4j.properties

```
log4j.rootLogger=DEBUG,A1
log4j.logger.com.taotao = DEBUG
log4j.logger.org.mybatis = DEBUG

log4j.appender.A1=org.apache.log4j.ConsoleAppender
log4j.appender.A1.layout=org.apache.log4j.PatternLayout
log4j.appender.A1.layout.ConversionPattern=%-d{yyyy-MM-dd HH:mm:ss,SSS} [%t]
[%c]-[%p] %m%n
```

四、业务代码编写

1、 通用页面跳转

```
package com.kang.mybatis.controller;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

//通用的页面跳转 Controller
@RequestMapping("page")
@Controller
public class PageController {

    @RequestMapping(value = "{pageName}", method = RequestMethod.GET)
    public String toPage(@PathVariable("pageName") String pageName) {
        return pageName;
    }

 }
```

2、 封装 EasyUIResult

```
package com.kang.mybatis.bean;

import java.util.List;

public class EasyUIResult {

    private Long total;

    private List<?> rows;

    public EasyUIResult() {

    }

    public EasyUIResult(Long total, List<?> rows) {
        this.total = total;
        this.rows = rows;
    }

    public Long getTotal() {
        return total;
    }

    public void setTotal(Long total) {
        this.total = total;
    }

    public List<?> getRows() {
        return rows;
    }
```

```
		public void setRows(List<?> rows) {
			this.rows = rows;
		}

	}
```
3、 编写 mapper 接口，此接口继承通用 mapper
```
	package com.kang.mybatis.mapper;

	import com.github.abel533.mapper.Mapper;
	import com.kang.mybatis.pojo.User;

	public interface NewUserMapper extends Mapper<User>{

}
```
4、 添加 pojo 类
```
	package com.kang.mybatis.pojo;

	import java.util.Date;

	import javax.persistence.GeneratedValue;
	import javax.persistence.GenerationType;
	import javax.persistence.Id;
	import javax.persistence.Table;

	@Table(name = "tb_user")
	public class User {

		@Id
		@GeneratedValue(strategy = GenerationType.IDENTITY)
		private Long id;

		// 用户名
		private String userName;

		// 密码
		private String password;

		// 姓名
		private String name;

		// 年龄
		private Integer age;
```

```
// 性别，1 男性，2 女性
private Integer sex;

// 出生日期
private Date birthday;

// 创建时间
private Date created;

// 更新时间
private Date updated;

public Long getId() {
	return id;
}

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

public String getuserName() {
	return userName;
}

public void setuserName(String userName) {
	this.userName = userName;
}

public String getPassword() {
	return password;
}

public void setPassword(String password) {
	this.password = password;
}

public String getName() {
	return name;
}

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

```
	public Integer getAge() {
		return age;
	}

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

	public Integer getSex() {
		return sex;
	}

	public void setSex(Integer sex) {
		this.sex = sex;
	}

	public Date getBirthday() {
		return birthday;
	}

	public void setBirthday(Date birthday) {
		this.birthday = birthday;
	}

	public Date getCreated() {
		return created;
	}

	public void setCreated(Date created) {
		this.created = created;
	}

	public Date getUpdated() {
		return updated;
	}

	public void setUpdated(Date updated) {
		this.updated = updated;
	}

	@Override
	public String toString() {
		return "User [id=" + id + ", userName=" + userName + ", password=" +
password + ", name=" + name
```

```
                    + ", age=" + age + ", sex=" + sex + ", birthday=" + birthday + ",
created=" + created
                    + ", updated=" + updated + "]";
        }

}
```

5、 编写 service 方法

```
    package com.kang.mybatis.service;

    import java.util.Date;
    import java.util.List;

    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.stereotype.Service;

    import com.github.abel533.entity.Example;
    import com.github.pagehelper.PageHelper;
    import com.github.pagehelper.PageInfo;
    import com.kang.mybatis.bean.EasyUIResult;
    import com.kang.mybatis.mapper.NewUserMapper;
    import com.kang.mybatis.pojo.User;

    @Service
    public class NewUserService {

        @Autowired
        private NewUserMapper newUserMapper;

        public EasyUIResult queryUserList(Integer page, Integer rows) {
            // 设置分页参数
            PageHelper.startPage(page, rows);

            // 设置查询条件
            Example example = new Example(User.class);
            example.setOrderByClause("created DESC"); // 设置排序条件
            List<User> users = this.newUserMapper.selectByExample(example);

            PageInfo<User> pageInfo = new PageInfo<User>(users);
            return new EasyUIResult(pageInfo.getTotal(), pageInfo.getList());
        }

        public User queryUserById(Long id) {
            return this.newUserMapper.selectByPrimaryKey(id);
        }
        public void saveUser(User user) {
            user.setCreated(new Date());
            user.setUpdated(new Date());
            this.newUserMapper.insertSelective(user);
        }

        public void updateUser(User user) {
            this.newUserMapper.updateByPrimaryKeySelective(user);
        }

        public void deleteUserById(Long id) {
            this.newUserMapper.deleteByPrimaryKey(id);
        }

}
```

6、 编写 controller 方法

```
package com.kang.mybatis.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.ResponseBody;

import com.kang.mybatis.bean.EasyUIResult;
import com.kang.mybatis.pojo.User;
import com.kang.mybatis.service.NewUserService;

@RequestMapping("user")
@Controller
public class UserController {

    // @Autowired
    // private UserService userService;

    @Autowired
    private NewUserService userService;

    @RequestMapping(value = "list", method = RequestMethod.GET)
```

```
	@ResponseBody
	public EasyUIResult queryUserList(@RequestParam(value = "page",
defaultValue = "1") Integer page,
			@RequestParam(value = "rows", defaultValue = "5") Integer rows) {
		return this.userService.queryUserList(page, rows);
	}

	//查询数据
	@RequestMapping(value="{userId}",method=RequestMethod.GET)
	public ResponseEntity<User> quseryUserById(@PathVariable("userId") Long
userId){
		try {
			User user=this.userService.queryUserById(userId);
			if(null==user){
				//返回 404 资源不存在
				return ResponseEntity.status(HttpStatus.NOT_FOUND).body(null);
			}
			//返回 200，成功
			//return ResponseEntity.status(HttpStatus.OK).body(user);
			return ResponseEntity.ok(user);
		} catch (Exception e) {
			e.printStackTrace();
		}
		//查询出错，返回 500
		return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(null);
	}

	//新增数据
	@RequestMapping(method=RequestMethod.POST)
	public ResponseEntity<Void> saveUser(User user){
		try {
			this.userService.saveUser(user);;
			return ResponseEntity.status(HttpStatus.CREATED).build();
		} catch (Exception e) {
			e.printStackTrace();
		}
		//查询出错，返回 500
		return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(null);
	}

	//更新数据
	@RequestMapping(method=RequestMethod.PUT)
```

```
        public ResponseEntity<Void> updateUser(User user){
         try {
                this.userService.updateUser(user);
                return ResponseEntity.status(HttpStatus.NO_CONTENT).build();
         } catch (Exception e) {
                e.printStackTrace();
         }
         //查询出错，返回 500
         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(null);
        }

        //删除数据
        @RequestMapping(method=RequestMethod.DELETE)
        public                                          ResponseEntity<Void>
deleteUser(@RequestParam(value="id",defaultValue="0") Long id){
         try {
                if(id.intValue()==0){
                       //参数有误
                       return ResponseEntity.status(HttpStatus.BAD_REQUEST).build();
                }
                this.userService.deleteUserById(id);
                return ResponseEntity.status(HttpStatus.NO_CONTENT).build();
         } catch (Exception e) {
                e.printStackTrace();
         }
         //查询出错，返回 500
         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(null);
        }

}
```

7、 编写对通用 mapper 的单元测试文件

```
    package com.kang.mabatis.test;



    import static org.junit.Assert.fail;

    import java.util.ArrayList;
    import java.util.Date;
    import java.util.List;

    import org.junit.Before;
    import org.junit.Test;
```

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.github.abel533.entity.Example;
import com.kang.mybatis.mapper.NewUserMapper;
import com.kang.mybatis.pojo.User;
public class NewUserMapperTest {

	private NewUserMapper newUserMapper;

	@Before
	public void setUp() throws Exception {
		ApplicationContext applicationContext = new
ClassPathXmlApplicationContext(
				"classpath:spring/applicationContext*.xml");
		this.newUserMapper =
applicationContext.getBean(NewUserMapper.class);
	}

	@Test
	public void testSelectOne() {
		User record = new User();
		// 设置查询条件
		record.setuserName("zhangsan");
		record.setPassword("123456");
		User user = this.newUserMapper.selectOne(record);
		System.out.println(user);
	}

	@Test
	public void testSelect() {
		User record = new User();
		// 设置查询条件
		record.setuserName("zhangsan");
		List<User> list = this.newUserMapper.select(record);
		for (User user : list) {
			System.out.println(user);
		}
	}

	@Test
	public void testSelectCount() {
		System.out.println(this.newUserMapper.selectCount(null));
	}
```

```
        @Test
        public void testSelectByPrimaryKey() {
         //注意 selectByPrimaryKey 的输入参数必须是 Long 型，不能是 string 类型。
new Long(1)
            User user = this.newUserMapper.selectByPrimaryKey(new Long(1));
            System.out.println(user);
        }

        @Test
        public void testInsert() {
            User record = new User();
            //  设置查询条件
            record.setuserName("test_username_3");
            //record.setAge(20);
            //record.setBirthday(new Date());
            record.setCreated(new Date());
            //record.setName("test_name_1");
            //record.setPassword("123456");
            record.setSex(1);
            record.setUpdated(new Date());
            //使用所有的字段作为插入语句的字段
            int count = this.newUserMapper.insert(record);
            System.out.println(count);
            System.out.println(record.getId());
        }

        @Test
        public void testInsertSelective() {
            User record = new User();
            //  设置查询条件
            record.setuserName("test_username_2");
            //record.setAge(20);
           // record.setBirthday(new Date());
            record.setCreated(new Date());
           // record.setName("test_name_1");
           // record.setPassword("123456");
            record.setSex(1);
            record.setUpdated(new Date());
            //将不为 null 的字段作为插入语句的字段
            int count = this.newUserMapper.insertSelective(record);
            System.out.println(count);
            System.out.println(record.getId());
        }
```

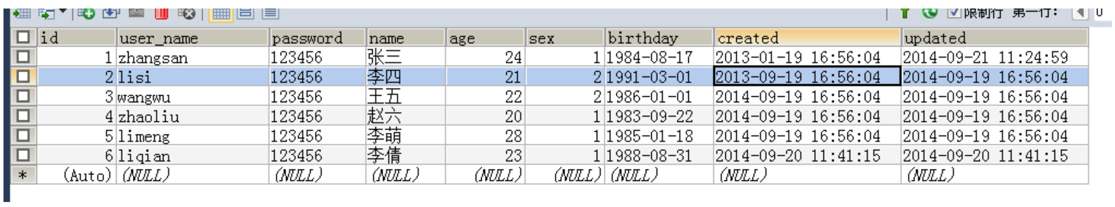
```
	@Test
	public void testDelete() {
//		this.newUserMapper.delete(null);
	}

	@Test
	public void testDeleteByPrimaryKey() {
		System.out.println(this.newUserMapper.deleteByPrimaryKey(9L));
	}

	@Test
	public void testUpdateByPrimaryKey() {
		fail("Not yet implemented");
	}

	@Test
	public void testUpdateByPrimaryKeySelective() {
		User record = new User();
		record.setId(1L);
		record.setAge(24);
		this.newUserMapper.updateByPrimaryKeySelective(record);
	}

	@Test
	public void testSelectCountByExample() {
		fail("Not yet implemented");
	}

	@Test
	public void testDeleteByExample() {
		fail("Not yet implemented");
	}

	@Test
	public void testSelectByExample() {
		Example example = new Example(User.class);
		List<Object> values = new ArrayList<Object>();
		values.add(1L);
		values.add(2L);
		values.add(3L);
		//批量查询
		//example.createCriteria().andIn("id", values);
		//单个查询
```

```
                example.createCriteria().andEqualTo("id", 1L);
                List<User> list = this.newUserMapper.selectByExample(example);

                for (User user : list) {
                        System.out.println(user);
                }
        }

        @Test
        public void testUpdateByExampleSelective() {
                fail("Not yet implemented");
        }

        @Test
        public void testUpdateByExample() {
                fail("Not yet implemented");
        }

}
```

五、项目测试

1、 查询所有信息

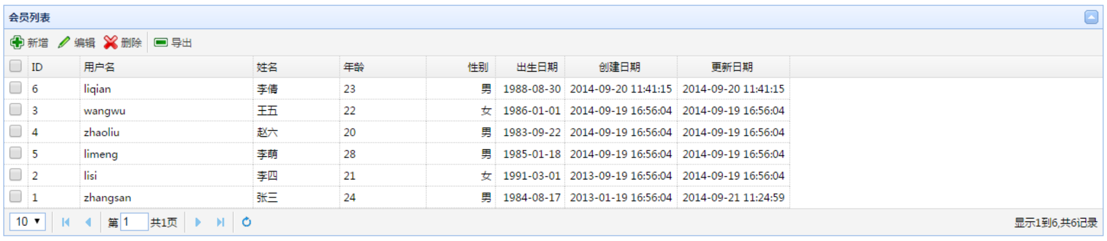
数据库内容如下：



id	user_name	password	name	age	sex	birthday	created	updated
1	zhangsan	123456	张三	24	1	1984-08-17	2013-01-19 16:56:04	2014-09-21 11:24:59
2	lisi	123456	李四	21	2	1991-03-01	2013-09-19 16:56:04	2014-09-19 16:56:04
3	wangwu	123456	王五	22	2	1986-01-01	2014-09-19 16:56:04	2014-09-19 16:56:04
4	zhaoliu	123456	赵六	20	1	1983-09-22	2014-09-19 16:56:04	2014-09-19 16:56:04
5	limeng	123456	李萌	28	1	1985-01-18	2014-09-19 16:56:04	2014-09-19 16:56:04
6	liqian	123456	李倩	23	1	1988-08-31	2014-09-20 11:41:15	2014-09-20 11:41:15
(Auto)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

在浏览器中输入: http://127.0.0.1:8081/rest/page/users

得到页面如下：前端页面基于 easyUI 实现，数据按照更新时间排序。

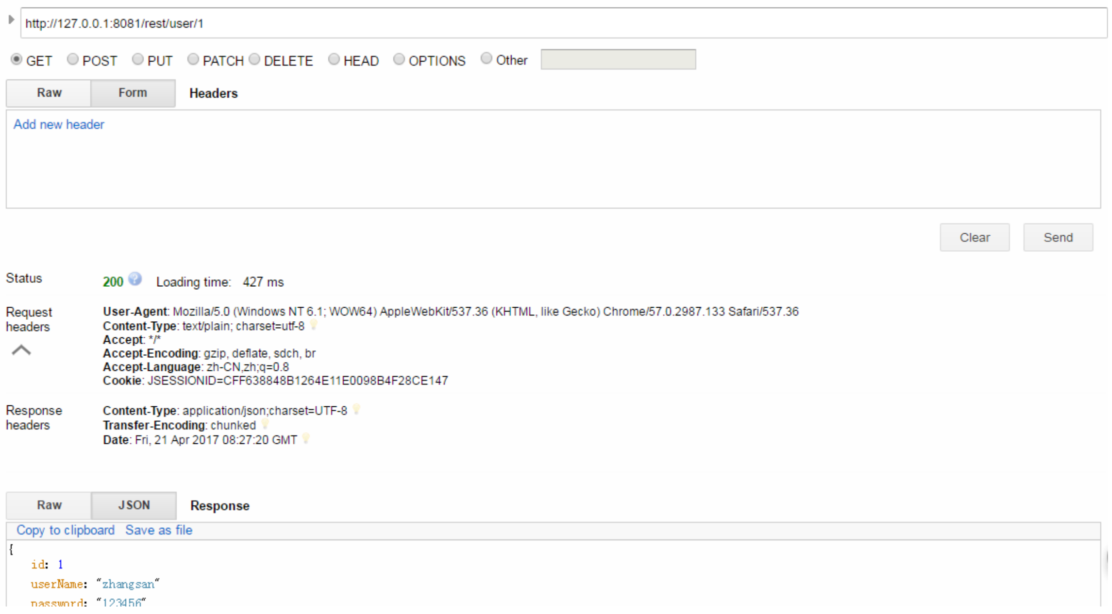


ID	用户名	姓名	年龄	性别	出生日期	创建日期	更新日期
6	liqian	李倩	23	男	1988-08-30	2014-09-20 11:41:15	2014-09-20 11:41:15
3	wangwu	王五	22	女	1986-01-01	2014-09-19 16:56:04	2014-09-19 16:56:04
4	zhaoliu	赵六	20	男	1983-09-22	2014-09-19 16:56:04	2014-09-19 16:56:04
5	limeng	李萌	28	男	1985-01-18	2014-09-19 16:56:04	2014-09-19 16:56:04
2	lisi	李四	21	女	1991-03-01	2013-09-19 16:56:04	2014-09-19 16:56:04
1	zhangsan	张三	24	男	1984-08-17	2013-01-19 16:56:04	2014-09-21 11:24:59

10 第 1 共1页 显示1到6,共6记录

2、 restful 的相关测试

(1)使用 get 获取数据



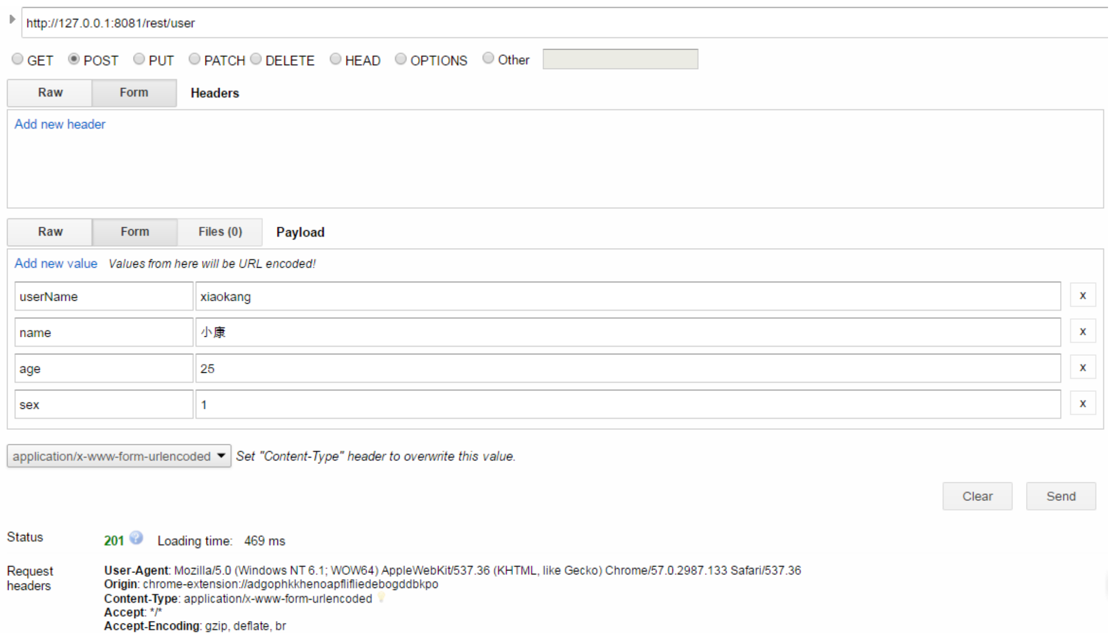
```
http://127.0.0.1:8081/rest/user/1
```

[✓ GET] POST PUT PATCH DELETE HEAD OPTIONS Other

```
Status            200   Loading time:   427 ms

Request           User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36
headers           Content-Type: text/plain; charset=utf-8
                  Accept: */*
                  Accept-Encoding: gzip, deflate, sdch, br
                  Accept-Language: zh-CN,zh;q=0.8
                  Cookie: JSESSIONID=CFF638848B1264E11E0098B4F28CE147

Response          Content-Type: application/json;charset=UTF-8
headers           Transfer-Encoding: chunked
                  Date: Fri, 21 Apr 2017 08:27:20 GMT
```

```
{
  id: 1
  userName: "zhangsan"
  password: "123456"
```

(2)使用 post 提交数据



```
http://127.0.0.1:8081/rest/user
```

GET [✓ POST] PUT PATCH DELETE HEAD OPTIONS Other

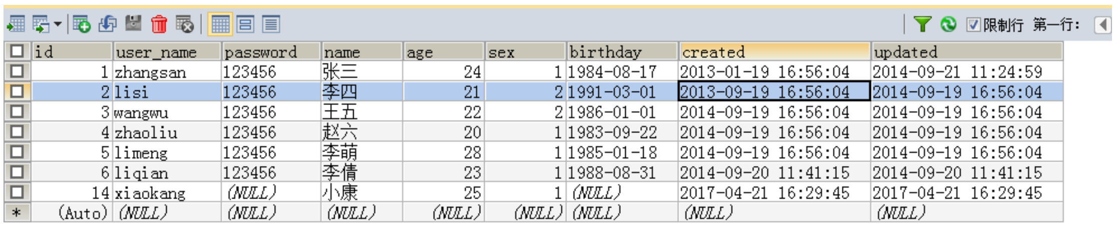
Payload — Add new value *Values from here will be URL encoded!*

userName	xiaokang
name	小康
age	25
sex	1

application/x-www-form-urlencoded *Set "Content-Type" header to overwrite this value.*

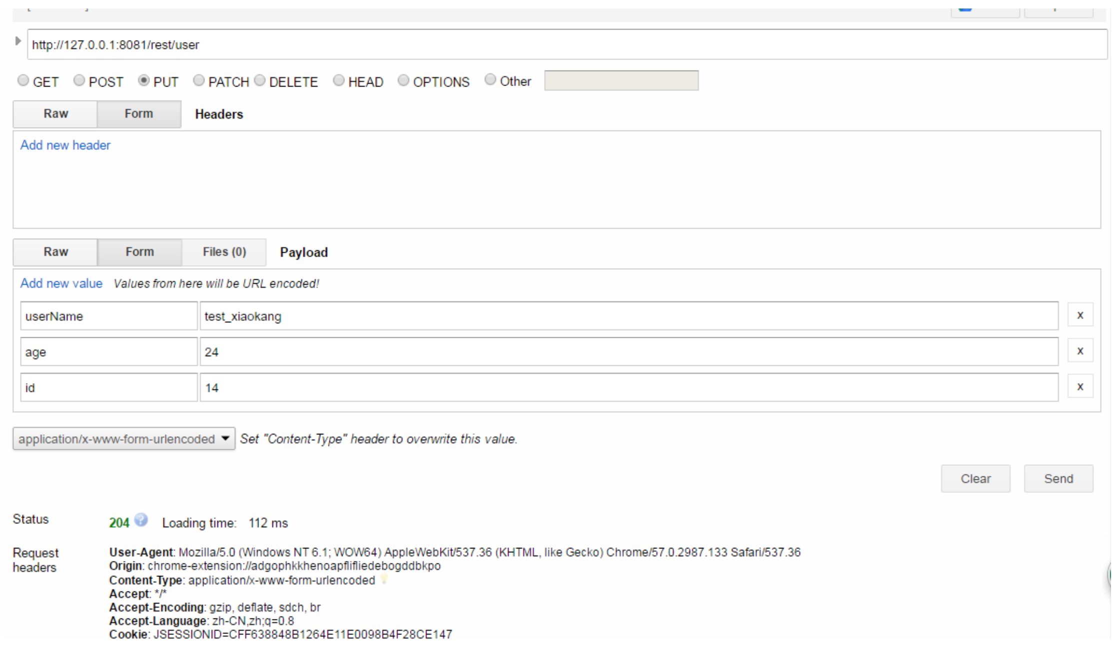
```
Status            201   Loading time:   469 ms

Request           User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36
headers           Origin: chrome-extension://adgophkkhenoapfliflie debogddbkpo
                  Content-Type: application/x-www-form-urlencoded
                  Accept: */*
                  Accept-Encoding: gzip, deflate, br
```

数据库数据：



id	user_name	password	name	age	sex	birthday	created	updated
1	zhangsan	123456	张三	24	1	1984-08-17	2013-01-19 16:56:04	2014-09-21 11:24:59
2	lisi	123456	李四	21	2	1991-03-01	2013-09-19 16:56:04	2014-09-19 16:56:04
3	wangwu	123456	王五	22	2	1986-01-01	2014-09-19 16:56:04	2014-09-19 16:56:04
4	zhaoliu	123456	赵六	20	1	1983-09-22	2014-09-19 16:56:04	2014-09-19 16:56:04
5	limeng	123456	李萌	28	1	1985-01-18	2014-09-19 16:56:04	2014-09-19 16:56:04
6	liqian	123456	李倩	23	1	1988-08-31	2014-09-20 11:41:15	2014-09-20 11:41:15
14	xiaokang	(NULL)	小康	25	1	(NULL)	2017-04-21 16:29:45	2017-04-21 16:29:45
(Auto)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

(3)使用 put 更新数据



http://127.0.0.1:8081/rest/user

GET POST [✓ PUT] PATCH DELETE HEAD OPTIONS Other

Raw Form **Headers**

Add new header

Raw Form Files (0) **Payload**

Add new value *Values from here will be URL encoded!*

userName	test_xiaokang
age	24
id	14

application/x-www-form-urlencoded *Set "Content-Type" header to overwrite this value.*

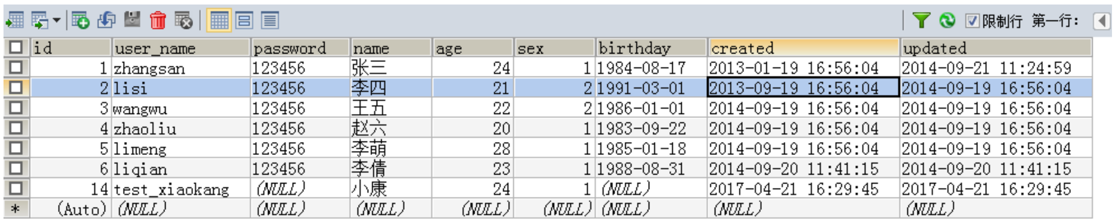
Clear Send

Status 204 Loading time: 112 ms

Request headers

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36
Origin: chrome-extension://adgophkkhenoapflifliedebogddbkpo
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=CFF638848B1264E11E0098B4F28CE147

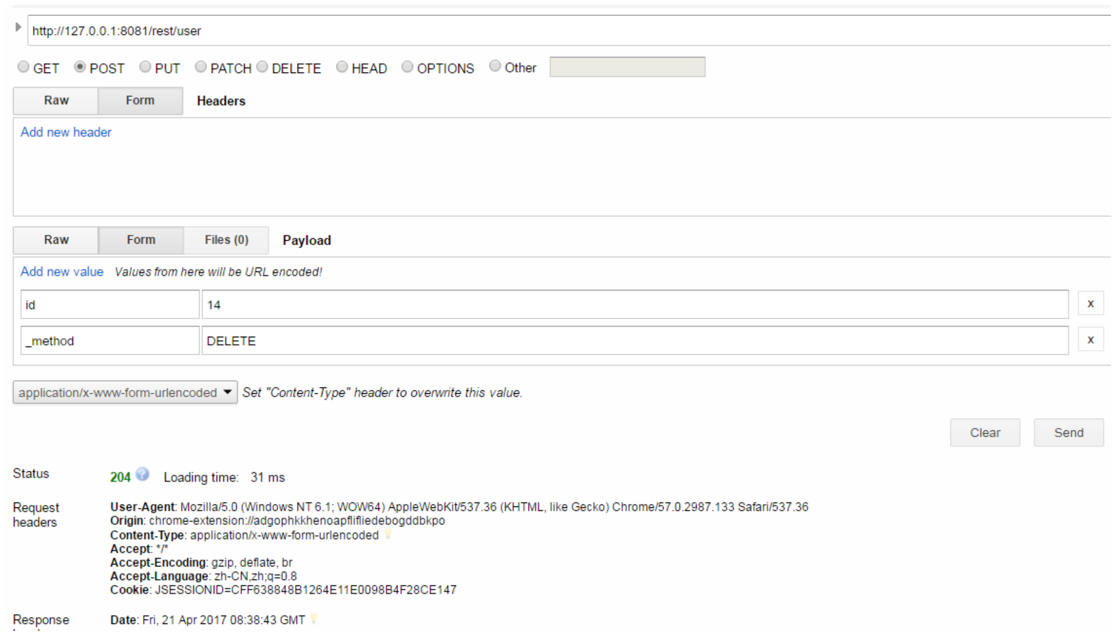
数据库数据：



限制行 第一行：

id	user_name	password	name	age	sex	birthday	created	updated
1	zhangsan	123456	张三	24	1	1984-08-17	2013-01-19 16:56:04	2014-09-21 11:24:59
2	lisi	123456	李四	21	2	1991-03-01	2013-09-19 16:56:04	2014-09-19 16:56:04
3	wangwu	123456	王五	22	2	1986-01-01	2014-09-19 16:56:04	2014-09-19 16:56:04
4	zhaoliu	123456	赵六	20	1	1983-09-22	2014-09-19 16:56:04	2014-09-19 16:56:04
5	limeng	123456	李萌	28	1	1985-01-18	2014-09-19 16:56:04	2014-09-19 16:56:04
6	liqian	123456	李倩	23	1	1988-08-31	2014-09-20 11:41:15	2014-09-20 11:41:15
14	test_xiaokang	(NULL)	小康	24	1	(NULL)	2017-04-21 16:29:45	2017-04-21 16:29:45
(Auto)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)

（4）使用 delete 删除数据



http://127.0.0.1:8081/rest/user

GET [✓ POST] PUT PATCH DELETE HEAD OPTIONS Other

Raw Form **Headers**

Add new header

Raw Form Files (0) **Payload**

Add new value *Values from here will be URL encoded!*

id	14
_method	DELETE

application/x-www-form-urlencoded *Set "Content-Type" header to overwrite this value.*

Clear Send

Status 204 Loading time: 31 ms

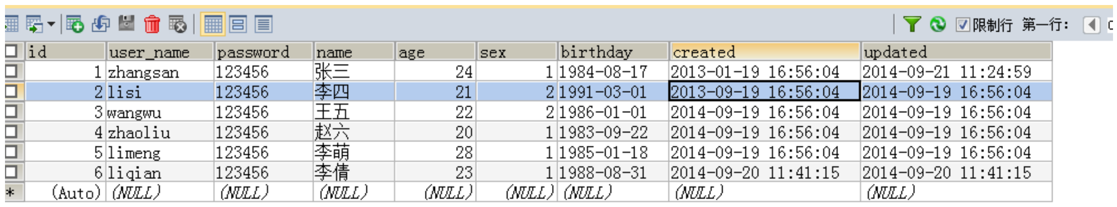
Request headers

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/57.0.2987.133 Safari/537.36
Origin: chrome-extension://adgophkkhenoapflifliedebogddbkpo
Content-Type: application/x-www-form-urlencoded
Accept: */*
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=CFF638848B1264E11E0098B4F28CE147

Response

Date: Fri, 21 Apr 2017 08:38:43 GMT

数据库数据：



	id	user_name	password	name	age	sex	birthday	created	updated
☐	1	zhangsan	123456	张三	24	1	1984-08-17	2013-01-19 16:56:04	2014-09-21 11:24:59
☐	2	lisi	123456	李四	21	2	1991-03-01	2013-09-19 16:56:04	2014-09-19 16:56:04
☐	3	wangwu	123456	王五	22	2	1986-01-01	2014-09-19 16:56:04	2014-09-19 16:56:04
☐	4	zhaoliu	123456	赵六	20	1	1983-09-22	2014-09-19 16:56:04	2014-09-19 16:56:04
☐	5	limeng	123456	李萌	28	1	1985-01-18	2014-09-19 16:56:04	2014-09-19 16:56:04
☐	6	liqian	123456	李倩	23	1	1988-08-31	2014-09-20 11:41:15	2014-09-20 11:41:15
*	(Auto)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)	(NULL)