AntiSamy

XSS介绍

XSS：跨站脚本攻击(Cross Site Scripting)，为不和 CSS混淆，故将跨站脚本攻击缩写为XSS。XSS是指恶意攻击者往Web页面里插入恶意Script代码，当用户浏览该页时，嵌入其中Web里面的Script代码会被执行，从而达到恶意攻击用户的目的。有点类似于sql注入。

XSS攻击原理：

HTML是一种超文本标记语言，通过将一些字符特殊地对待来区别文本和标记，例如，小于符号（<）被看作是HTML标签的开始，之间的字符是页面的标题等等。当动态页面中插入的内容含有这些特殊字符时，用户浏览器会将其误认为是插入了HTML标签，当这些HTML标签引入了一段JavaScript脚本时，这些脚本程序就将会在用户浏览器中执行。所以，当这些特殊字符不能被动态页面检查或检查出现失误时，就将会产生XSS漏洞。

AntiSamy介绍

AntiSamy是OWASP的一个开源项目，通过对用户输入的 HTML / CSS / JavaScript 等内容进行检验和清理，确保输入符合应用规范。AntiSamy被广泛应用于Web服务对存储型和反射型XSS的防御中。

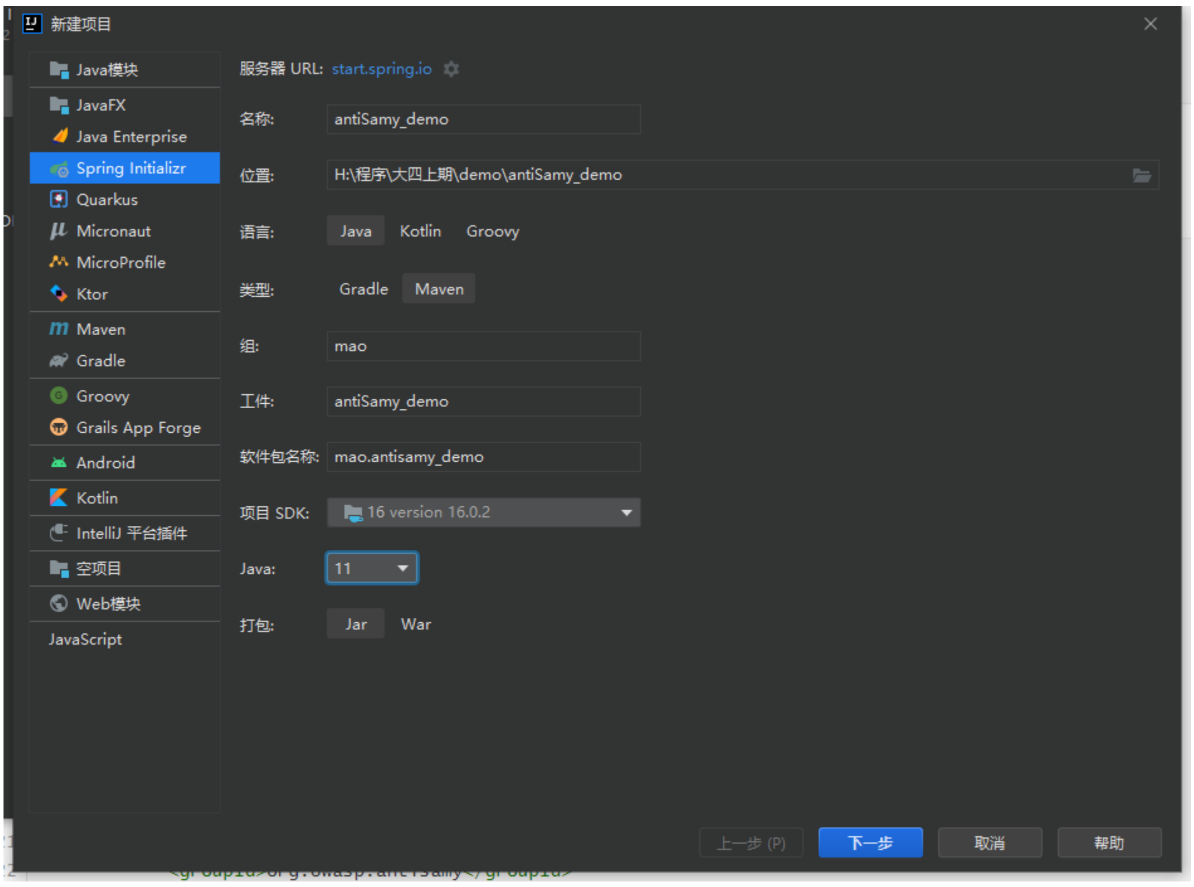
AntiSamy的maven坐标：

```xml
<dependency>
  <groupId>org.owasp.antisamy</groupId>
  <artifactId>antisamy</artifactId>
  <version>1.5.7</version>
</dependency>
```

AntiSamy入门案例

演示xss攻击

第一步：创建工程antiSamy_demo



服务器 URL: start.spring.io
名称: antiSamy_demo
位置: H:\程序\大四上期\demo\antiSamy_demo
语言: [✓ Java] Kotlin Groovy
类型: Gradle [✓ Maven]
组: mao
工件: antiSamy_demo
软件包名称: mao.antisamy_demo
项目 SDK: 16 version 16.0.2
Java: 11
打包: [✓ Jar] War

第二步：编写实体类Student

```java
package mao.antisamy_demo.entity;

/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.entity
 * Class(类名): Student
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/29
 * Time(创建时间)： 20:01
 * Version(版本): 1.0
 * Description(描述)： 无
 */


public class Student
{
    /**
     * id
     */
    private long id;
    /**
     * 名字
     */
    private String name;
    /**
     * 性别
     */
    private String sex;
    /**
     * 年龄
     */
    private int age;

    /**
     * Instantiates a new Student.
     */
    public Student()
    {

    }

    /**
     * Instantiates a new Student.
     *
     * @param id   the id
     * @param name the name
     * @param sex  the sex
     * @param age  the age
     */
    public Student(long id, String name, String sex, int age)
    {
        this.id = id;
        this.name = name;
        this.sex = sex;
        this.age = age;
    }

    /**
     * Gets id.
     *
     * @return the id
     */
    public long getId()
    {
        return id;
    }

    /**
     * Sets id.
     *
     * @param id the id
     */
    public void setId(long id)
    {
        this.id = id;
    }

    /**
     * Gets name.
     *
     * @return the name
     */
    public String getName()
    {
        return name;
    }

    /**
     * Sets name.
     *
     * @param name the name
     */
    public void setName(String name)
    {
        this.name = name;
    }

    /**
     * Gets sex.
     *
     * @return the sex
     */
    public String getSex()
    {
        return sex;
    }

    /**
     * Sets sex.
     *
     * @param sex the sex
     */
    public void setSex(String sex)
    {
        this.sex = sex;
    }

    /**
     * Gets age.
     *
     * @return the age
     */
    public int getAge()
    {
        return age;
    }

    /**
     * Sets age.
     *
     * @param age the age
     */
    public void setAge(int age)
    {
        this.age = age;
    }

    @Override
    @SuppressWarnings("all")
    public String toString()
    {
        final StringBuilder stringbuilder = new StringBuilder();
        stringbuilder.append("id: ").append(id).append('\n');
        stringbuilder.append("name: ").append(name).append('\n');
        stringbuilder.append("sex: ").append(sex).append('\n');
        stringbuilder.append("age: ").append(age).append('\n');
        return stringbuilder.toString();
    }
}
```

第三步：编写StudentController

```java
package mao.antisamy_demo.controller;

import mao.antisamy_demo.entity.Student;
import org.apache.juli.logging.Log;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.controller
 * Class(类名): StudentController
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/29
 * Time(创建时间)： 20:05
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@RestController
@RequestMapping("/student")
public class StudentController
{
    private static final List<Student> list =
Collections.synchronizedList(new ArrayList<>());

    private static final Logger log =
LoggerFactory.getLogger(StudentController.class);

    @PostMapping("/init")
    public synchronized void init()
    {
        Student student1 = new Student(10001, "张三", "男", 18);
        Student student2 = new Student(10002, "李四", "女", 16);
        Student student3 = new Student(10003, "王五", "男", 20);
        list.clear();
        list.add(student1);
        list.add(student2);
        list.add(student3);
        log.info("初始化完成");
    }

    @PostMapping
    public boolean save(@RequestBody Student student)
    {
        list.add(student);
        log.info("添加成功：\n" + student);
        return true;
    }

    @GetMapping
    public List<Student> getAll()
    {
        log.info("查询所有：\n" + list);
        return list;
    }

}
```

第四步：编写样式表form.css

```css
/*
  Project name(项目名称)：antiSamy_demo
  File name(文件名): form
  Author(作者）: mao
  Author QQ：1296193245
  GitHub：https://github.com/maomao124/
  Date(创建日期)：  2022/10/29
  Time(创建时间)：  20:54
  Version(版本): 1.0
  Description(描述)：  无
 */


/*表单位置*/
div.form_position {
    position: absolute;
    top: 50%;
    left: 50%;
    transform: translate(-50%, -50%);
}

/*表单的边框*/
div.form {
    border: 10px skyblue dotted;
}


/*表*/
table {
    width: 600px;
    border-collapse: collapse;
    color: #4edaff;
    background-color: #ffe4da;
    transition: all 1s linear 0s;
}

table:hover {
    background-color: #b4ffab;
    /*width: 800px;*/
    /*transition: all 1s linear 0s;*/
}


/*设置字体*/
td, input, input.input {
    font-size: 30px;
}
```

```css

input {
    /*font-size: 24px;*/
    /*width: 90%;*/
    color: coral;
}

input.input {
    width: 85%;
    color: coral;
    transition: all 0.5s linear 0s;
}

input.input:hover {
    width: 98%;
    transition: all 0.5s linear 0s;
    background-color: #fcffee;
}


/*设置提示*/
.prompt {
    text-align: center;
    width: 200px;
    transition: all 1s linear 0.2s;
}

.prompt:hover {
    transition: all 1s linear 0.2s;
    color: #ff2e2f;
}

/*提交按钮*/
input.submit {
    color: #6739ff;
}

input.submit:hover {

}

/*最上面的字*/
div.text {
    border: 10px violet dotted;
    text-align: center;
    font-size: 32px;
    color: tomato;
    background: bisque;
}

/*最上面的字的位置*/
div.text_position {
    position: absolute;
    top: 2%;
    left: 50%;
    transform: translate(-50%, 0%);
}
```

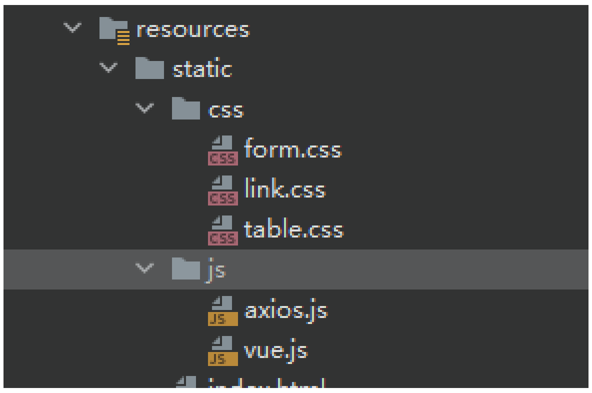
第五步：编写样式表link.css

```css
/*
  Project name(项目名称): antiSamy_demo
  File name(文件名): link
  Author(作者）: mao
  Author QQ: 1296193245
  GitHub: https://github.com/maomao124/
  Date(创建日期):  2022/10/29
  Time(创建时间):  20:17
  Version(版本): 1.0
  Description(描述):  无
 */


a {
    color: tomato;
    text-decoration: none;
    display: flex;
    justify-content: center;
    align-items: center;

    padding: 0;
    list-style-type: none;
    font-size: 22px;
    width: 10em;
    height: 2em;
    text-align: center;
    line-height: 2em;
    font-family: sans-serif;
    text-transform: capitalize;
    position: relative;
    margin: 0.8em;
}

/* 添加了左右两个圆点 */
a::before, a::after {
    content: '';
    position: absolute;
    width: 0.6em;
    height: 0.6em;
    background-color: gainsboro;
    top: calc(50% - 0.3em);
    border-radius: 50%;
    transition: 0.5s cubic-bezier(0.5, -0.5, 0.25, 1.5);
}

a::before {
    left: 0;
    z-index: -1;
}

a::after {
    right: 0;
    z-index: -2;
}

/* 添加悬浮效果 */
a:hover {
    color: deeppink;
}

/* 给前后伪元素添加悬浮效果，注意先后顺序，先是hover后是伪元素 */
a:hover::before, a:hover::after {
    width: 100%;
    height: 100%;
    border-radius: 0;
    background-color: dodgerblue;
}

a:hover::before {
    top: 0;
    left: -0.2em;
}

a:hover::after {
    right: -0.2em;
    filter: brightness(0.8);
}
```

第六步：编写样式表table.css

```css
/*
  Project name(项目名称): antiSamy_demo
  File name(文件名): table
  Author(作者）: mao
  Author QQ：1296193245
  GitHub：https://github.com/maomao124/
  Date(创建日期)：  2022/10/29
  Time(创建时间)：  20:33
  Version(版本): 1.0
  Description(描述)：  无
 */


table {
    width: 80%;
    background: #ccc;
    margin: 10px auto;
    /*border-collapse CSS 属性设置<table>内的单元格是否具有共享或单独的边框。*/
    /*collapse - 相邻单元格具有共享边框（折叠边框表格渲染模型）。*/
    border-collapse: collapse;
}

th, td {
    height: 25px;
    line-height: 25px;
    text-align: center;
    border: 1px solid #ccc;
    transition: all 0.4s linear 0s;
}

th {
    background: #eee;
    font-weight: normal;
}

tr {
    background: #fff;
    transition: all 2s linear 0s;
}

tr:hover {
    background: aqua;
    transition: all 0.4s linear 0s;
}

td:hover {
    color: magenta;
    transition: all 0.4s linear 0s;
}

td a {
    color: #06f;
    text-decoration: none;
}

td a:hover {
    color: #06f;
    text-decoration: underline;
    /*transition: all 1s linear 0s;*/
}

caption {
    color: #28ffc7;
    font-size: 28px;
}
```

第七步：添加vue和axios库



cdn：

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></script>
```

```
<script src="https://cdn.bootcss.com/vue/2.5.2/vue.min.js"></script>
```

第八步：编写index.html

```
<!DOCTYPE html>

<!--
Project name(项目名称)：antiSamy_demo
  File name(文件名): index
  Authors(作者）: mao
  Author QQ：1296193245
  GitHub：https://github.com/maomao124/
  Date(创建日期)：  2022/10/29
  Time(创建时间)：  20:03
  Description(描述)：  无
-->

<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>索引</title>
    <link rel="stylesheet" href="/css/link.css">
    <script src="/js/axios.js"></script>

```

```html

    <style>
        body {
            background-color: skyblue;
        }

        div.a {
            position: absolute;
            top: 50%;
            left: 50%;
            transform: translate(-50%, -50%);
        }
    </style>
</head>
<body>

<div class="a">

    <a href="/show.html">显示所有学生信息</a>
    <a href="/save.html">添加学生信息</a>
    <a href="" onclick="init()">初始化数据</a>

</div>

<script>

    function init()
    {
        //axios发起ajax请求
        axios({
            //请求的方式:
            method: "post",
            //请求的url:
            url: "/student/init",
            //url参数:
            params:
                {},
            //头信息:
            headers:
                {},
            //请求体参数:
            data:
                {},
        }).then(response =>
        {
            console.log(response);
            alert("初始化数据成功");
        }).catch(error =>
        {
            //console.log(error);
            alert("网络异常！");
        })
    }

</script>

</body>
</html>
```

第九步：编写show.html

```html
<!DOCTYPE html>

<!--
Project name(项目名称)：antiSamy_demo
  File name(文件名): show
  Authors(作者）: mao
  Author QQ：1296193245
  GitHub：https://github.com/maomao124/
  Date(创建日期)：  2022/10/29
  Time(创建时间)：  20:23
  Description(描述)：  无
-->

<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="/css/table.css">
    <script src="/js/axios.js"></script>
    <script src="/js/vue.js"></script>
</head>
<body>

<div id="app">

    <table>
        <tr>
            <th>学号</th>
            <th>姓名</th>
            <th>性别</th>
            <th>年龄</th>
        </tr>
        <tr v-for="student in studentList">
            <td v-html="student.id"></td>
            <td v-html="student.name"></td>
            <td v-html="student.sex"></td>
            <td v-html="student.age"></td>
        </tr>
    </table>

</div>

</body>

<script>

    var app = new Vue({
        el: "#app",
        data: {
            studentList: null
        },
        method: {},
        mounted: function ()
        {
            //axios发起ajax请求
            axios({
                //请求的方式:
                method: "get",
                //请求的url:
                url: "/student",
                //url参数:
                params:
                    {},
                //头信息:
                headers:
                    {},
                //请求体参数:
                data:
                    {},
            }).then(response =>
            {
                console.log(response);
                this.studentList = response.data;

            }).catch(error =>
            {
                //console.log(error);
                alert("网络异常！");
            })
        }
    })

</script>
</html>
```

第十步：编写save.html

```
<!DOCTYPE html>

<!--
Project name(项目名称)：antiSamy_demo
  File name(文件名): save
  Authors(作者）: mao
  Author QQ：1296193245
  GitHub：https://github.com/maomao124/
  Date(创建日期)：  2022/10/29
  Time(创建时间)：  20:23
  Description(描述)：  无
-->

```

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="css/form.css">
    <script src="js/axios.js"></script>
    <script src="js/vue.js"></script>
    <style>
        body {
            background-color: skyblue;
        }
    </style>
</head>
<body>

<div id="app">


    <div class="form_position">
        <div class="animated bounceInDown">
            <div class="form">
                <form action="" method="post">
                    <table border="1">
                        <tr>
                            <td colspan="2" align="center">
                            </td>
                        </tr>
                        <tr>
                            <td class="prompt">学生学号</td>
                            <td>
                                <label>
                                    <input v-model="student.id" class="input" type="number" name="id">
                                </label>
                            </td>
                        </tr>
                        <tr>
                            <td class="prompt">学生姓名</td>
                            <td>
                                <label>
                                    <input v-model="student.name" class="input" type="text" name="name">
                                </label>
                            </td>
                        </tr>
                        <tr>
                            <td class="prompt">学生性别</td>
                            <td>
                                <label>
                                    <input v-model="student.sex" class="input" type="text" name="sex">
                                </label>
                            </td>
                        </tr>
                        <tr>
                            <td class="prompt">学生年龄</td>
                            <td>
                                <label>
```

```
                                    <input v-model="student.age" class="input" type="number" name="age">
                                </label>
                            </td>
                        </tr>
                        <tr>
                            <td colspan="2" align="center">
                                <input @click="f()" class="submit" type="button" value="提交"/>
                            </td>
                        </tr>
                    </table>
                </form>
            </div>
        </div>
    </div>

</div>

<!--<script>alert("xss攻击")</script>-->
<!--<h1>xss攻击</h1>-->

<script>

    var app = new Vue(
        {
            el: "#app",
            data: {
                student: {
                    id: null,
                    name: null,
                    sex: null,
                    age: null,
                }
            },
            methods: {
                f: function ()
                {
                    console.log("提交")
                    console.log(this.student)
                    var that = this;
                    //axios发起ajax请求
                    axios({
                        //请求的方式：
                        method: "post",
                        //请求的url:
                        url: "/student",
                        //url参数：
                        params:
                            {},
                        //头信息：
                        headers:
                            {},
                        //请求体参数：
                        data:
                            {
                                id: that.student.id,
                                name: that.student.name,
                                sex: that.student.sex,
                                age: that.student.age,
                            },
                    }).then(response =>
                    {
                        console.log(response);
                        if (response.data === true)
                        {
                            alert("请求成功")
                        }
                        else
                        {
                            alert("失败")
                        }

                    }).catch(error =>
                    {
                        //console.log(error);
                        alert("网络异常！");
                    })
                }
            }
        }
    )

</script>

</body>
</html>
```

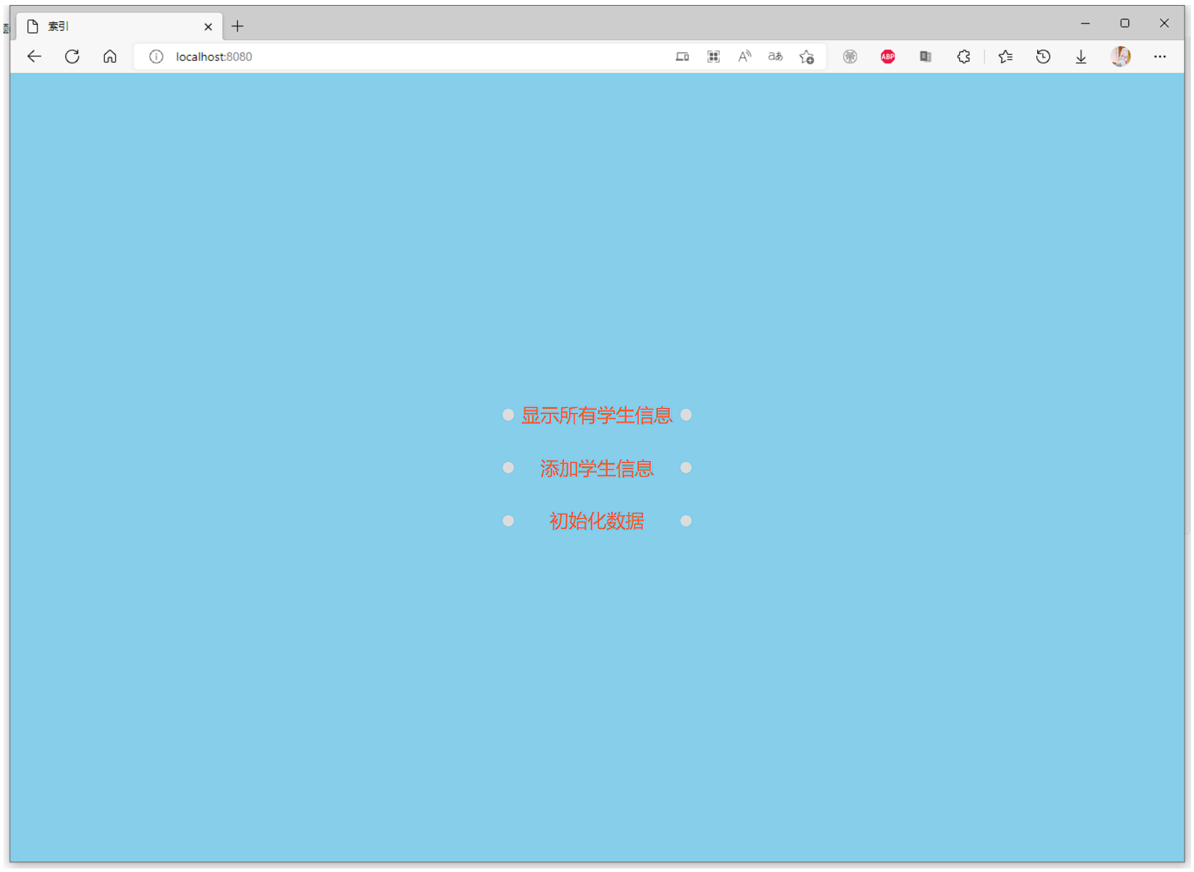
第十一步：启动程序

```

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.1)

2022-10-30 13:14:22.269  INFO 17656 --- [           main]
m.antisamy_demo.AntiSamyDemoApplication  : Starting AntiSamyDemoApplication
using Java 16.0.2 on mao with PID 17656 (H:\程序\大四上期
\demo\antiSamy_demo\target\classes started by mao in H:\程序\大四上期
\demo\antiSamy_demo)
```

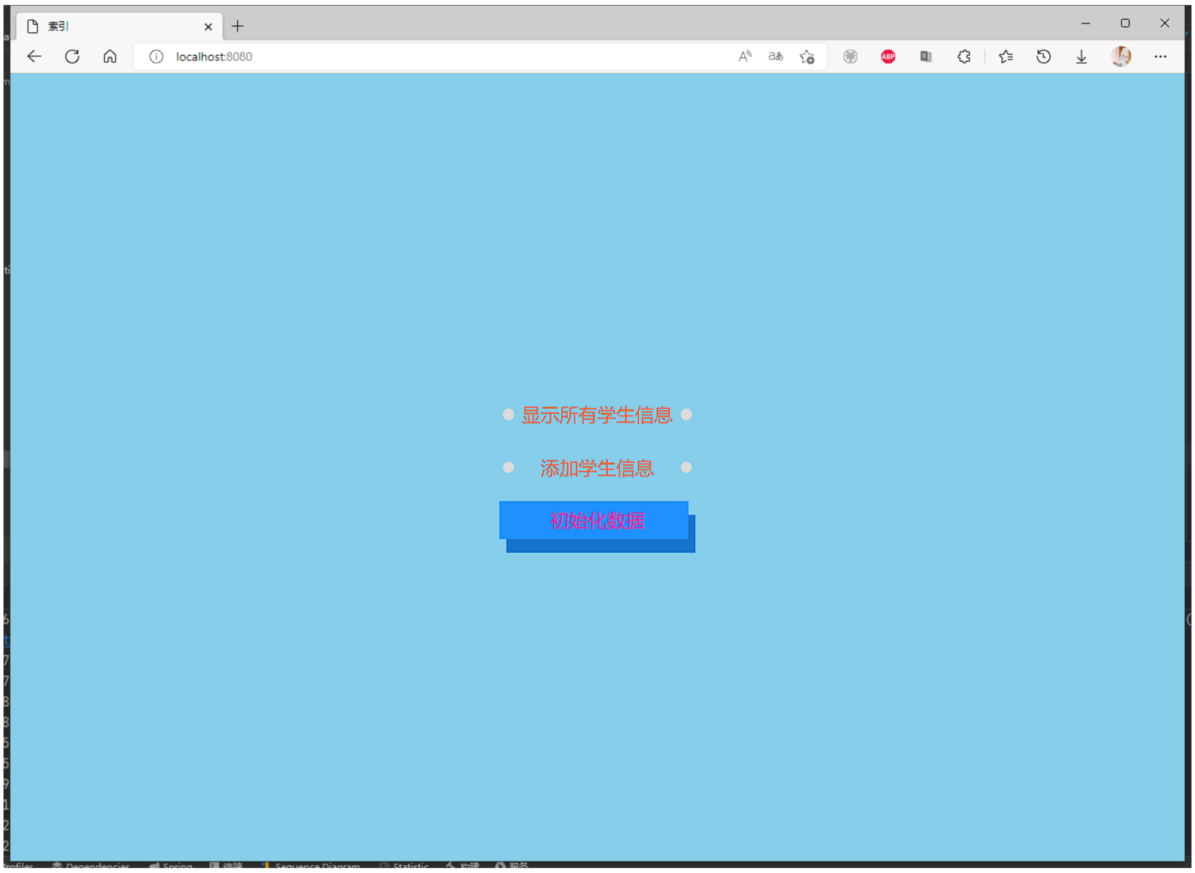
```
2022-10-30 13:14:22.271  INFO 17656 --- [           main] m.antisamy_demo.AntiSamyDemoApplication  : No active profile set, falling back to 1 default profile: "default"
2022-10-30 13:14:22.970  INFO 17656 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2022-10-30 13:14:22.980  INFO 17656 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-10-30 13:14:22.980  INFO 17656 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.64]
2022-10-30 13:14:23.057  INFO 17656 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-10-30 13:14:23.057  INFO 17656 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 746 ms
2022-10-30 13:14:23.198  INFO 17656 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path resource [static/index.html]
2022-10-30 13:14:23.318  INFO 17656 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2022-10-30 13:14:23.328  INFO 17656 --- [           main] m.antisamy_demo.AntiSamyDemoApplication  : Started AntiSamyDemoApplication in 1.355 seconds (JVM running for 2.735)
```

第十二步：访问

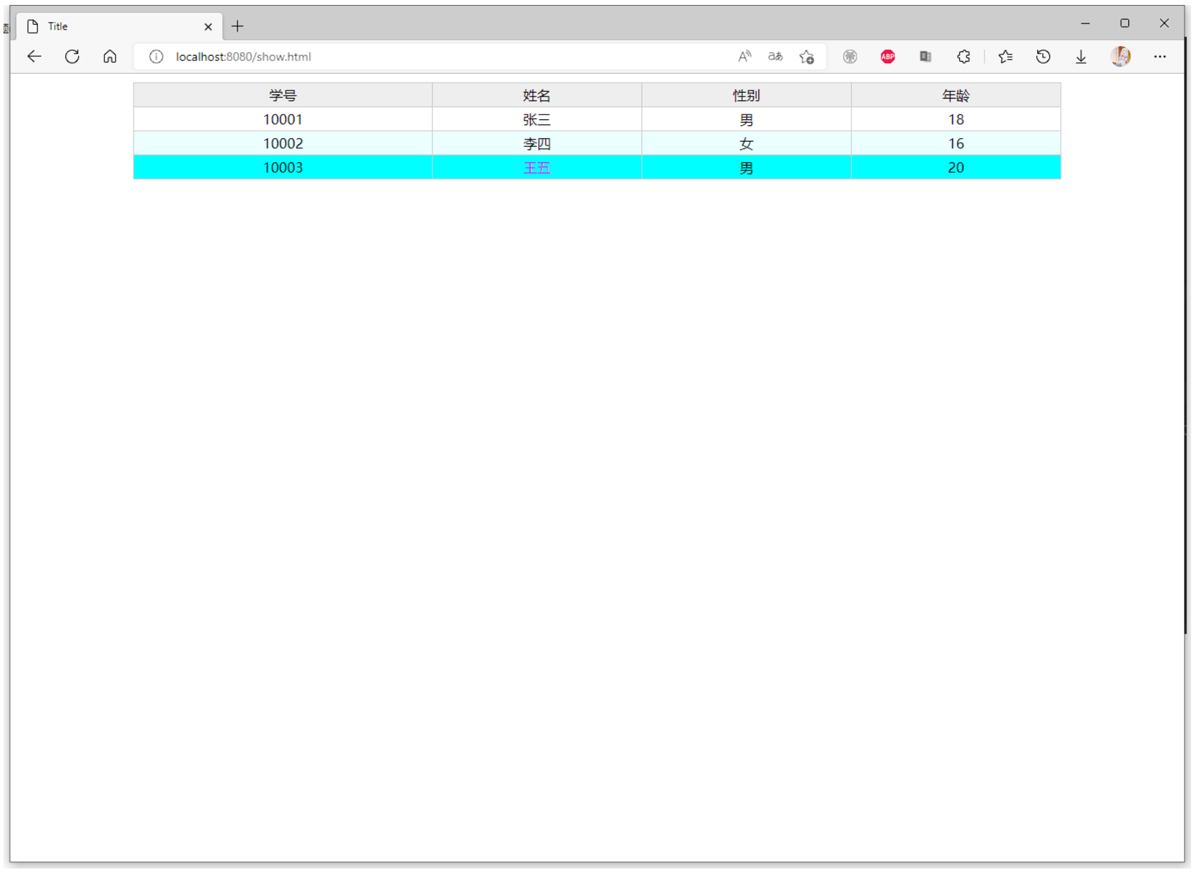
http://localhost:8080/


索引
localhost:8080
显示所有学生信息
添加学生信息
初始化数据

第十三步：初始化数据

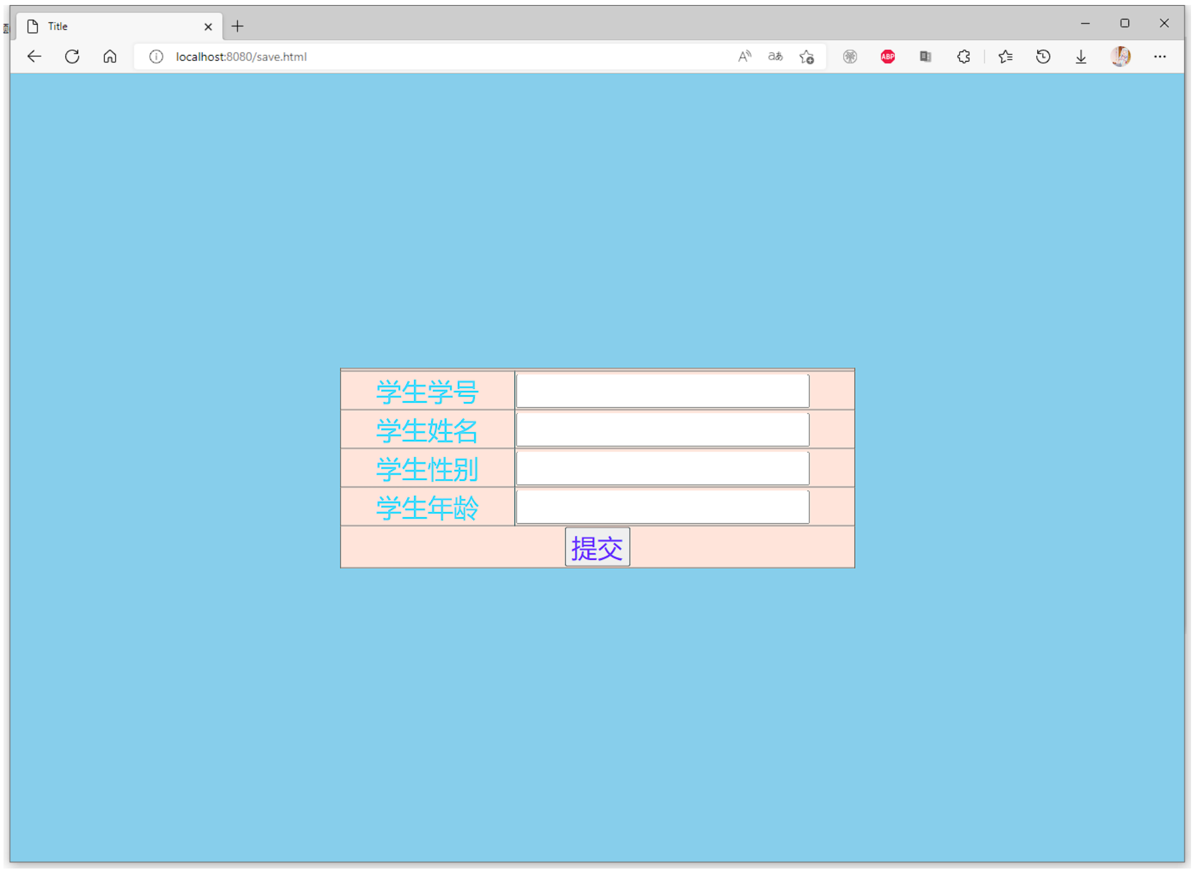

索引
localhost:8080
显示所有学生信息
添加学生信息
初始化数据

第十四步：查看所有学生的信息

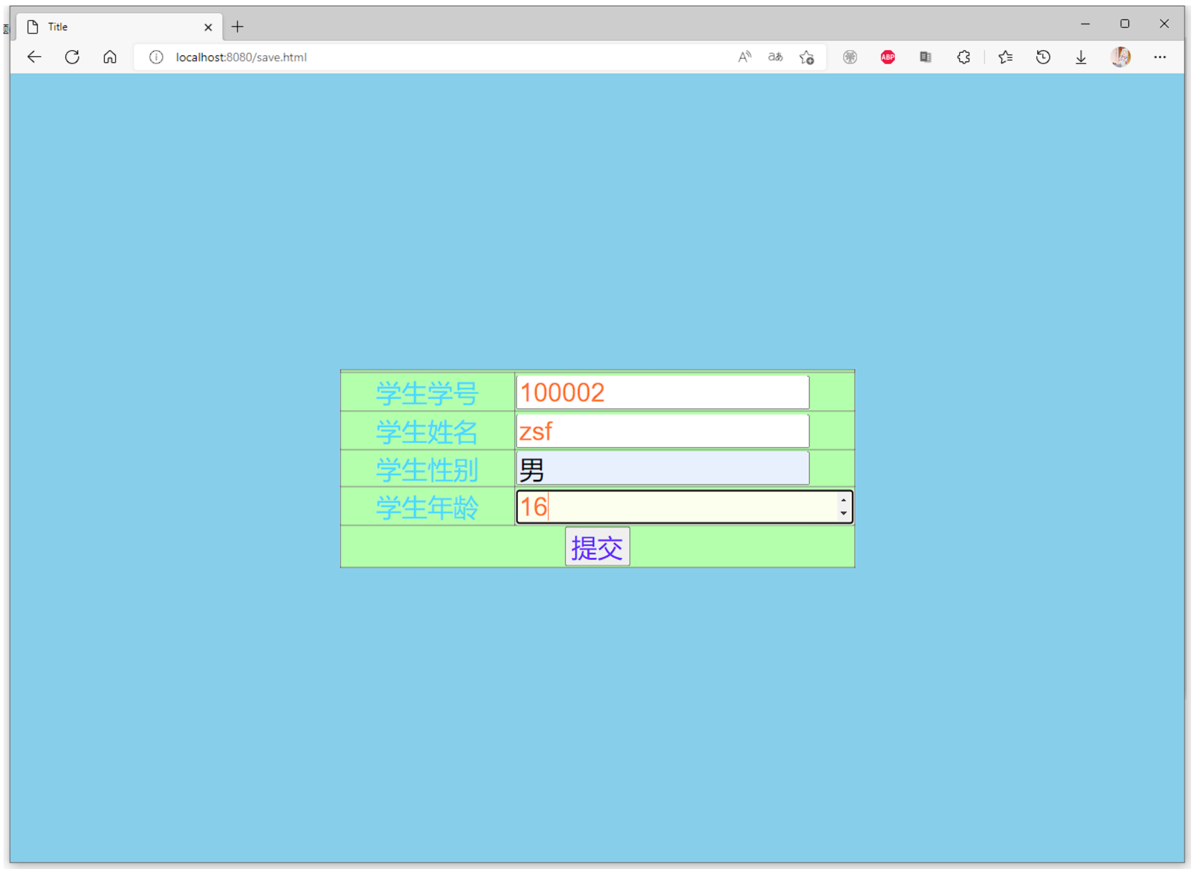


学号	姓名	性别	年龄
10001	张三	男	18
10002	李四	女	16
10003	王五	男	20

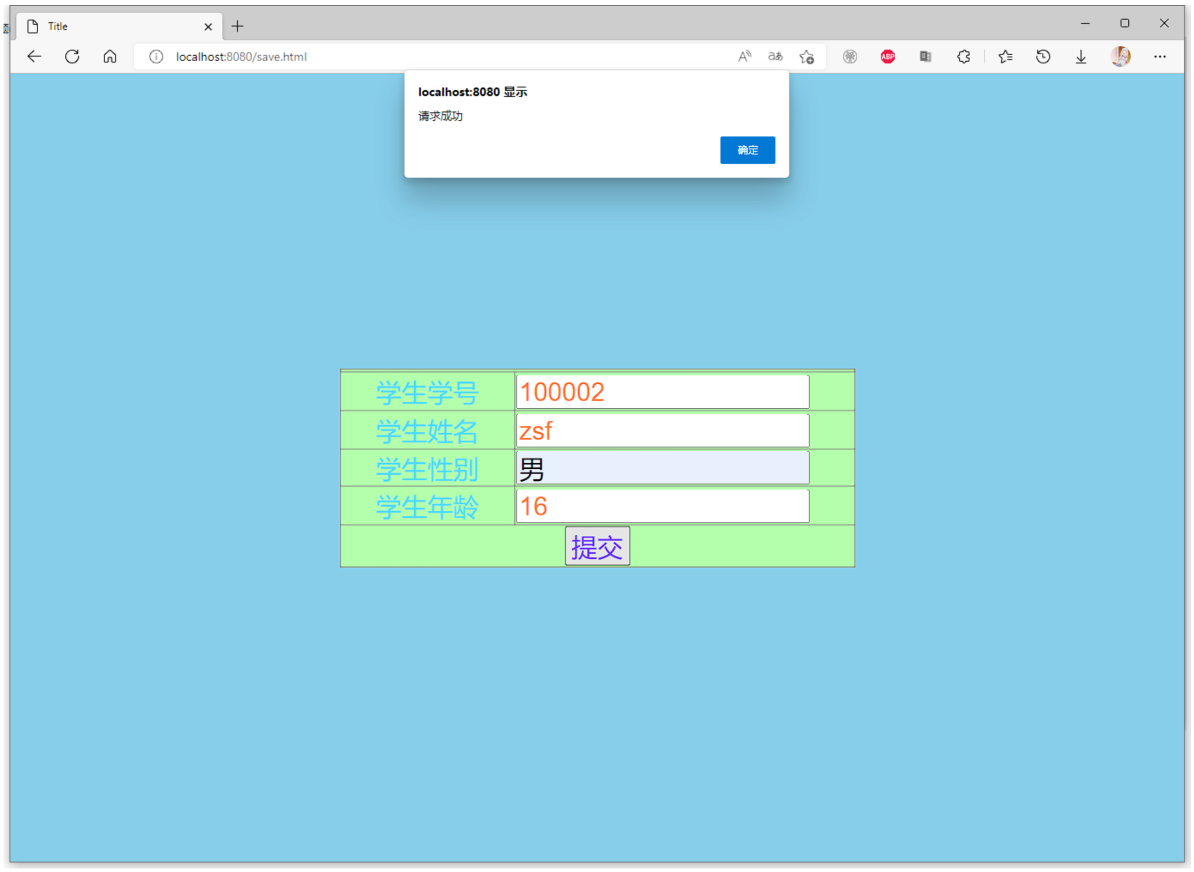
第十五步：添加学生信息



学生学号	
学生姓名	
学生性别	
学生年龄	



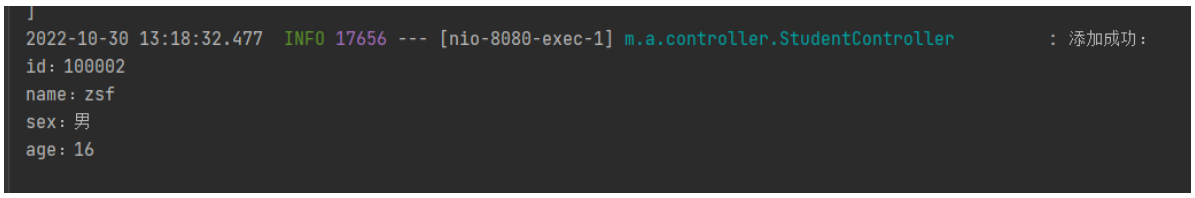
学生学号	100002
学生姓名	zsf
学生性别	男
学生年龄	16



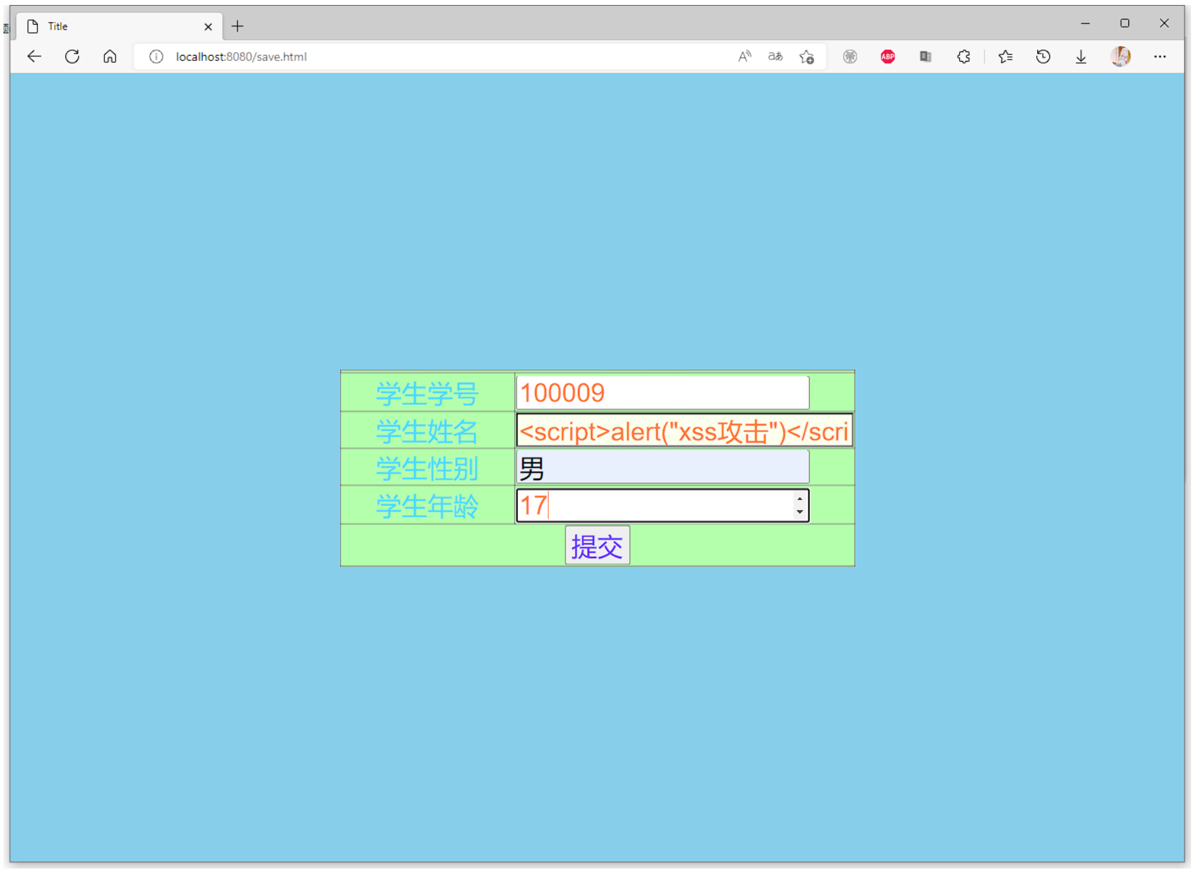
localhost:8080 显示

请求成功

学生学号	100002
学生姓名	zsf
学生性别	男
学生年龄	16



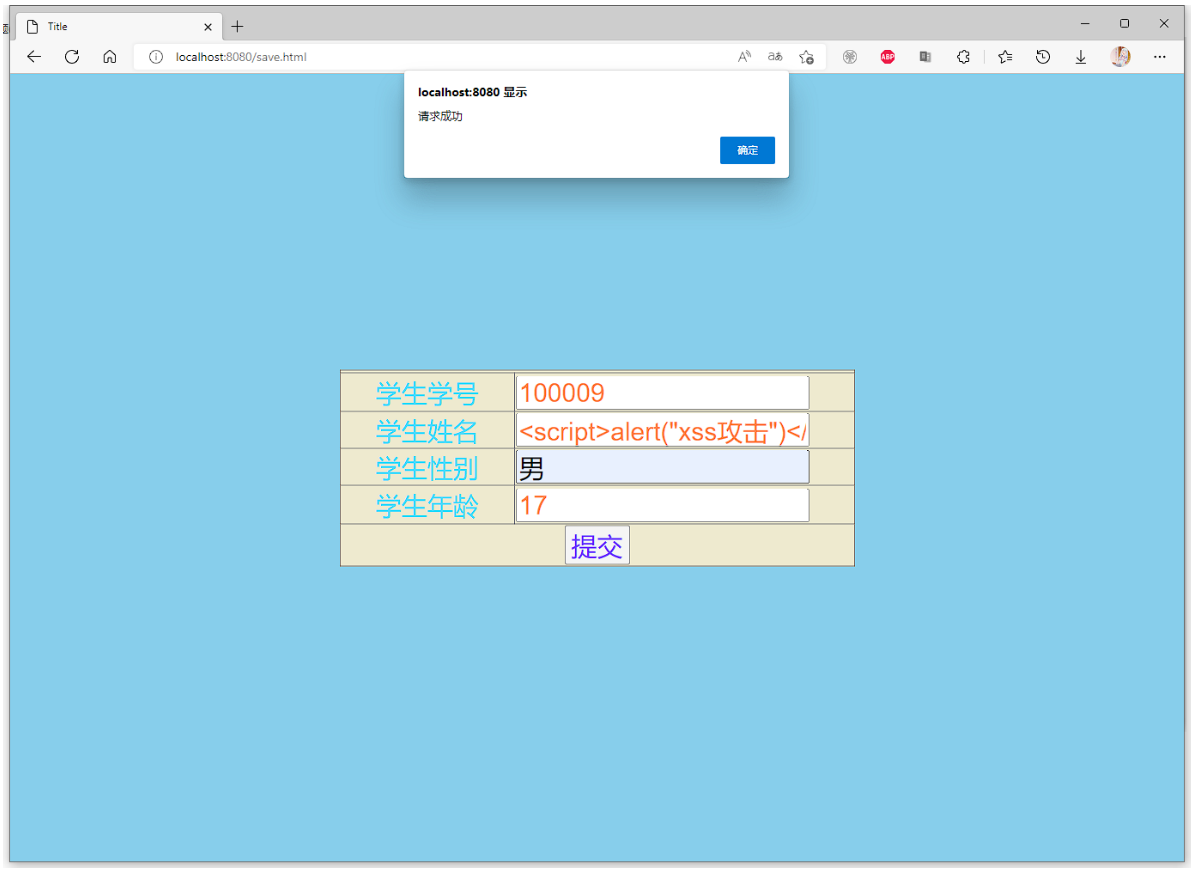
```
2022-10-30 13:18:32.477  INFO 17656 --- [nio-8080-exec-1] m.a.controller.StudentController         : 添加成功：
id：100002
name：zsf
sex：男
age：16
```



Title

localhost:8080/save.html

学生学号	100009
学生姓名	<script>alert("xss攻击")</scri
学生性别	男
学生年龄	17



Title

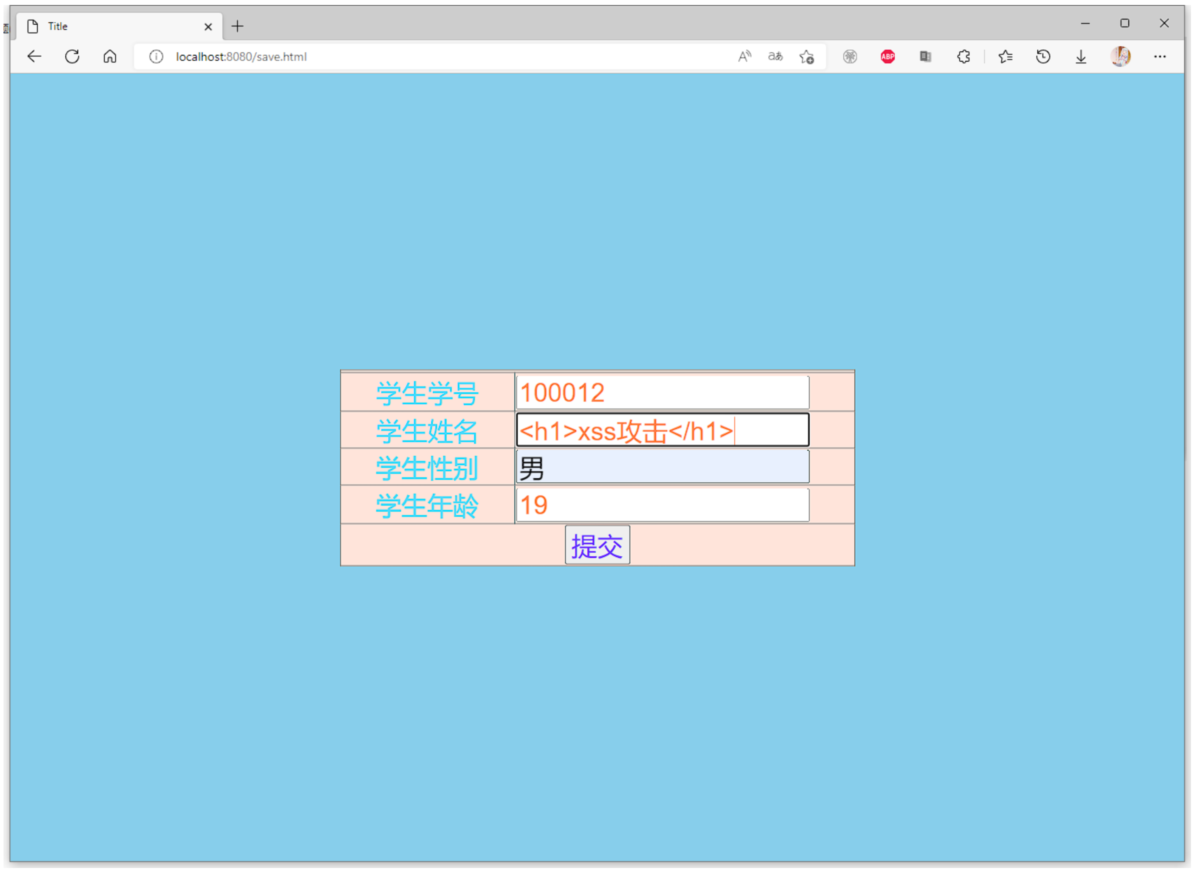
localhost:8080/save.html

localhost:8080 显示

请求成功

学生学号	100009
学生姓名	<script>alert("xss攻击")</
学生性别	男
学生年龄	17

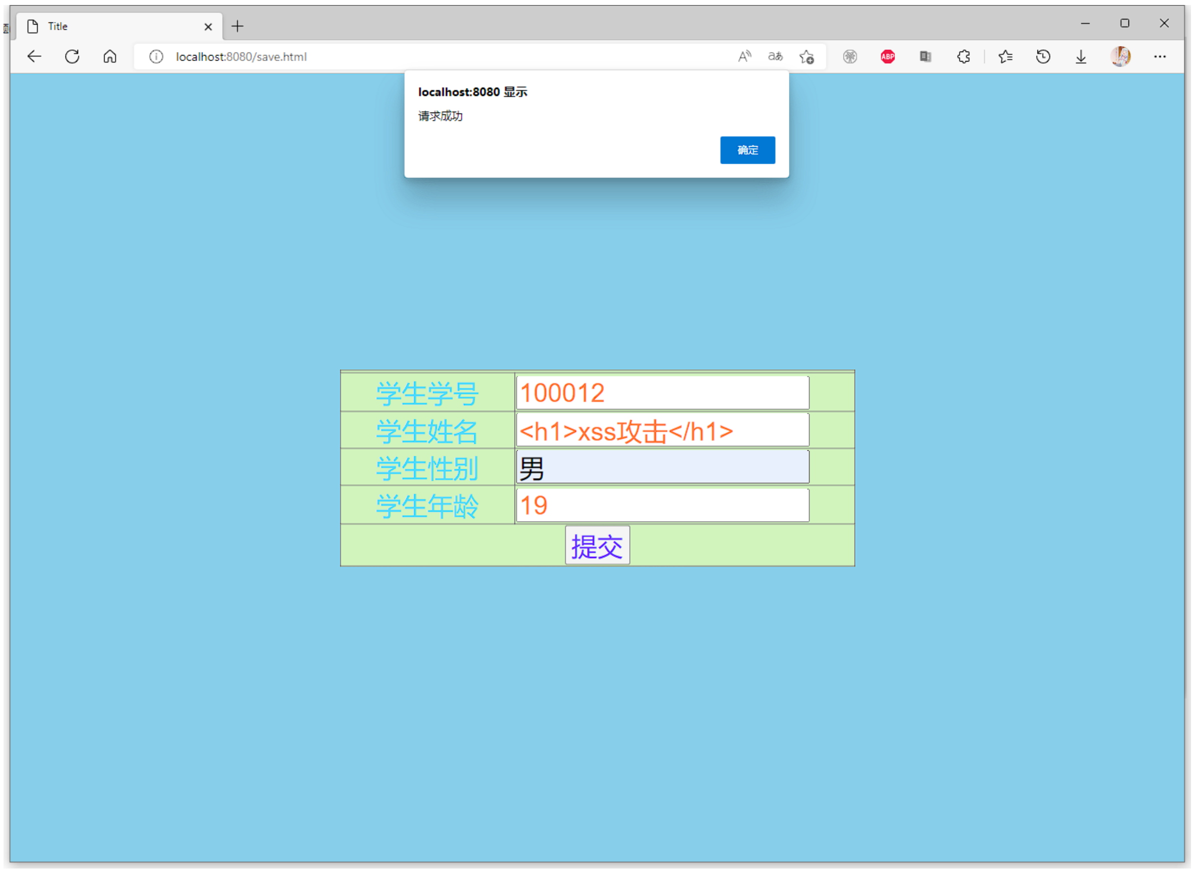
```
age: 18

2022-10-30 13:20:06.888  INFO 17656 --- [nio-8080-exec-3] m.a.controller.StudentController         : 添加成功：
id: 100009
name: <script>alert("xss攻击")</script>
sex: 男
age: 17
```



Title

localhost:8080/save.html

学生学号	100012
学生姓名	<h1>xss攻击</h1>
学生性别	男
学生年龄	19



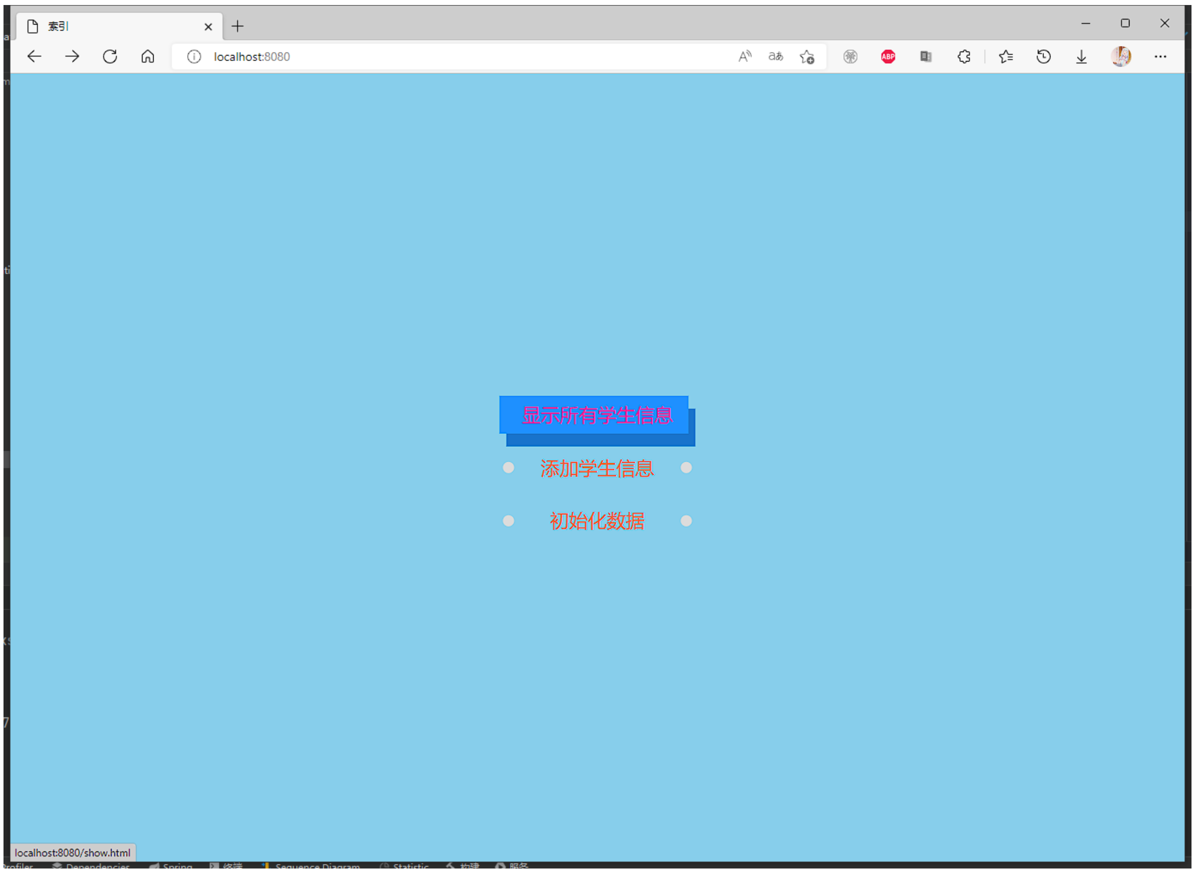
localhost:8080 显示

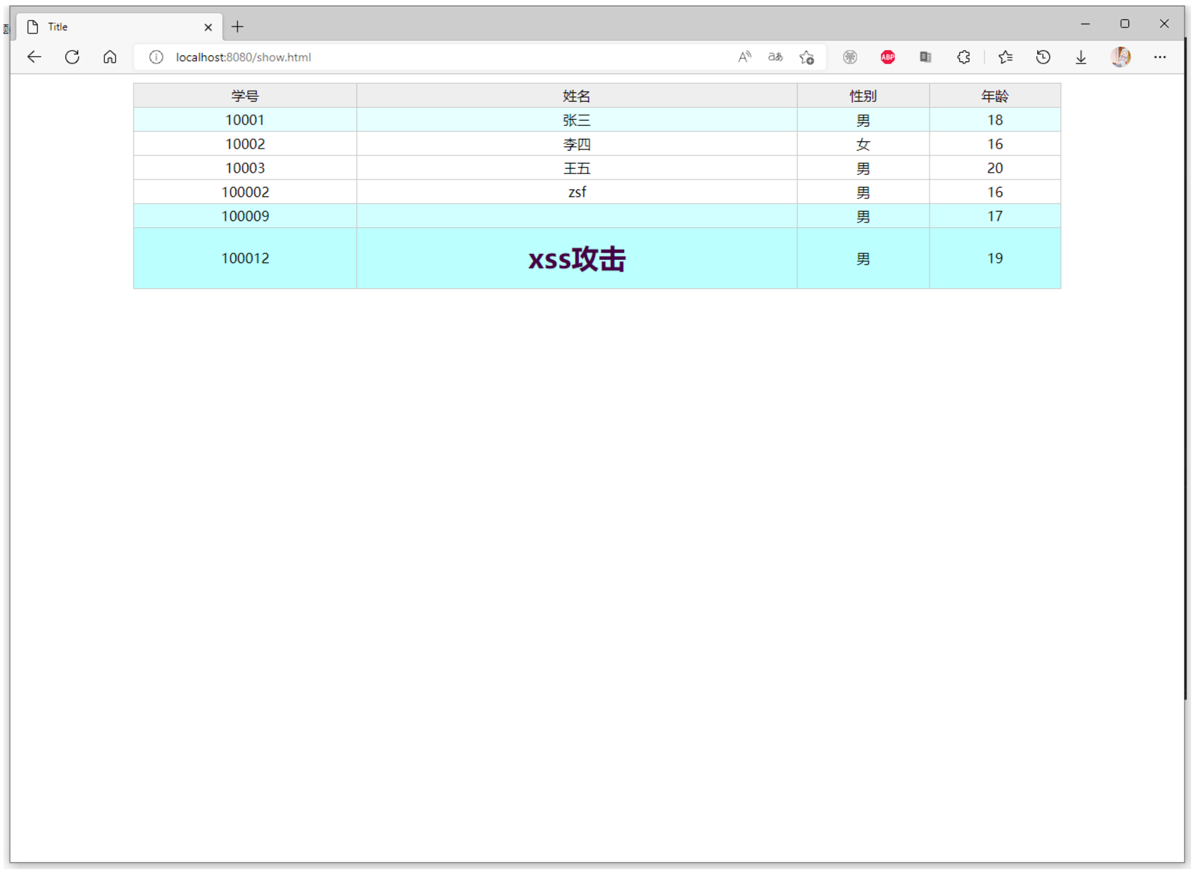
请求成功

学生学号	100012
学生姓名	<h1>xss攻击</h1>
学生性别	男
学生年龄	19

```
2022-10-30 13:20:49.775  INFO 17656 --- [nio-8080-exec-4] m.a.controller.StudentController         : 添加成功：
id：100012
name：<h1>xss攻击</h1>
sex：男
age：19
```

十六步：查询所有学生的信息

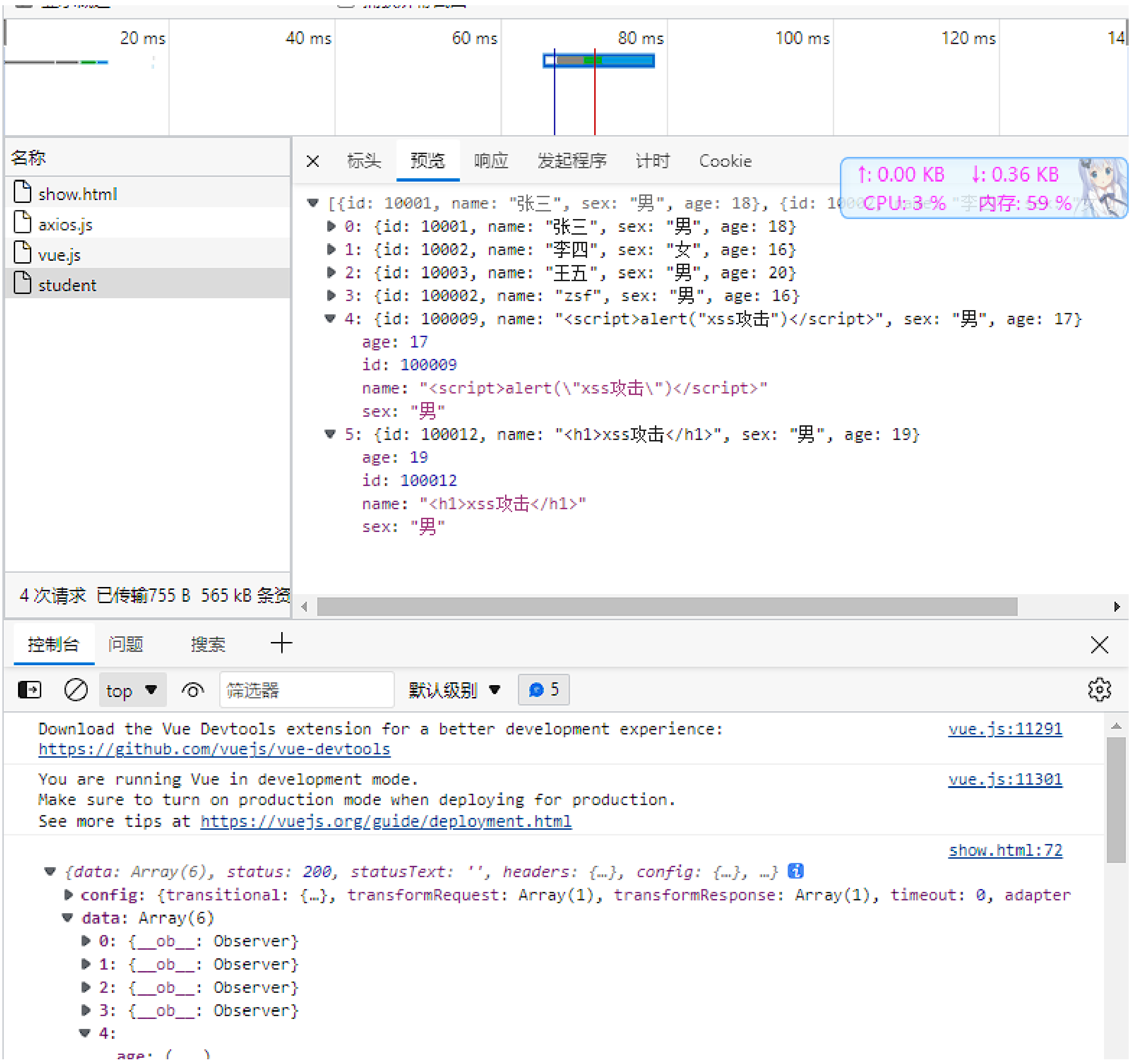




学号	姓名	性别	年龄
10001	张三	男	18
10002	李四	女	16
10003	王五	男	20
100002	zsf	男	16
100009		男	17
100012	**xss攻击**	男	19

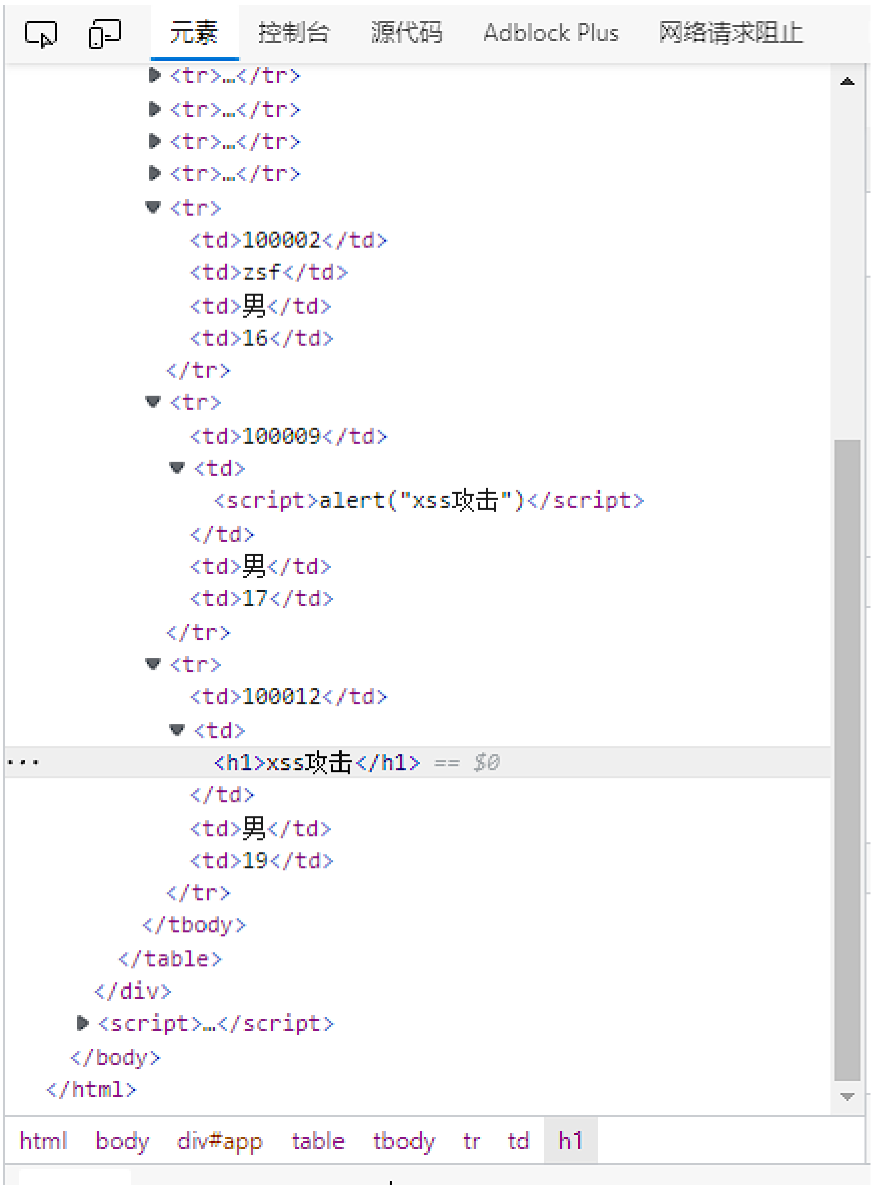
因为是浏览器的原因，浏览器直接把脚本拦截了，但是h1标签还在

数据接收是正常的



```
▼ [{id: 10001, name: "张三", sex: "男", age: 18}, {id: 1000
  ▶ 0: {id: 10001, name: "张三", sex: "男", age: 18}
  ▶ 1: {id: 10002, name: "李四", sex: "女", age: 16}
  ▶ 2: {id: 10003, name: "王五", sex: "男", age: 20}
  ▶ 3: {id: 100002, name: "zsf", sex: "男", age: 16}
  ▼ 4: {id: 100009, name: "<script>alert("xss攻击")</script>", sex: "男", age: 17}
      age: 17
      id: 100009
      name: "<script>alert(\"xss攻击\")</script>"
      sex: "男"
  ▼ 5: {id: 100012, name: "<h1>xss攻击</h1>", sex: "男", age: 19}
      age: 19
      id: 100012
      name: "<h1>xss攻击</h1>"
      sex: "男"
```

```
Download the Vue Devtools extension for a better development experience:        vue.js:11291
https://github.com/vuejs/vue-devtools
You are running Vue in development mode.                                         vue.js:11301
Make sure to turn on production mode when deploying for production.
See more tips at https://vuejs.org/guide/deployment.html
                                                                                 show.html:72
▼ {data: Array(6), status: 200, statusText: '', headers: {…}, config: {…}, …}
  ▶ config: {transitional: {…}, transformRequest: Array(1), transformResponse: Array(1), timeout: 0, adapter
  ▼ data: Array(6)
    ▶ 0: {__ob__: Observer}
    ▶ 1: {__ob__: Observer}
    ▶ 2: {__ob__: Observer}
    ▶ 3: {__ob__: Observer}
    ▼ 4:
        age: (...)
```



```
▶ <tr>…</tr>
▶ <tr>…</tr>
▶ <tr>…</tr>
▶ <tr>…</tr>
▼ <tr>
    <td>100002</td>
    <td>zsf</td>
    <td>男</td>
    <td>16</td>
  </tr>
▼ <tr>
    <td>100009</td>
  ▼ <td>
      <script>alert("xss攻击")</script>
    </td>
    <td>男</td>
    <td>17</td>
  </tr>
▼ <tr>
    <td>100012</td>
  ▼ <td>
      <h1>xss攻击</h1> == $0
    </td>
    <td>男</td>
    <td>19</td>
  </tr>
</tbody>
</table>
</div>
▶ <script>…</script>
</body>
</html>
```

解决xss攻击

第一步：添加依赖

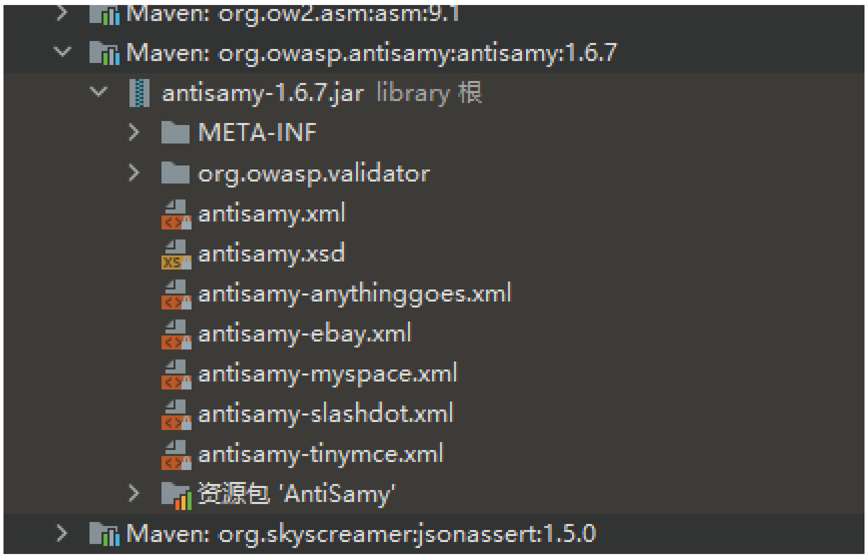
```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
```

```xml
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.7.1</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>mao</groupId>
    <artifactId>antiSamy_demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>antiSamy_demo</name>
    <description>antiSamy_demo</description>
    <properties>
        <java.version>11</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <!--解决xss攻击-->
        <dependency>
            <groupId>org.owasp.antisamy</groupId>
            <artifactId>antisamy</artifactId>
            <version>1.5.7</version>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>

```

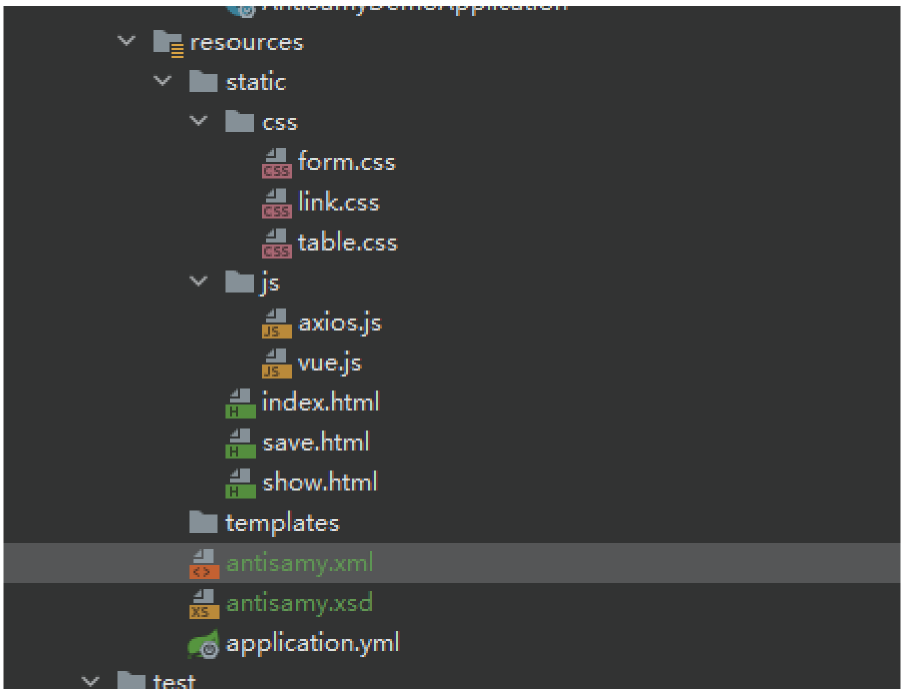
第二步：创建策略文件/resources/antisamy.xml

文件内容可以从antisamy的jar包中获取

AntiSamy对“恶意代码”的过滤依赖于策略文件。策略文件规定了AntiSamy对各个标签、属性的处理方法，策略文件定义的严格与否，决定了AntiSamy对XSS漏洞的防御效果。在AntiSamy的jar包中，包含了几个常用的策略文件



```
> Maven: org.ow2.asm:asm:9.1
v Maven: org.owasp.antisamy:antisamy:1.6.7
    v antisamy-1.6.7.jar  library 根
        > META-INF
        > org.owasp.validator
          antisamy.xml
          antisamy.xsd
          antisamy-anythinggoes.xml
          antisamy-ebay.xml
          antisamy-myspace.xml
          antisamy-slashdot.xml
          antisamy-tinymce.xml
        > 资源包 'AntiSamy'
> Maven: org.skyscreamer:jsonassert:1.5.0
```



```
  v resources
      v static
          v css
              form.css
              link.css
              table.css
          v js
              axios.js
              vue.js
            index.html
            save.html
            show.html
        templates
        antisamy.xml
        antisamy.xsd
        application.yml
v test
```

antisamy.xml

默认规则，允许大部分HTML通过

antisamy-slashdot.xml
用户只能提交下列的html标签：<b>, <u>, <i>, <a>, <blockquote>。

antisamy-ebay.xml
用户可以输入一系列的HTML的内容，不包含JavaScript。

antisamy-myspace.xml

更多的HTML和CSS，只要不包含JavaScript。

antisamy-anythinggoes.xml
更多的HTML和CSS元素输入，但不包含JavaScript。

antisamy-tinymce.xml

只允许文本格式通过，相对较安全

此时我们可以启动项目进行访问，但是还没有进行参数的过滤，所以如果我们输入任意参数都可以正常传递到Controller中，这在实际项目中是非常不安全的。为了对我们输入的数据进行过滤清理，需要通过过滤器来实现。

第三步：创建过滤器，用于过滤所有提交到服务器的请求参数

```java
package mao.antisamy_demo.filter;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;

/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.filter
 * Class(类名): XssFilter
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/30
 * Time(创建时间)： 13:44
 * Version(版本): 1.0
 * Description(描述)： 无
 */

public class XssFilter implements Filter
{

    @Override
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain filterChain)
```

```java
            throws IOException, ServletException
    {
        HttpServletRequest request = (HttpServletRequest) servletRequest;
        //传入重写后的Request
        filterChain.doFilter(new XssRequestWrapper(request),
servletResponse);
    }
}
```

通过上面的过滤器可以发现我们并没有在过滤器中直接进行请求参数的过滤清理，而是直接放行了，那么我们还怎么进行请求参数的过滤清理呢？其实过滤清理的工作是在另外一个类XssRequestWrapper中进行的，当上面的过滤器放行时需要调用filterChain.doFilter()方法，此方法需要传入请求Request对象，此时我们可以将当前的request对象进行包装，而XssRequestWrapper就是Request对象的包装类，在过滤器放行时会自动调用包装类的getParameterValues方法，我们可以在包装类的getParameterValues方法中进行统一的请求参数过滤清理。

第四步：创建XssRequestWrapper类

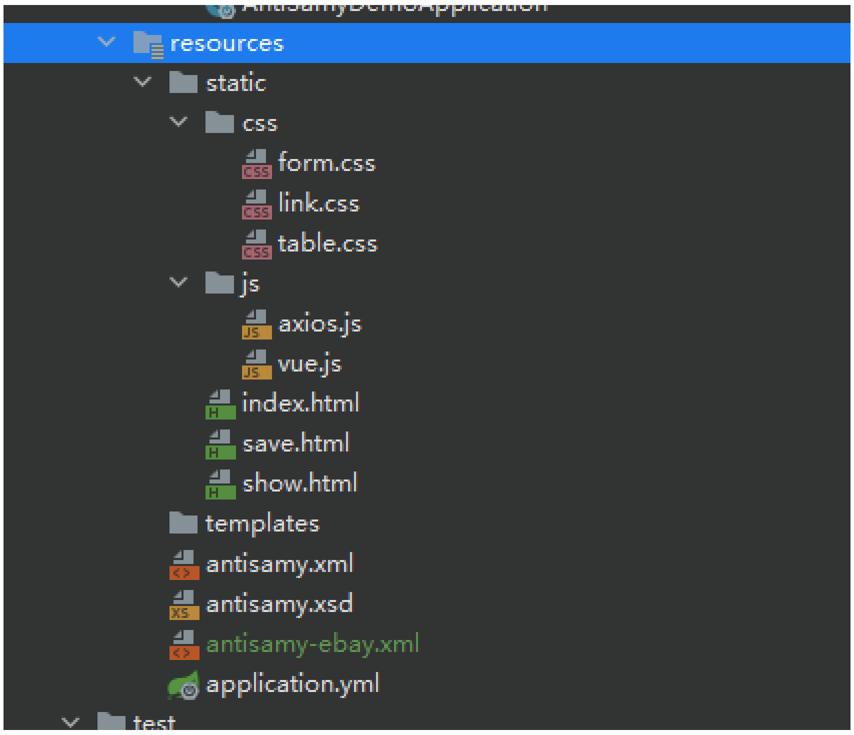
```java
package mao.antisamy_demo.wrapper;

import org.owasp.validator.html.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;
import java.io.InputStream;


/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.wrapper
 * Class(类名): XssRequestWrapper
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/30
 * Time(创建时间)： 13:46
 * Version(版本): 1.0
 * Description(描述)： 无
 */

public class XssRequestWrapper extends HttpServletRequestWrapper
{
    /**
     * 策略文件 需要将要使用的策略文件放到项目资源文件路径下
     */
    @SuppressWarnings("all")
    private static final String antiSamyPath = "antisamy.xml";

    public static Policy policy = null;

    private static final Logger log =
LoggerFactory.getLogger(XssRequestWrapper.class);

    static
    {
        // 指定策略文件
        try
        {
            InputStream inputStream =
XssRequestWrapper.class.getClassLoader().getResourceAsStream(antiSamyPath);
            assert inputStream != null;
            policy = Policy.getInstance(inputStream);
        }
        catch (PolicyException e)
        {
            e.printStackTrace();
        }
    }

    /**
     * AntiSamy过滤数据
     *
     * @param taintedHTML 需要进行过滤的数据
     * @return 返回过滤后的数据
     */
    private String xssClean(String taintedHTML)
    {
        try
        {
            // 使用AntiSamy进行过滤
            AntiSamy antiSamy = new AntiSamy();
            CleanResults cleanResults = antiSamy.scan(taintedHTML, policy);
            taintedHTML = cleanResults.getCleanHTML();
        }
        catch (ScanException | PolicyException e)
        {
            e.printStackTrace();
        }
        return taintedHTML;
    }

    public XssRequestWrapper(HttpServletRequest request)
    {
        super(request);
    }

    @Override
    public String[] getParameterValues(String name)
    {
        String[] values = super.getParameterValues(name);
        if (values == null)
        {
            return null;
        }
        int len = values.length;
        String[] newArray = new String[len];
        for (int j = 0; j < len; j++)
        {
            log.info("Antisamy过滤清理，清理之前的参数值：" + values[j]);
            // 过滤清理
            newArray[j] = xssClean(values[j]);
            log.info("Antisamy过滤清理，清理之后的参数值：" + newArray[j]);
        }
        return newArray;
    }
}
```

第五步：使上面定义的过滤器生效，创建配置类，用于初始化过滤器对象

```java
package mao.antisamy_demo.config;

import mao.antisamy_demo.filter.XssFilter;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.config
 * Class(类名): AntiSamyConfig
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/30
 * Time(创建时间)： 13:53
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@Configuration
public class AntiSamyConfig
{
    @Bean
    public FilterRegistrationBean<XssFilter> filterRegistrationBean()
    {
        FilterRegistrationBean<XssFilter> filterRegistrationBean = new
FilterRegistrationBean<>(new XssFilter());
        filterRegistrationBean.addUrlPatterns("/*");
        filterRegistrationBean.setOrder(1);
        return filterRegistrationBean;
    }
}
```

注意：当前我们在进行请求参数过滤时只是在包装类的getParameterValues方法中进行了处理，真实项目中可能用户提交的数据在请求头中，也可能用户提交的是json数据，所以如果考虑所有情况，我们可以在包装类中的多个方法中都进行清理处理即可

第六步：复制antisamy-ebay.xml文件到/resources目录下



第七步：修改XssRequestWrapper类

```java
package mao.antisamy_demo.wrapper;

import org.owasp.validator.html.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;
import java.io.InputStream;
import java.util.Map;


/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.wrapper
 * Class(类名): XssRequestWrapper
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/30
 * Time(创建时间)： 13:46
 * Version(版本): 1.0
 * Description(描述)： 无
 */

public class XssRequestWrapper extends HttpServletRequestWrapper
{
    /**
     * 策略文件 需要将要使用的策略文件放到项目资源文件路径下
     */
    @SuppressWarnings("all")
    private static final String antiSamyPath = "antisamy-ebay.xml";

    public static Policy policy = null;

    private static final Logger log =
LoggerFactory.getLogger(XssRequestWrapper.class);

    static
    {
        // 指定策略文件
        try
        {
            InputStream inputStream =
XssRequestWrapper.class.getClassLoader().getResourceAsStream(antiSamyPath);
            assert inputStream != null;
            policy = Policy.getInstance(inputStream);
        }
        catch (PolicyException e)
        {
            e.printStackTrace();
        }
    }

    /**
     * AntiSamy过滤数据
     *
     * @param taintedHTML 需要进行过滤的数据
     * @return 返回过滤后的数据
     */
    private String xssClean(String taintedHTML)
    {
        try
        {
            // 使用AntiSamy进行过滤
            AntiSamy antiSamy = new AntiSamy();
            CleanResults cr = antiSamy.scan(taintedHTML, policy);
            taintedHTML = cr.getCleanHTML();
        }
        catch (ScanException | PolicyException e)
        {
            e.printStackTrace();
        }
        return taintedHTML;
    }

    public XssRequestWrapper(HttpServletRequest request)
    {
        super(request);
    }

    @Override
    public String[] getParameterValues(String name)
    {
        String[] values = super.getParameterValues(name);
        if (values == null)
        {
            return null;
        }
        int len = values.length;
        String[] newArray = new String[len];
        for (int j = 0; j < len; j++)
        {
            // 过滤清理
            newArray[j] = xssClean(values[j]);
            log.debug("Antisamy过滤清理，清理之前的参数值：" + values[j]);
            log.debug("Antisamy过滤清理，清理之后的参数值：" + newArray[j]);
        }
        return newArray;
    }

    @Override
    public String getParameter(String paramString)
    {
        String str = super.getParameter(paramString);
        if (str == null)
        {
            return null;
        }
        return xssClean(str);
    }


    @Override
    public String getHeader(String paramString)
    {
        String str = super.getHeader(paramString);
        if (str == null)
        {
            return null;
        }
        return xssClean(str);
    }

    @Override
```

```java
    public Map<String, String[]> getParameterMap()
    {
        Map<String, String[]> requestMap = super.getParameterMap();
        for (Map.Entry<String, String[]> stringEntry : requestMap.entrySet())
        {
            String[] values = stringEntry.getValue();
            for (int i = 0; i < values.length; i++)
            {
                values[i] = xssClean(values[i]);
            }
        }
        return requestMap;
    }
}
```

第八步：启动程序

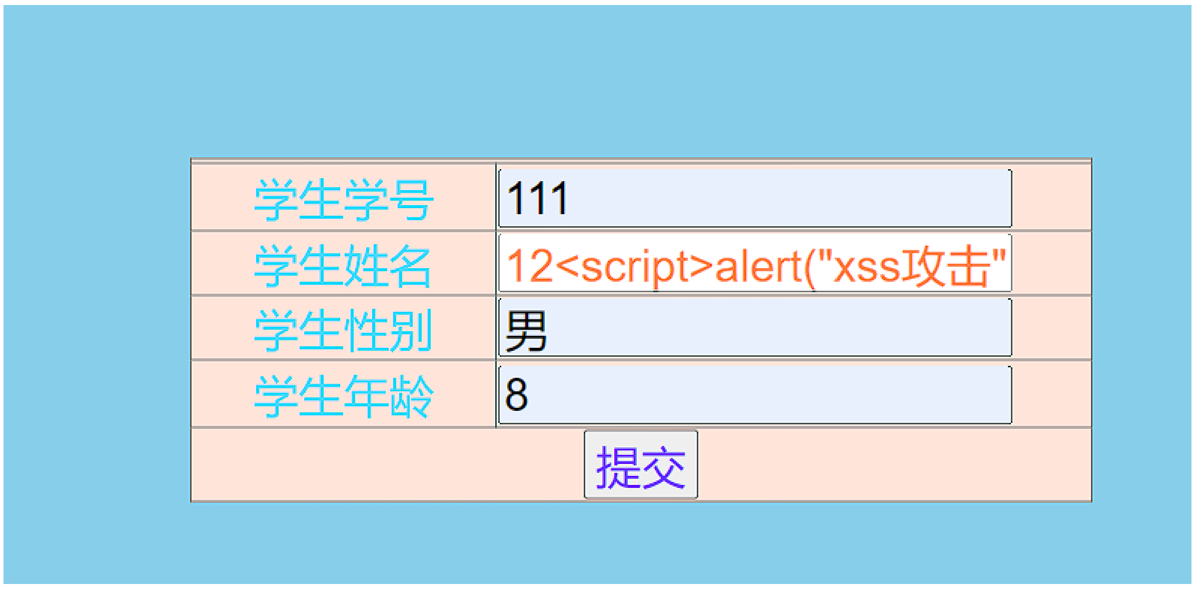
```

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.1)

2022-10-30 15:29:40.631  INFO 19548 --- [           main] m.antisamy_demo.AntiSamyDemoApplication  : Starting AntiSamyDemoApplication using Java 16.0.2 on mao with PID 19548 (H:\程序\大四上期\demo\antiSamy_demo\target\classes started by mao in H:\程序\大四上期\demo\antiSamy_demo)
2022-10-30 15:29:40.634  INFO 19548 --- [           main] m.antisamy_demo.AntiSamyDemoApplication  : No active profile set, falling back to 1 default profile: "default"
2022-10-30 15:29:41.269  INFO 19548 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2022-10-30 15:29:41.277  INFO 19548 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-10-30 15:29:41.277  INFO 19548 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.64]
2022-10-30 15:29:41.355  INFO 19548 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-10-30 15:29:41.355  INFO 19548 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 682 ms
```

```
2022-10-30 15:29:41.494  INFO 19548 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path resource [static/index.html]
2022-10-30 15:29:41.596  INFO 19548 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2022-10-30 15:29:41.605  INFO 19548 --- [           main] m.antisamy_demo.AntiSamyDemoApplication  : Started AntiSamyDemoApplication in 1.262 seconds (JVM running for 1.729)
```

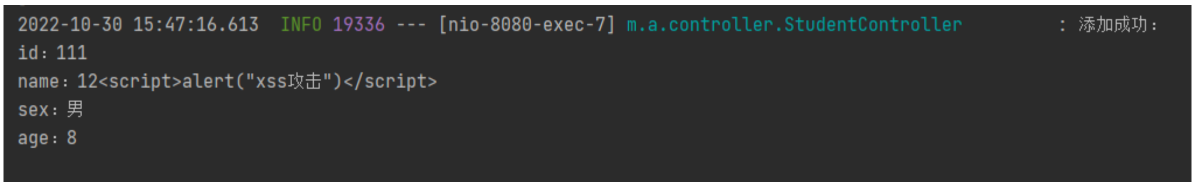
到了这里，我发现一个问题，那就是无法清理通过请求体传过来json数据的情况，有时候需要将数据封装到请求体中在使用json的格式发送到后端。

对此，有三种方案，第一种就是放弃异步提交的方式，采用同步提交，就是传统的表单提交，第二种就是在controller里编写繁琐的业务代码，将没一个字段都过滤一次，第三种就是重写JsonDeserializer

第九步：访问



学生学号	111
学生姓名	12<script>alert("xss攻击"
学生性别	男
学生年龄	8



```
2022-10-30 15:47:16.613  INFO 19336 --- [nio-8080-exec-7] m.a.controller.StudentController         : 添加成功：
id：111
name：12<script>alert("xss攻击")</script>
sex：男
age：8
```

请求体的数据不会被过滤

以下是解决方案1

第十步：更改StudentController

```java
package mao.antisamy_demo.controller;

import mao.antisamy_demo.entity.Student;
import mao.antisamy_demo.wrapper.XssRequestWrapper;
import org.apache.juli.logging.Log;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.controller
 * Class(类名): StudentController
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/29
 * Time(创建时间)： 20:05
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@RestController
@RequestMapping("/student")
public class StudentController
{
    private static final List<Student> list = 
Collections.synchronizedList(new ArrayList<>());

    private static final Logger log = 
LoggerFactory.getLogger(StudentController.class);


    @PostMapping("/init")
    public synchronized void init()
    {
        Student student1 = new Student(10001, "张三", "男", 18);
        Student student2 = new Student(10002, "李四", "女", 16);
        Student student3 = new Student(10003, "王五", "男", 20);
        list.clear();
        list.add(student1);
        list.add(student2);
        list.add(student3);
        log.info("初始化完成");
    }

    @PostMapping
    public boolean save(@RequestBody Student student)
    {
        list.add(student);
        log.info("添加成功：\n" + student);
        return true;
    }

    @PostMapping("/sync")
    public boolean saveSync(Student student)
    {
        list.add(student);
        log.info("添加成功：\n" + student);
        return true;
    }

    @GetMapping
    public List<Student> getAll()
    {
        log.info("查询所有：\n" + list);
        return list;
    }

}
```

第十一步：编写save2.html

```html
<!DOCTYPE html>

<!--
Project name(项目名称)：antiSamy_demo
  File name(文件名): save
  Authors(作者）: mao
  Author QQ：1296193245
  GitHub：https://github.com/maomao124/
  Date(创建日期)：  2022/10/29
  Time(创建时间)：  20:23
  Description(描述)：  无
-->

<html lang="en">
<head>
```

```html
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" href="css/form.css">
    <script src="js/axios.js"></script>
    <script src="js/vue.js"></script>
    <style>
        body {
            background-color: skyblue;
        }
    </style>
</head>
<body>

<div id="app">


    <div class="form_position">
        <div class="animated bounceInDown">
            <div class="form">
                <form action="/student/sync" method="post">
                    <table border="1">
                        <tr>
                            <td colspan="2" align="center">
                            </td>
                        </tr>
                        <tr>
                            <td class="prompt">学生学号</td>
                            <td>
                                <label>
                                    <input v-model="student.id" class="input" type="number" name="id">
                                </label>
                            </td>
                        </tr>
                        <tr>
                            <td class="prompt">学生姓名</td>
                            <td>
                                <label>
                                    <input v-model="student.name" class="input" type="text" name="name">
                                </label>
                            </td>
                        </tr>
                        <tr>
                            <td class="prompt">学生性别</td>
                            <td>
                                <label>
                                    <input v-model="student.sex" class="input" type="text" name="sex">
                                </label>
                            </td>
                        </tr>
                        <tr>
                            <td class="prompt">学生年龄</td>
                            <td>
                                <label>
                                    <input v-model="student.age" class="input" type="number" name="age">
                                </label>
                            </td>
                        </tr>
                        <tr>
                            <td colspan="2" align="center">
                                <input @click="f()" class="submit"
type="submit" value="提交"/>
                            </td>
                        </tr>
                    </table>
                </form>
            </div>
        </div>
    </div>

</div>

<!--<script>alert("xss攻击")</script>-->
<!--<h1>xss攻击</h1>-->

<script>

    // var app = new Vue(
    //     {
    //         el: "#app",
    //         data: {
    //             student: {
    //                 id: null,
    //                 name: null,
    //                 sex: null,
    //                 age: null,
    //             }
    //         },
    //         methods: {
    //             f: function ()
    //             {
    //                 console.log("提交")
    //                 console.log(this.student)
    //                 var that = this;
    //                 //axios发起ajax请求
    //                 axios({
    //                     //请求的方式:
    //                     method: "post",
    //                     //请求的url:
    //                     url: "/student",
    //                     //url参数:
    //                     params:
    //                         {},
    //                     //头信息:
    //                     headers:
    //                         {},
    //                     //请求体参数:
    //                     data:
    //                         {
    //                             id: that.student.id,
    //                             name: that.student.name,
    //                             sex: that.student.sex,
    //                             age: that.student.age,
    //                          },
    //                  }).then(response =>
    //                  {
    //                      console.log(response);
    //                      if (response.data === true)
    //                      {
    //                          alert("请求成功")
    //                      }
    //                      else
    //                      {
    //                          alert("失败")
    //                      }
    //
    //                  }).catch(error =>
    //                  {
    //                      //console.log(error);
    //                      alert("网络异常！");
    //                  })
    //              }
    //          }
    //      }
    // )

</script>

</body>
</html>
```

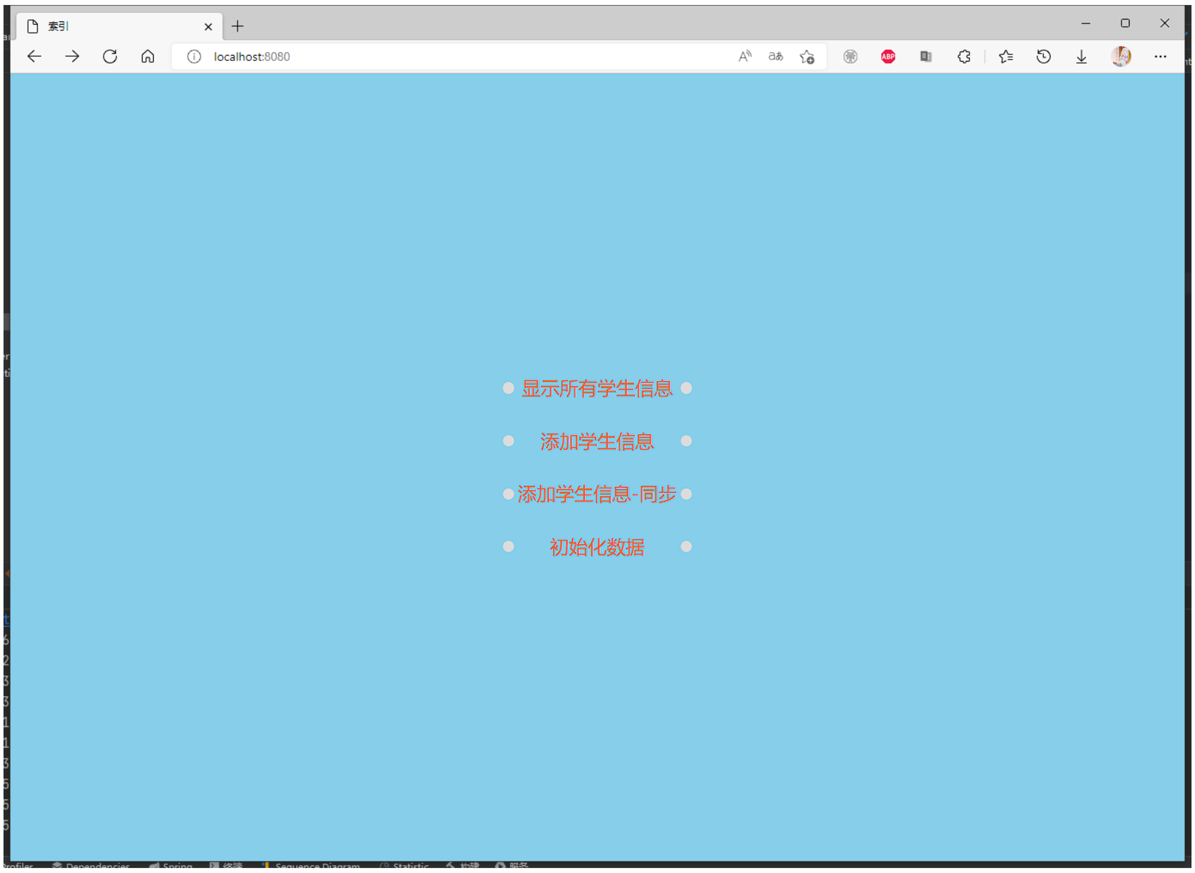
第十二步：修改index.html

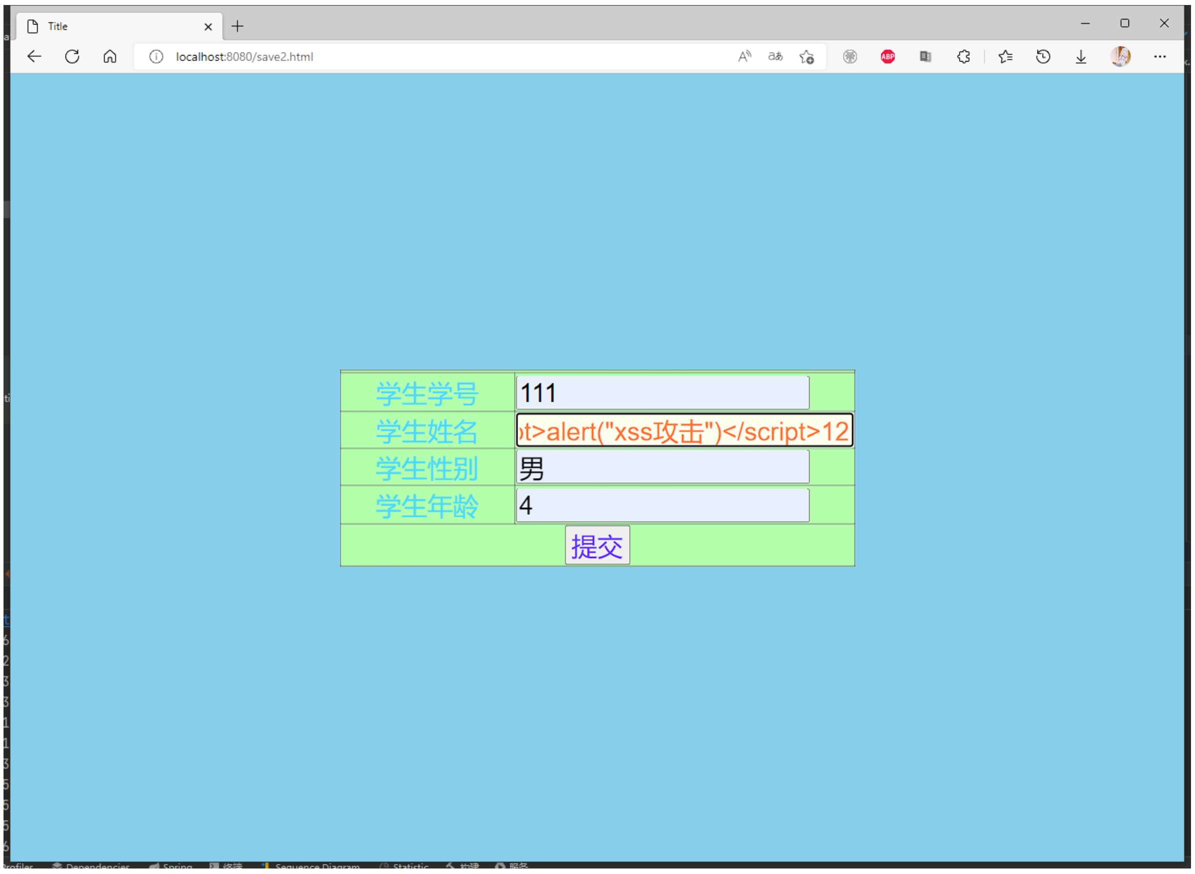
```
<!DOCTYPE html>

<!--
Project name(项目名称)：antiSamy_demo
  File name(文件名): index
  Authors(作者）: mao
  Author QQ：1296193245
  GitHub：https://github.com/maomao124/
  Date(创建日期)：  2022/10/29
  Time(创建时间)：  20:03
  Description(描述)：  无
-->

<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>索引</title>
    <link rel="stylesheet" href="/css/link.css">
    <script src="/js/axios.js"></script>

```

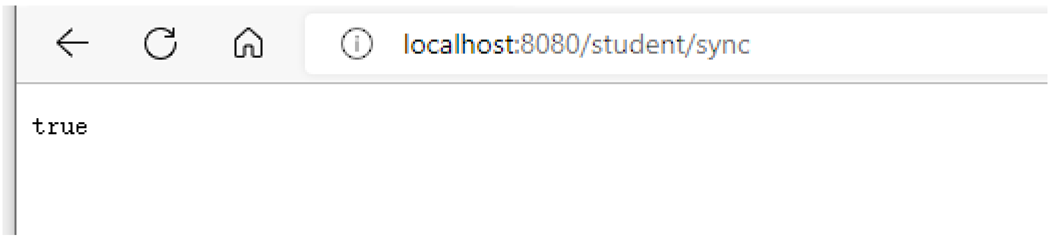
```html

    <style>
        body {
            background-color: skyblue;
        }

        div.a {
            position: absolute;
            top: 50%;
            left: 50%;
            transform: translate(-50%, -50%);
        }
    </style>
</head>
<body>

<div class="a">

    <a href="/show.html">显示所有学生信息</a>
    <a href="/save.html">添加学生信息</a>
    <a href="/save2.html">添加学生信息-同步</a>
    <a href="" onclick="init()">初始化数据</a>

</div>

<script>

    function init()
    {
        //axios发起ajax请求
        axios({
            //请求的方式:
            method: "post",
            //请求的url:
            url: "/student/init",
            //url参数:
            params:
                {},
            //头信息:
            headers:
                {},
            //请求体参数:
            data:
                {},
        }).then(response =>
        {
            console.log(response);
            alert("初始化数据成功");
        }).catch(error =>
        {
            //console.log(error);
            alert("网络异常！");
        })
    }

</script>

</body>
```

```
</html>
```

第十三步：重启并访问

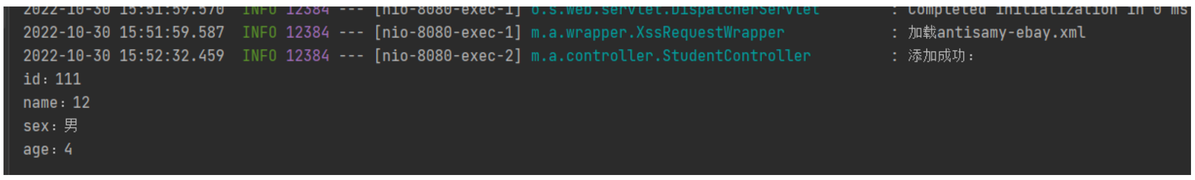






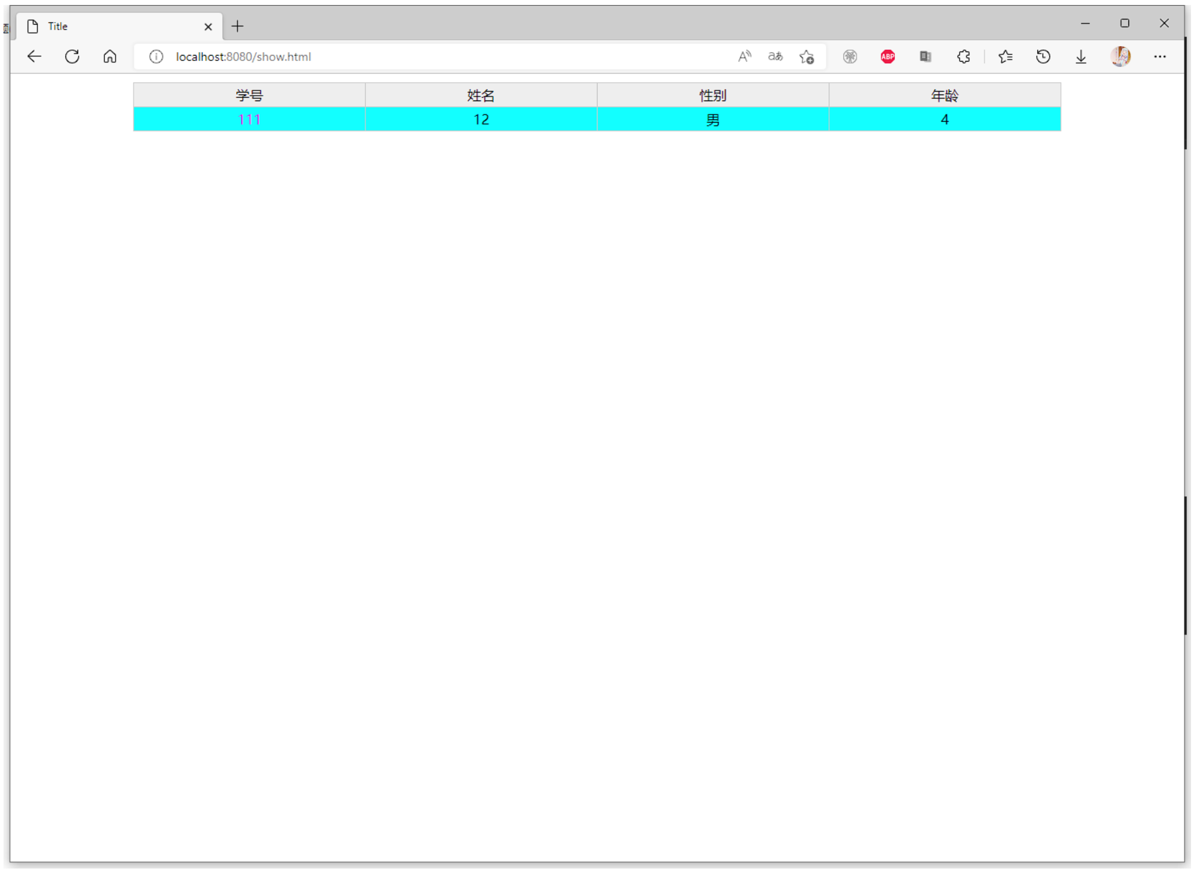
```
true
```

第十四步：查看控制台日志



```
2022-10-30 15:51:59.570  INFO 12384 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 0 ms
2022-10-30 15:51:59.587  INFO 12384 --- [nio-8080-exec-1] m.a.wrapper.XssRequestWrapper            : 加载antisamy-ebay.xml
2022-10-30 15:52:32.459  INFO 12384 --- [nio-8080-exec-2] m.a.controller.StudentController         : 添加成功：
id：111
name：12
sex：男
age：4
```

可以看到表单提交的数据是可以被清理的



学号	姓名	性别	年龄
111	12	男	4

以下是解决方案2

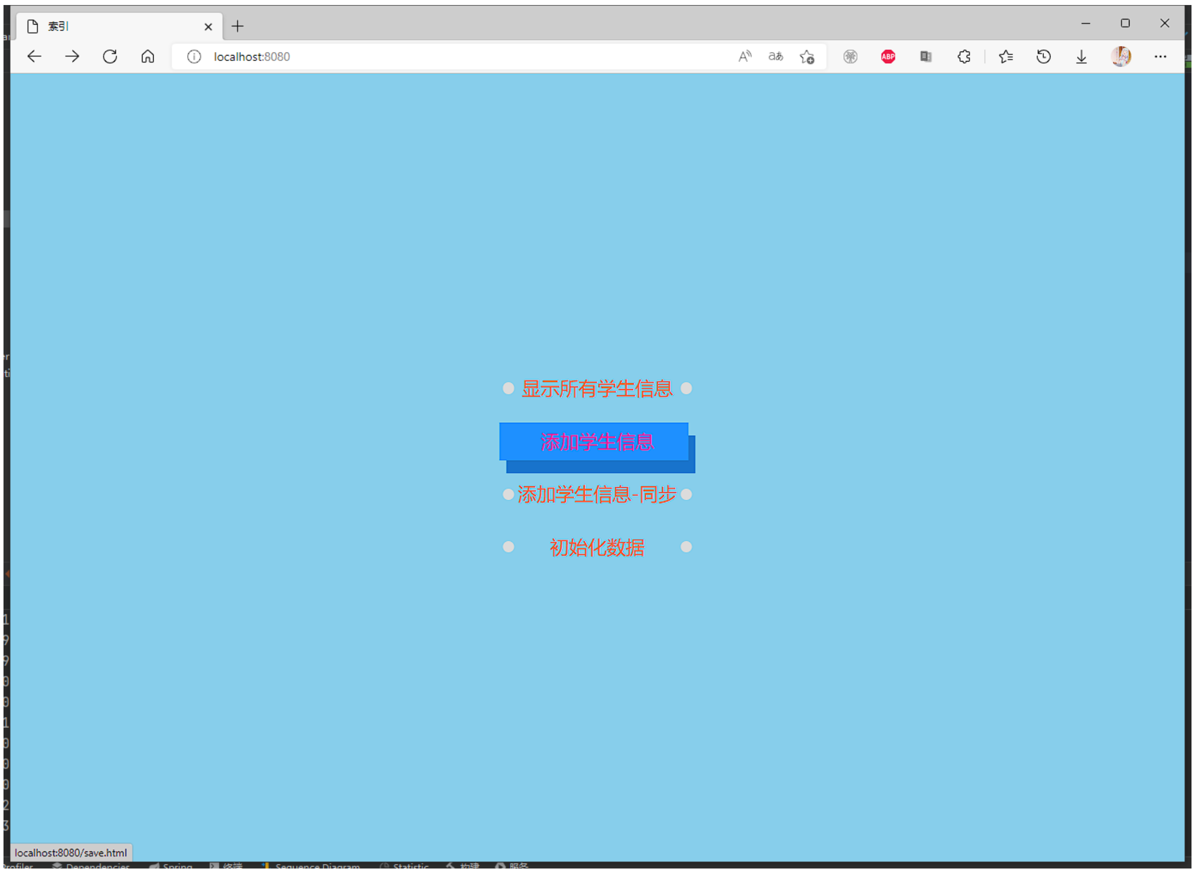
第十五步：编写XssFilterService

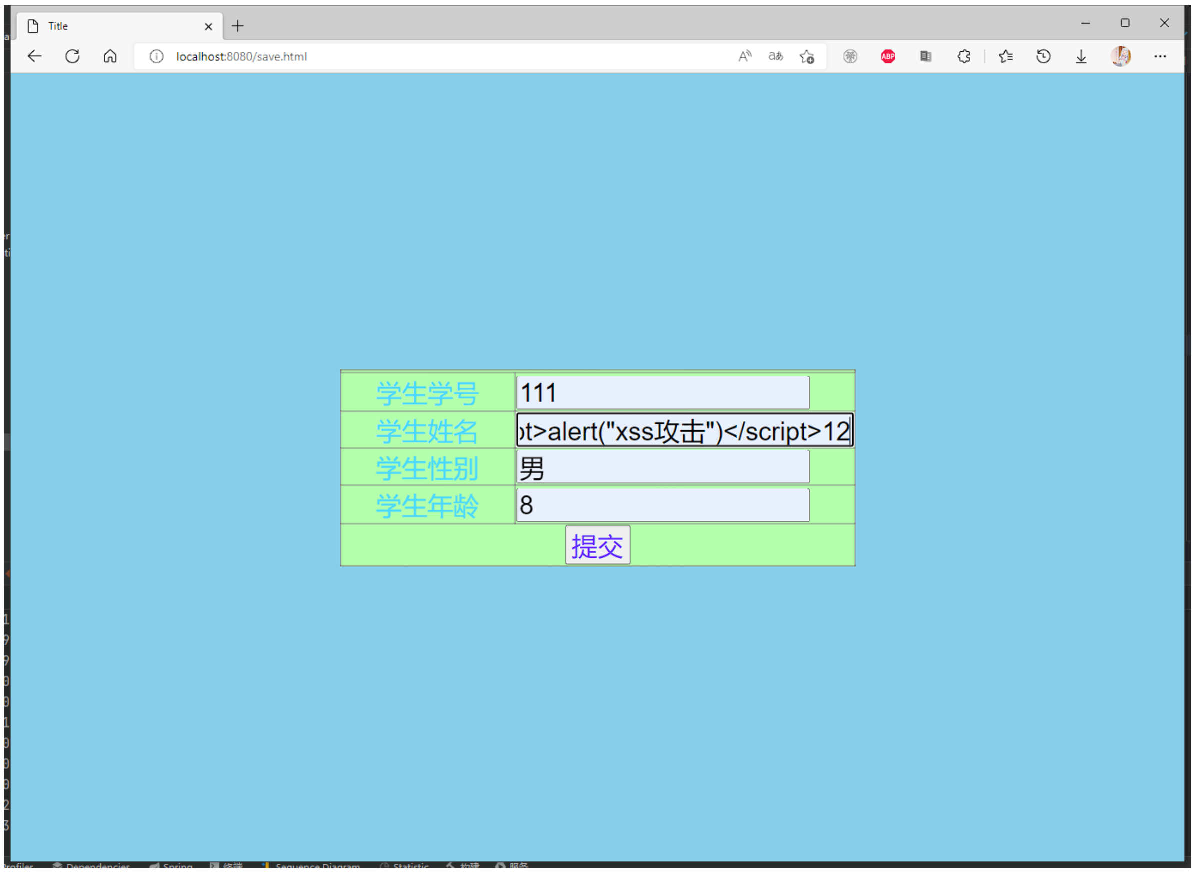
```java
package mao.antisamy_demo.service;

import mao.antisamy_demo.wrapper.XssRequestWrapper;
import org.owasp.validator.html.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Service;

import java.io.InputStream;

/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.service
 * Class(类名): XssFilterService
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)：  2022/10/30
 * Time(创建时间)：  15:43
 * Version(版本): 1.0
 * Description(描述)：  无
 */

@Service
public class XssFilterService
{
    /**
     * 策略文件 需要将要使用的策略文件放到项目资源文件路径下
     */
    @SuppressWarnings("all")
    private static final String antiSamyPath = "antisamy-ebay.xml";

    public static Policy policy = null;

    private static final Logger log =
LoggerFactory.getLogger(XssRequestWrapper.class);

    static
    {
        // 指定策略文件
        try
        {
            InputStream inputStream =
XssRequestWrapper.class.getClassLoader().getResourceAsStream(antiSamyPath);
            assert inputStream != null;
            policy = Policy.getInstance(inputStream);
        }
        catch (PolicyException e)
        {
            e.printStackTrace();
        }
    }

    /**
     * AntiSamy过滤数据
     *
     * @param taintedHTML 需要进行过滤的数据
     * @return 返回过滤后的数据
     */
    public String xssClean(String taintedHTML)
    {
        try
        {
            // 使用AntiSamy进行过滤
            AntiSamy antiSamy = new AntiSamy();
            CleanResults cleanResults = antiSamy.scan(taintedHTML, policy);
            taintedHTML = cleanResults.getCleanHTML();
        }
        catch (ScanException | PolicyException e)
        {
            e.printStackTrace();
        }
        return taintedHTML;
    }
}
```

第十六步：修改StudentController

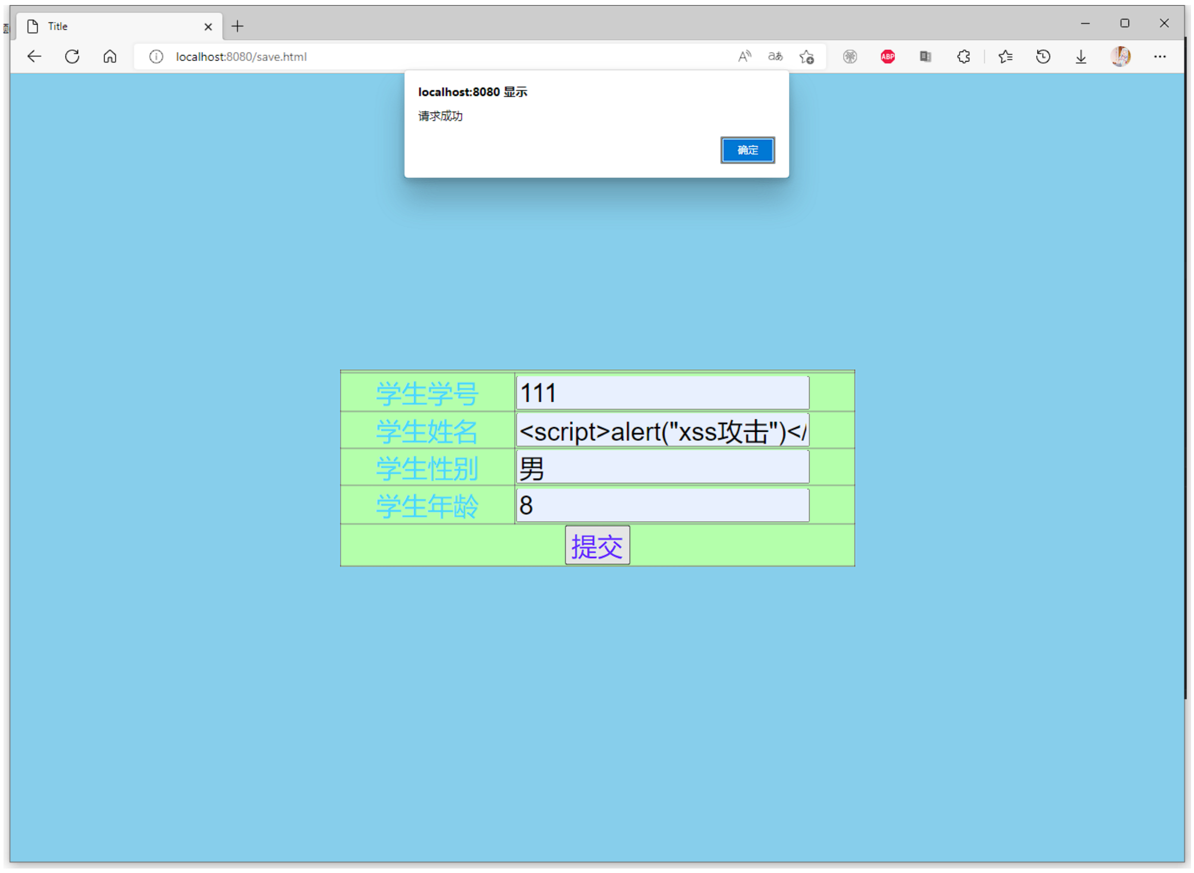
```java
package mao.antisamy_demo.controller;

import mao.antisamy_demo.entity.Student;
import mao.antisamy_demo.service.XssFilterService;
import mao.antisamy_demo.wrapper.XssRequestWrapper;
import org.apache.juli.logging.Log;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.controller
 * Class(类名): StudentController
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/29
 * Time(创建时间)： 20:05
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@RestController
@RequestMapping("/student")
public class StudentController
{
    private static final List<Student> list = 
Collections.synchronizedList(new ArrayList<>());

    private static final Logger log = 
LoggerFactory.getLogger(StudentController.class);

    @Autowired
    private XssFilterService xssFilterService;

    @PostMapping("/init")
    public synchronized void init()
    {
        Student student1 = new Student(10001, "张三", "男", 18);
        Student student2 = new Student(10002, "李四", "女", 16);
        Student student3 = new Student(10003, "王五", "男", 20);
        list.clear();
        list.add(student1);
        list.add(student2);
        list.add(student3);
        log.info("初始化完成");
    }

//    @PostMapping
//    public boolean save(@RequestBody Student student)
//    {
//        list.add(student);
//        log.info("添加成功：\n" + student);
//        return true;
//    }


    @PostMapping
    public boolean save(@RequestBody Student student)
    {
        student.setName(xssFilterService.xssClean(student.getName()));
        student.setSex(xssFilterService.xssClean(student.getSex()));

        list.add(student);
        log.info("添加成功：\n" + student);
        return true;
    }


    @PostMapping("/sync")
    public boolean saveSync(Student student)
    {
        list.add(student);
        log.info("添加成功：\n" + student);
        return true;
    }

    @GetMapping
    public List<Student> getAll()
    {
        log.info("查询所有：\n" + list);
        return list;
    }

}
```

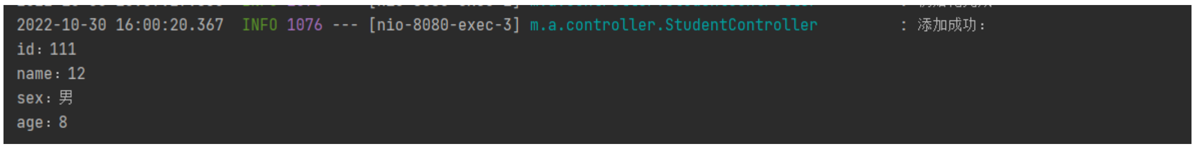
第十七步：重启并访问





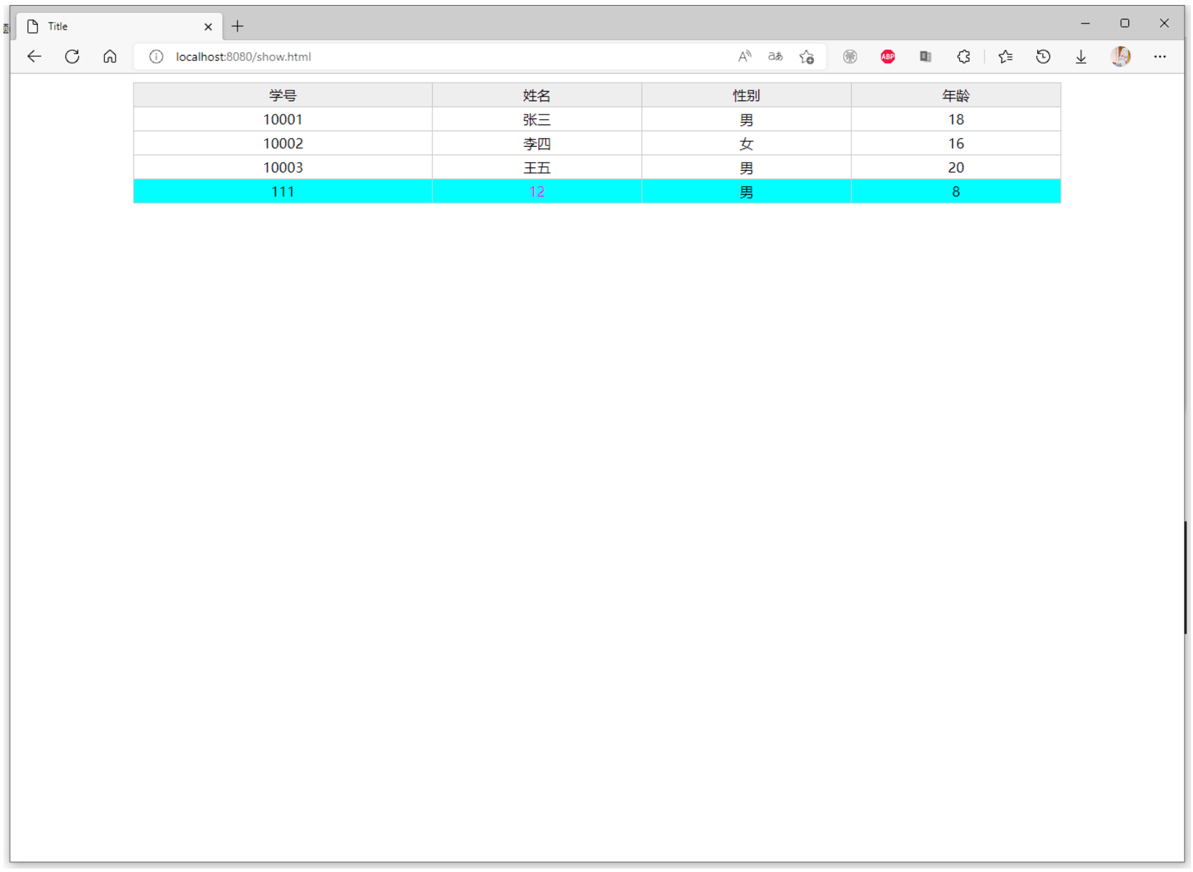
学生学号	111
学生姓名	t>alert("xss攻击")</script>12
学生性别	男
学生年龄	8
提交	





```
2022-10-30 16:00:20.367  INFO 1076 --- [nio-8080-exec-3] m.a.controller.StudentController         : 添加成功：
id：111
name：12
sex：男
age：8
```

可以看到脚本代码被清除了



localhost:8080/show.html

学号	姓名	性别	年龄
10001	张三	男	18
10002	李四	女	16
10003	王五	男	20
111	12	男	8

以下是解决方案3

第十八步：编写类XssStringJsonDeserializer

```java
package mao.antisamy_demo.converter;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.core.JsonToken;
import com.fasterxml.jackson.databind.DeserializationContext;
import com.fasterxml.jackson.databind.JsonDeserializer;
import mao.antisamy_demo.service.XssFilterService;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 * Project name(项目名称)：AntiSamy_spring_boot_starter_demo
 * Package(包名): mao.tools_xss.converter
 * Class(类名): XssStringJsonDeserializer
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/30
 * Time(创建时间)： 20:14
 * Version(版本): 1.0
 * Description(描述)： 过滤跨站脚本的 反序列化工具
 */

public class XssStringJsonDeserializer extends JsonDeserializer<String>
{

    private final XssFilterService xssFilterService;

    public XssStringJsonDeserializer(XssFilterService xssFilterService)
    {
        this.xssFilterService = xssFilterService;
    }

    @Override
    public String deserialize(JsonParser p, DeserializationContext dc)
throws IOException, JsonProcessingException
    {
        if (p.hasToken(JsonToken.VALUE_STRING))
        {
            String value = p.getValueAsString();

            if (value == null || "".equals(value))
            {
                return value;
            }

            List<String> list = new ArrayList<>();
            list.add("<script>");
            list.add("</script>");
            list.add("<iframe>");
            list.add("</iframe>");
            list.add("<noscript>");
            list.add("</noscript>");
            list.add("<frameset>");
            list.add("</frameset>");
            list.add("<frame>");
            list.add("</frame>");
            list.add("<noframes>");
            list.add("</noframes>");
            boolean flag = list.stream().anyMatch(value::contains);
            if (flag)
            {
                return xssFilterService.xssClean(value);
            }
            return value;
        }
        return null;
    }
}
```

第十九步：修改配置类AntiSamyConfig

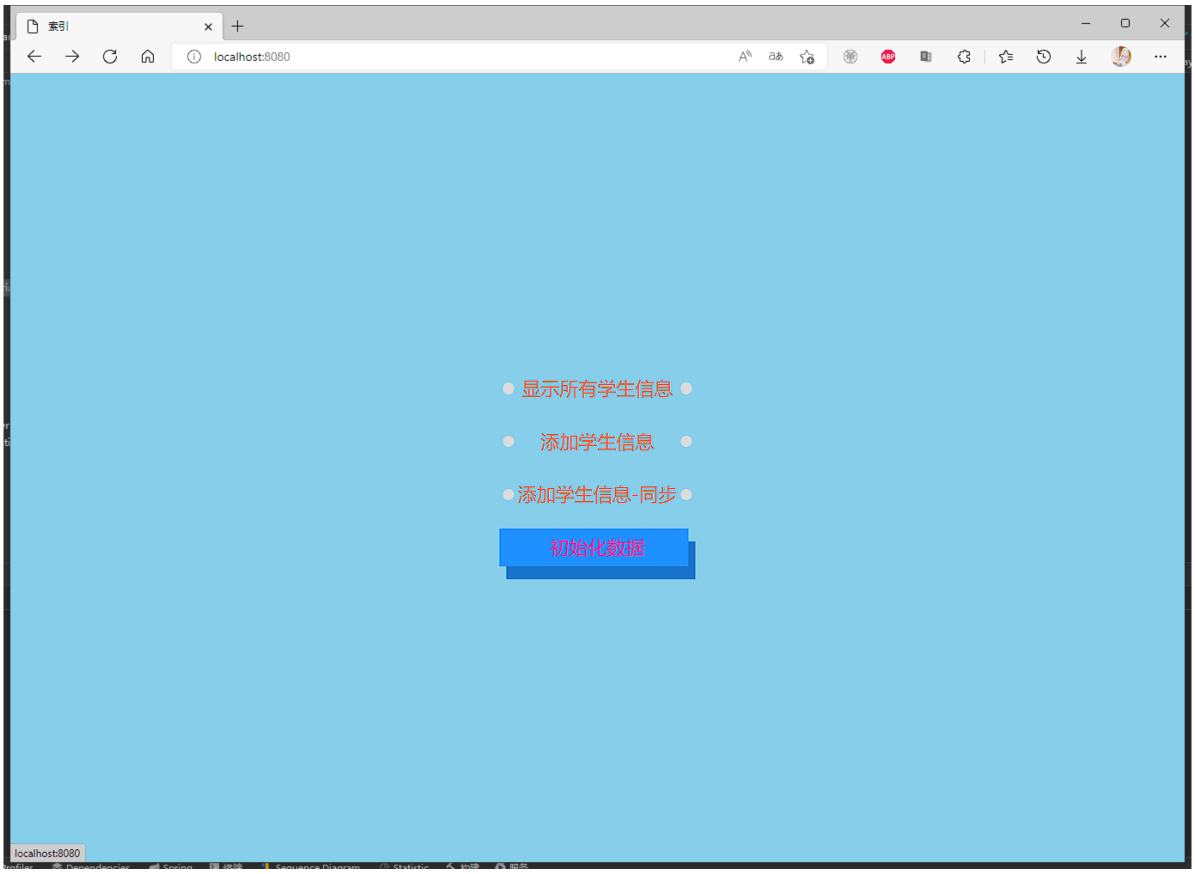
```java
package mao.antisamy_demo.config;

import mao.antisamy_demo.converter.XssStringJsonDeserializer;
import mao.antisamy_demo.filter.XssFilter;
import mao.antisamy_demo.service.XssFilterService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.autoconfigure.jackson.Jackson2ObjectMapperBuilderCustomizer;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import javax.servlet.DispatcherType;

/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.config
 * Class(类名): AntiSamyConfig
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/30
 * Time(创建时间)： 13:53
 * Version(版本): 1.0
 * Description(描述)： 无
 */


@Configuration
public class AntiSamyConfig
{
    @Bean
    public FilterRegistrationBean<XssFilter> filterRegistrationBean()
    {
        FilterRegistrationBean<XssFilter> filterRegistration = new FilterRegistrationBean<>(new XssFilter());
        filterRegistration.addUrlPatterns("/*");
        filterRegistration.setDispatcherTypes(DispatcherType.REQUEST);
        filterRegistration.setName("xssFilter");
        filterRegistration.setOrder(1);

        return filterRegistration;
    }

    /**
     * 配置跨站攻击 反序列化处理器
     *
     * @return Jackson2ObjectMapperBuilderCustomizer
     */
    @Bean
    public Jackson2ObjectMapperBuilderCustomizer
jackson2ObjectMapperBuilderCustomizer2(@Autowired XssFilterService
xssFilterService)
    {
        return builder -> builder.deserializerByType(String.class, new
XssStringJsonDeserializer(xssFilterService));
    }

}
```

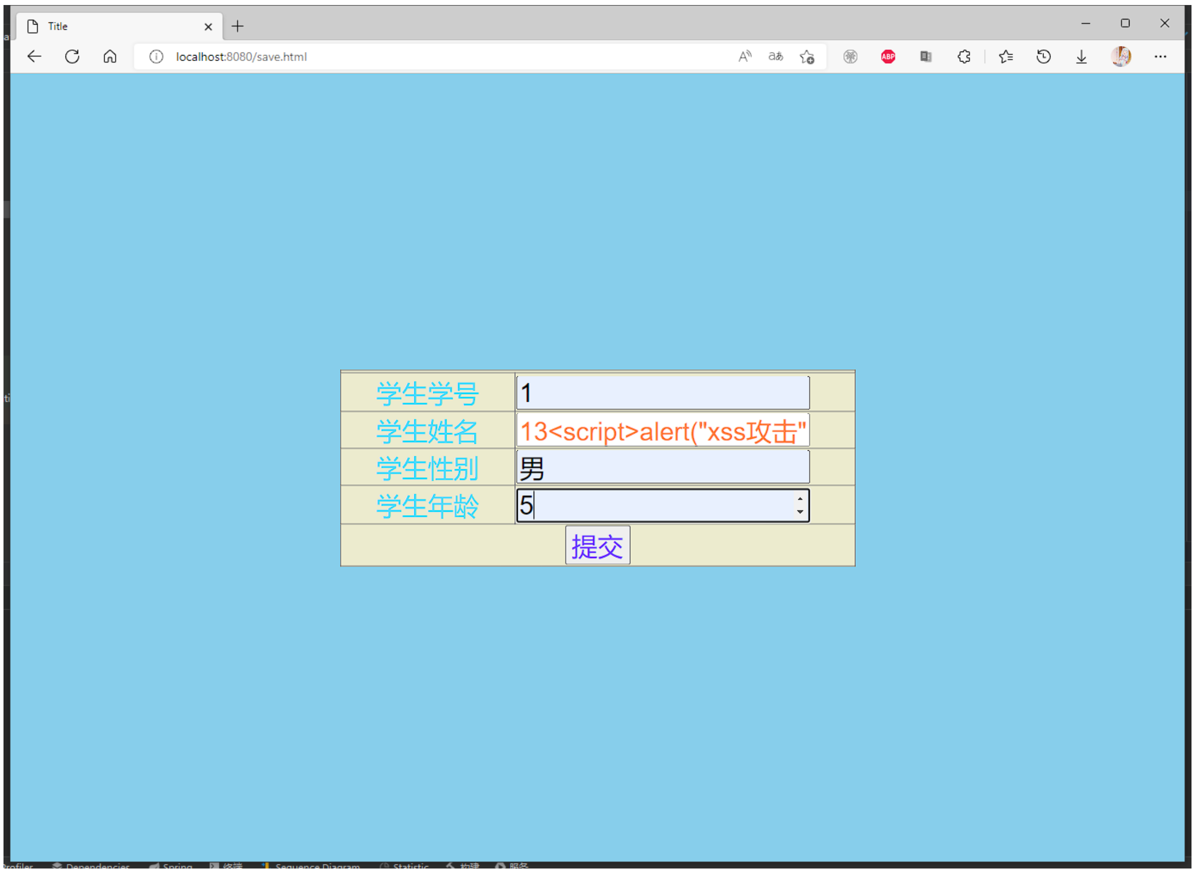
第二十步：修改StudentController

```java
package mao.antisamy_demo.controller;

import mao.antisamy_demo.entity.Student;
import mao.antisamy_demo.service.XssFilterService;
import mao.antisamy_demo.wrapper.XssRequestWrapper;
import org.apache.juli.logging.Log;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;

/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.controller
 * Class(类名): StudentController
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/29
 * Time(创建时间)： 20:05
 * Version(版本): 1.0
 * Description(描述)： 无
 */

@RestController
@RequestMapping("/student")
public class StudentController
{
    private static final List<Student> list =
Collections.synchronizedList(new ArrayList<>());

    private static final Logger log =
LoggerFactory.getLogger(StudentController.class);

```

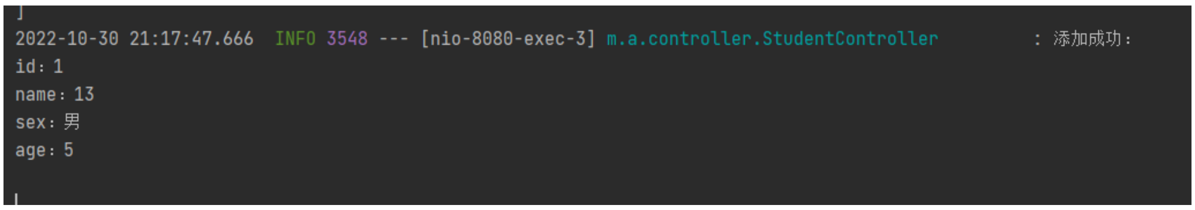
```java
    @Autowired
    private XssFilterService xssFilterService;

    @PostMapping("/init")
    public synchronized void init()
    {
        Student student1 = new Student(10001, "张三", "男", 18);
        Student student2 = new Student(10002, "李四", "女", 16);
        Student student3 = new Student(10003, "王五", "男", 20);
        list.clear();
        list.add(student1);
        list.add(student2);
        list.add(student3);
        log.info("初始化完成");
    }

    @PostMapping
    public boolean save(@RequestBody Student student)
    {
        list.add(student);
        log.info("添加成功：\n" + student);
        return true;
    }


//    @PostMapping
//    public boolean save(@RequestBody Student student)
//    {
//        student.setName(xssFilterService.xssClean(student.getName()));
//        student.setSex(xssFilterService.xssClean(student.getSex()));
//
//        list.add(student);
//        log.info("添加成功：\n" + student);
//        return true;
//    }


    @PostMapping("/sync")
    public boolean saveSync(Student student)
    {
        list.add(student);
        log.info("添加成功：\n" + student);
        return true;
    }

    @GetMapping
    public List<Student> getAll()
    {
        log.info("查询所有：\n" + list);
        return list;
    }

}
```

第二十一步：重启并访问服务





学生学号	1
学生姓名	13<script>alert("xss攻击"
学生性别	男
学生年龄	5
提交	



```
2022-10-30 21:17:47.666  INFO 3548 --- [nio-8080-exec-3] m.a.controller.StudentController         : 添加成功：
id：1
name：13
sex：男
age：5
```

可以看到标签成功地被过滤掉了

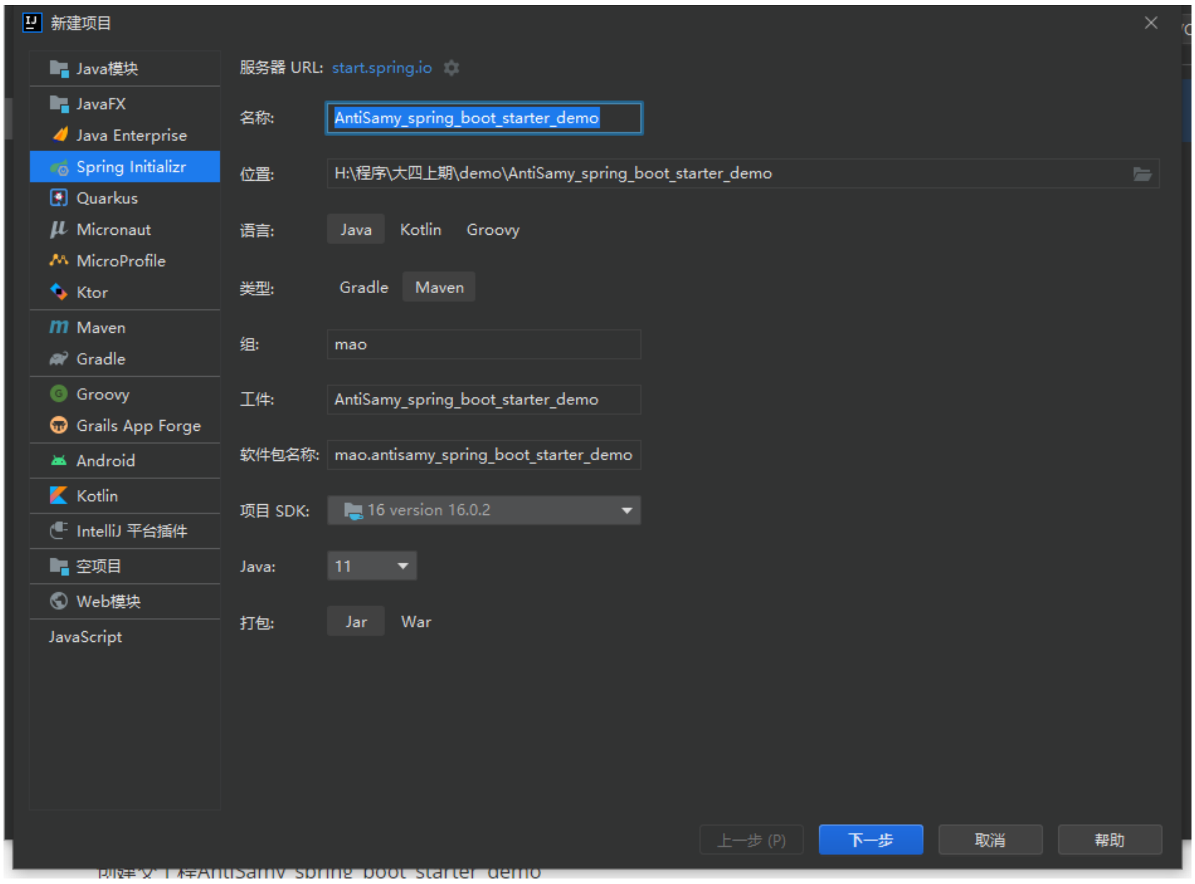
自定义spring boot starter

为了方便使用，需要定义成一个starter，不需要额外进行任何配置就可以使用

开发starter

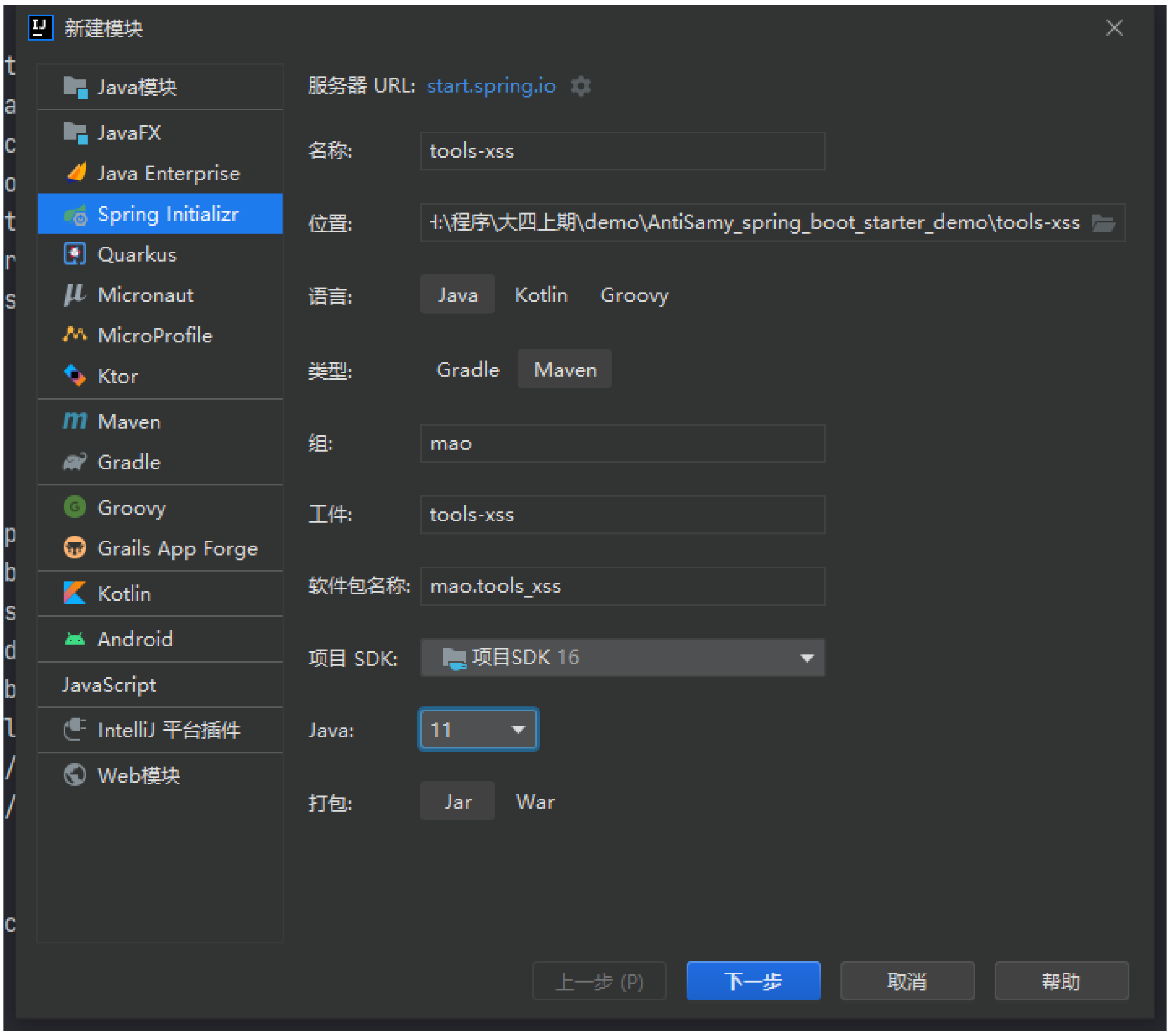
第一步：初始化项目

创建父工程AntiSamy_spring_boot_starter_demo



服务器 URL: start.spring.io
名称: AntiSamy_spring_boot_starter_demo
位置: H:\程序\大四上期\demo\AntiSamy_spring_boot_starter_demo
语言: Java Kotlin Groovy
类型: Gradle Maven
组: mao
工件: AntiSamy_spring_boot_starter_demo
软件包名称: mao.antisamy_spring_boot_starter_demo
项目 SDK: 16 version 16.0.2
Java: 11
打包: Jar War

创建子工程tools-xss



新建模块

服务器 URL: start.spring.io

名称: tools-xss

位置: :\程序\大四上期\demo\AntiSamy_spring_boot_starter_demo\tools-xss

语言: [✓ Java] Kotlin Groovy

类型: Gradle [✓ Maven]

组: mao

工件: tools-xss

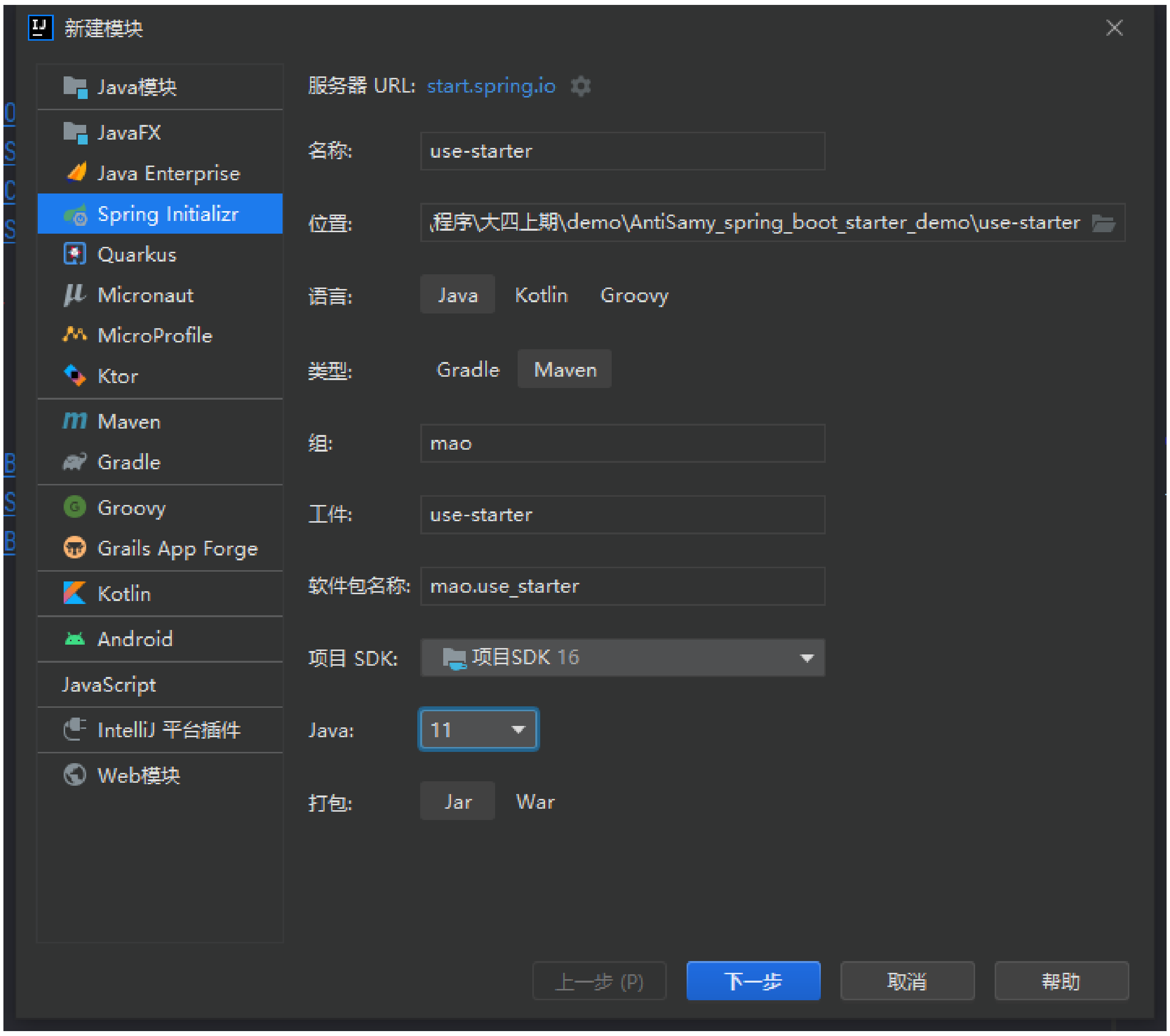
软件包名称: mao.tools_xss

项目 SDK: 项目SDK 16

Java: 11

打包: [✓ Jar] War

创建子工程use-starter



服务器 URL: start.spring.io

名称: use-starter

位置: 程序\大四上期\demo\AntiSamy_spring_boot_starter_demo\use-starter

语言: [✓ Java] Kotlin Groovy

类型: Gradle [✓ Maven]

组: mao

工件: use-starter

软件包名称: mao.use_starter

项目 SDK: 项目SDK 16

Java: 11

打包: [✓ Jar] War

第二步：修改pom文件

父工程AntiSamy_spring_boot_starter_demo的pom文件：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <artifactId>AntiSamy_spring_boot_starter_demo</artifactId>
        <groupId>mao</groupId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>

    <artifactId>use-starter</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>use-starter</name>
    <description>use-starter</description>

    <properties>

    </properties>

    <dependencies>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

子工程tools-xss的pom文件：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <artifactId>AntiSamy_spring_boot_starter_demo</artifactId>
        <groupId>mao</groupId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>

    <artifactId>tools-xss</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>tools-xss</name>
    <description>tools-xss</description>

    <properties>

    </properties>

    <dependencies>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <!--spring boot starter开发依赖-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-autoconfigure</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-configuration-processor</artifactId>
        </dependency>

        <!--解决xss攻击-->
        <dependency>
            <groupId>org.owasp.antisamy</groupId>
            <artifactId>antisamy</artifactId>
            <version>1.5.7</version>
        </dependency>

        <!--阿里巴巴的FastJson json解析-->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.79</version>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
                <configuration>
                    <skip>true</skip>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>

```

子工程use-starter的pom文件：

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <artifactId>AntiSamy_spring_boot_starter_demo</artifactId>
        <groupId>mao</groupId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>

    <artifactId>use-starter</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>use-starter</name>
    <description>use-starter</description>

    <properties>

    </properties>

    <dependencies>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

第三步：编写工具类XssUtils

```java
package mao.tools_xss.utils;


import org.owasp.validator.html.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.io.IOException;
import java.io.InputStream;
import java.util.List;

/**
 * Project name(项目名称)：AntiSamy_spring_boot_starter_demo
 * Package(包名): mao.tools_xss.utils
 * Class(类名): XssUtils
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/30
 * Time(创建时间)： 20:03
 * Version(版本): 1.0
 * Description(描述)： XSS 工具类， 用于过滤特殊字符
 */

public class XssUtils
{
    private static final Logger log = LoggerFactory.getLogger(XssUtils.class);

    private static final String ANTISAMY_SLASHDOT_XML = "antisamy-slashdot-1.4.4.xml";
    private static Policy policy = null;

    static
    {
        log.debug(" start read XSS configfile [" + ANTISAMY_SLASHDOT_XML + "]");
        InputStream inputStream = XssUtils.class.getClassLoader().getResourceAsStream(ANTISAMY_SLASHDOT_XML);
        try
        {
            policy = Policy.getInstance(inputStream);
            log.debug("read XSS configfile [" + ANTISAMY_SLASHDOT_XML + "] success");
        }
        catch (PolicyException e)
        {
            log.error("read XSS configfile [" + ANTISAMY_SLASHDOT_XML + "] fail , reason:", e);
        }
        finally
        {
            if (inputStream != null)
            {
                try
                {
                    inputStream.close();
                }
                catch (IOException e)
                {
                    log.error("close XSS configfile [" + ANTISAMY_SLASHDOT_XML + "] fail , reason:", e);
                }
            }
        }
    }

    /**
     * 跨站攻击语句过滤 方法
     *
     * @param paramValue           待过滤的参数
     * @param ignoreParamValueList 忽略过滤的参数列表
     * @return String
     */
    public static String xssClean(String paramValue, List<String> ignoreParamValueList)
    {
        AntiSamy antiSamy = new AntiSamy();
        try
        {
            log.debug("raw value before xssClean: " + paramValue);
            if (isIgnoreParamValue(paramValue, ignoreParamValueList))
            {
                log.debug("ignore the xssClean,keep the raw paramValue: " + paramValue);
                return paramValue;
            }
            else
            {
                final CleanResults cleanResults = antiSamy.scan(paramValue, policy);
                cleanResults.getErrorMessages().forEach(log::debug);
                String str = cleanResults.getCleanHTML();
                /*String str = StringEscapeUtils.escapeHtml(cleanResults.getCleanHTML());
                str = str.replaceAll((antiSamy.scan("&nbsp;", policy)).getCleanHTML(), "");
                str = StringEscapeUtils.unescapeHtml(str);*/
                /*str = str.replaceAll("&quot;", "\"");
                str = str.replaceAll("&amp;", "&");
                str = str.replaceAll("'", "'");
                str = str.replaceAll("'", "' ");

                str = str.replaceAll("&lt;", "<");
                str = str.replaceAll("&gt;", ">");*/
                log.debug("xssfilter value after xssClean" + str);

                return str;
            }
        }
        catch (ScanException e)
        {
            log.error("scan failed armter is [" + paramValue + "]", e);
        }
        catch (PolicyException e)
        {
            log.error("antisamy convert failed  armter is [" + paramValue + "]", e);
        }
        return paramValue;
    }

    /**
     * 忽略参数值
     *
     * @param paramValue           参数值
     * @param ignoreParamValueList 忽略参数值列表
     * @return boolean
     */
    private static boolean isIgnoreParamValue(String paramValue, List<String> ignoreParamValueList)
    {
        if (paramValue == null || paramValue.equals(""))
        {
            return true;
        }
        if (ignoreParamValueList == null || ignoreParamValueList.size() == 0)
        {
            return false;
        }
        else
        {
            for (String ignoreParamValue : ignoreParamValueList)
            {
                if (paramValue.contains(ignoreParamValue))
                {
                    return true;
                }
            }
        }
        return false;
    }
}

```

第四步：编写类XssRequestWrapper

```java
package mao.tools_xss.wrapper;

import mao.tools_xss.utils.XssUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletRequestWrapper;
import java.util.List;
import java.util.Map;

/**
 * Project name(项目名称): AntiSamy_spring_boot_starter_demo
 * Package(包名): mao.tools_xss.wrapper
 * Class(类名): XssRequestWrapper
 * Author(作者）: mao
 * Author QQ: 1296193245
 * GitHub: https://github.com/maomao124/
 * Date(创建日期):  2022/10/30
 * Time(创建时间):  20:01
 * Version(版本): 1.0
 * Description(描述):  无
 */

public class XssRequestWrapper extends HttpServletRequestWrapper
{

    private static final Logger log = 
LoggerFactory.getLogger(XssRequestWrapper.class);

    private final List<String> ignoreParamValueList;

    public XssRequestWrapper(HttpServletRequest request, List<String> 
ignoreParamValueList)
    {
        super(request);
        this.ignoreParamValueList = ignoreParamValueList;
    }

    @Override
    public Map<String, String[]> getParameterMap()
    {
        Map<String, String[]> requestMap = super.getParameterMap();
        for (Map.Entry<String, String[]> me : requestMap.entrySet())
        {
            log.debug(me.getKey() + ":");
            String[] values = me.getValue();
            for (int i = 0; i < values.length; i++)
            {
                log.debug(values[i]);
                values[i] = XssUtils.xssClean(values[i], 
this.ignoreParamValueList);
            }
        }
        return requestMap;
    }

    @Override
    public String[] getParameterValues(String paramString)
    {
        String[] arrayOfString1 = super.getParameterValues(paramString);
        if (arrayOfString1 == null)
        {
            return null;
        }
        int i = arrayOfString1.length;
        String[] arrayOfString2 = new String[i];
        for (int j = 0; j < i; j++)
        {
            arrayOfString2[j] = XssUtils.xssClean(arrayOfString1[j], this.ignoreParamValueList);
        }
        return arrayOfString2;
    }

    @Override
    public String getParameter(String paramString)
    {
        String str = super.getParameter(paramString);
        if (str == null)
        {
            return null;
        }
        return XssUtils.xssClean(str, this.ignoreParamValueList);
    }

    @Override
    public String getHeader(String paramString)
    {
        String str = super.getHeader(paramString);
        if (str == null)
        {
            return null;
        }
        return XssUtils.xssClean(str, this.ignoreParamValueList);
    }
}
```

第五步：编写类XssStringJsonDeserializer

```java
package mao.tools_xss.converter;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.core.JsonProcessingException;
import com.fasterxml.jackson.core.JsonToken;
import com.fasterxml.jackson.databind.DeserializationContext;
import com.fasterxml.jackson.databind.JsonDeserializer;

import com.fasterxml.jackson.databind.JsonDeserializer;
import mao.tools_xss.utils.XssUtils;

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

/**
 * Project name(项目名称)：AntiSamy_spring_boot_starter_demo
 * Package(包名): mao.tools_xss.converter
 * Class(类名): XssStringJsonDeserializer
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/30
 * Time(创建时间)： 20:14
 * Version(版本): 1.0
 * Description(描述)： 过滤跨站脚本的 反序列化工具
 */

public class XssStringJsonDeserializer extends JsonDeserializer<String>
{

    private static final List<String> list = new ArrayList<>();

    static
    {
        list.add("<script>");
        list.add("</script>");
        list.add("<iframe>");
        list.add("</iframe>");
        list.add("<noscript>");
        list.add("</noscript>");
        list.add("<frameset>");
        list.add("</frameset>");
        list.add("<frame>");
        list.add("</frame>");
        list.add("<noframes>");
        list.add("</noframes>");
        list.add("<h1>");
        list.add("</h1>");
        list.add("<h2>");
        list.add("</h2>");
        list.add("<h3>");
        list.add("</h3>");
        list.add("<h4>");
        list.add("</h4>");
        list.add("<h5>");
        list.add("</h5>");
        list.add("<h6>");
        list.add("</h6>");
        list.add("<img>");
        list.add("</img>");
        list.add("<table>");
        list.add("</table>");
        list.add("<form>");
        list.add("</form>");
    }


    @Override
    public String deserialize(JsonParser p, DeserializationContext dc) throws IOException, JsonProcessingException
    {
        if (p.hasToken(JsonToken.VALUE_STRING))
        {
            String value = p.getValueAsString();

            if (value == null || "".equals(value))
            {
                return value;
            }

            boolean flag = list.stream().anyMatch(value::contains);
            if (flag)
            {
                return XssUtils.xssClean(value, null);
            }
            return value;
        }
        return null;
    }
}

```

第六步：编写过滤器类XssFilter

```java
package mao.tools_xss.filter;

import com.alibaba.fastjson.JSON;
import mao.tools_xss.wrapper.XssRequestWrapper;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

/**
 * Project name(项目名称): AntiSamy_spring_boot_starter_demo
 * Package(包名): mao.tools_xss.filter
 * Class(类名): XssFilter
 * Author(作者) : mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/30
 * Time(创建时间)： 20:18
 * Version(版本): 1.0
 * Description(描述)： 无
 */

public class XssFilter implements Filter
{

    private static final Logger log = LoggerFactory.getLogger(XssFilter.class);

    /**
     * 可放行的请求路径
     */
    private static final String IGNORE_PATH = "ignorePath";
    /**
     * 可放行的参数值
     */
    private static final String IGNORE_PARAM_VALUE = "ignoreParamValue";
    /**
     * 默认放行单点登录的登出响应(响应中包含samlp:LogoutRequest标签，直接放行)
     */
    private static final String CAS_LOGOUT_RESPONSE_TAG = "samlp:LogoutRequest";
    /**
     * 可放行的请求路径列表
     */
    private List<String> ignorePathList;
    /**
     * 可放行的参数值列表
     */
    private List<String> ignoreParamValueList;

    @Override
    public void init(FilterConfig filterConfig) throws ServletException
    {
        log.debug("XSS fiter [XSSFilter] init start ...");
        String ignorePaths = filterConfig.getInitParameter(IGNORE_PATH);
        String ignoreParamValues = filterConfig.getInitParameter(IGNORE_PARAM_VALUE);
        if (!(ignorePaths == null || ignorePaths.equals("")))
        {
            String[] ignorePathArr = ignorePaths.split(",");
            ignorePathList = Arrays.asList(ignorePathArr);
        }
        if (!(ignoreParamValues == null || ignoreParamValues.equals("")))
        {
            String[] ignoreParamValueArr = ignoreParamValues.split(",");
            ignoreParamValueList = Arrays.asList(ignoreParamValueArr);
            //默认放行单点登录的登出响应(响应中包含samlp:LogoutRequest标签，直接放行)
            if (!ignoreParamValueList.contains(CAS_LOGOUT_RESPONSE_TAG))
            {
                ignoreParamValueList.add(CAS_LOGOUT_RESPONSE_TAG);
            }
        }
        else
        {
            //默认放行单点登录的登出响应(响应中包含samlp:LogoutRequest标签，直接放行)
            ignoreParamValueList = new ArrayList<>();
            ignoreParamValueList.add(CAS_LOGOUT_RESPONSE_TAG);
        }
        log.debug("ignorePathList=" + JSON.toJSONString(ignorePathList));
        log.debug("ignoreParamValueList=" + JSON.toJSONString(ignoreParamValueList));
        log.debug("XSS fiter [XSSFilter] init end");
    }

    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
            throws IOException, ServletException
    {
        log.debug("XSS fiter [XSSFilter] starting");
        // 判断uri是否包含项目名称
        String uriPath = ((HttpServletRequest) request).getRequestURI();
        if (isIgnorePath(uriPath))
        {
            log.debug("ignore xssfilter,path[" + uriPath + "] pass through XssFilter, go ahead...");
            chain.doFilter(request, response);
            return;
        }
        else
        {
            log.debug("has xssfiter path[" + uriPath + "] need XssFilter, go to XssRequestWrapper");
            //传入重写后的Request
            chain.doFilter(new XssRequestWrapper((HttpServletRequest) request, ignoreParamValueList), response);
        }
        log.debug("XSS fiter [XSSFilter] stop");
    }

    @Override
    public void destroy()
    {
        log.debug("XSS fiter [XSSFilter] destroy");
    }

    /**
     * 是否为忽略的请求路径
     *
     * @param servletPath servlet路径
     * @return boolean
     */
    private boolean isIgnorePath(String servletPath)
    {
        if (servletPath == null || servletPath.equals(""))
        {
            return true;
        }
        if (ignorePathList == null || ignorePathList.size() == 0)
        {
            return false;
        }
        else
        {
            for (String ignorePath : ignorePathList)
            {
                if (!(ignorePath == null || ignorePath.equals("")) &&
servletPath.contains(ignorePath.trim()))
                {
                    return true;
                }
            }
        }
        return false;
    }
}
```

第七步：拷贝之前的service包到此项目中，并更改

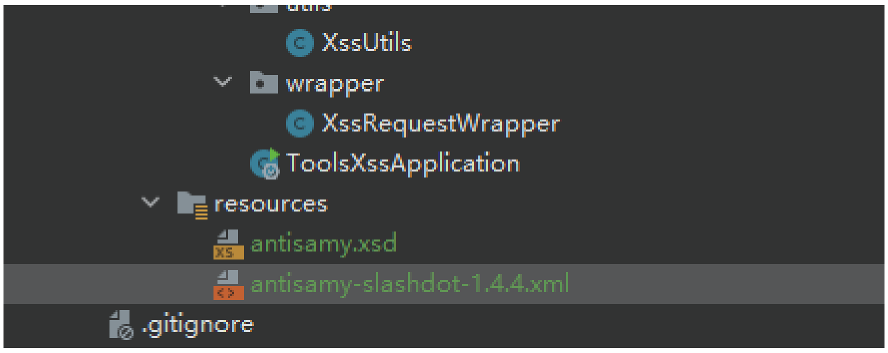
```java
package mao.tools_xss.service;

import org.owasp.validator.html.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;


import java.io.InputStream;

/**
 * Project name(项目名称)：antiSamy_demo
 * Package(包名): mao.antisamy_demo.service
 * Class(类名): XssFilterService
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/30
 * Time(创建时间)： 15:43
 * Version(版本): 1.0
 * Description(描述)： 无
 */

public class XssFilterService
{
    /**
     * 策略文件 需要将要使用的策略文件放到项目资源文件路径下
     */
    @SuppressWarnings("all")
    private static final String antiSamyPath = "antisamy-slashdot-
1.4.4.xml";

```

```java

    public static Policy policy = null;

    private static final Logger log = 
LoggerFactory.getLogger(XssFilterService.class);

    static
    {
        // 指定策略文件
        try
        {
            InputStream inputStream = 
XssFilterService.class.getClassLoader().getResourceAsStream(antiSamyPath);
            assert inputStream != null;
            policy = Policy.getInstance(inputStream);
        }
        catch (PolicyException e)
        {
            e.printStackTrace();
        }
    }

    /**
     * AntiSamy过滤数据
     *
     * @param taintedHTML 需要进行过滤的数据
     * @return 返回过滤后的数据
     */
    public String xssClean(String taintedHTML)
    {
        try
        {
            // 使用AntiSamy进行过滤
            AntiSamy antiSamy = new AntiSamy();
            CleanResults cleanResults = antiSamy.scan(taintedHTML, policy);
            taintedHTML = cleanResults.getCleanHTML();
        }
        catch (ScanException | PolicyException e)
        {
            e.printStackTrace();
        }
        return taintedHTML;
    }
}

```

第八步：编写配置类XssAuthConfiguration

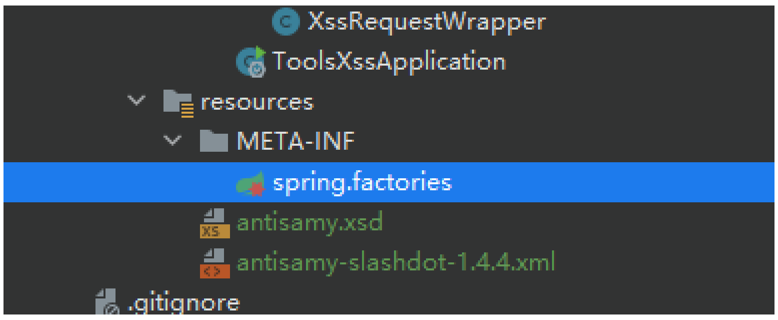
```java
package mao.tools_xss.config;

import mao.tools_xss.converter.XssStringJsonDeserializer;
import mao.tools_xss.filter.XssFilter;
import mao.tools_xss.service.XssFilterService;
import org.springframework.boot.autoconfigure.jackson.Jackson2ObjectMapperBuilderCustomizer;
import org.springframework.boot.web.servlet.FilterRegistrationBean;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import java.util.HashMap;
import java.util.Map;
import java.util.StringJoiner;

/**
 * Project name(项目名称)：AntiSamy_spring_boot_starter_demo
 * Package(包名): mao.tools_xss.config
 * Class(类名): XssAuthConfiguration
 * Author(作者）: mao
 * Author QQ：1296193245
 * GitHub：https://github.com/maomao124/
 * Date(创建日期)： 2022/10/30
 * Time(创建时间)： 20:31
 * Version(版本): 1.0
 * Description(描述)： XSS 跨站攻击自动配置
 */

@Configuration
public class XssAuthConfiguration
{
    /**
     * 配置跨站攻击 反序列化处理器
     *
     * @return Jackson2ObjectMapperBuilderCustomizer
     */
    @Bean
    public Jackson2ObjectMapperBuilderCustomizer jackson2ObjectMapperBuilderCustomizer2()
    {
        return builder -> builder.deserializerByType(String.class, new XssStringJsonDeserializer());
    }


    /**
     * 配置跨站攻击过滤器
     *
     * @return FilterRegistrationBean
     */
    @Bean
    public FilterRegistrationBean<XssFilter> filterRegistrationBean()
    {
        //可以拓展

        FilterRegistrationBean<XssFilter> filterRegistration = new FilterRegistrationBean<>(new XssFilter());
        filterRegistration.addUrlPatterns("/*");
        filterRegistration.setOrder(1);

        Map<String, String> initParameters = new HashMap<>(2);
        String excludes = new StringJoiner(",")
                .add("/favicon.ico")
                .add("/doc.html")
                .add("/swagger-ui.html")
                .add("/csrf")
                .add("/webjars/*")
                .add("/v2/*")
                .add("/swagger-resources/*")
                .add("/resources/*")
                .add("/static/*")
                .add("/public/*")
                .add("/classpath:*")
                .add("/actuator/*")
                .toString();
        initParameters.put("excludes", excludes);
        initParameters.put("isIncludeRichText", "true");
        filterRegistration.setInitParameters(initParameters);
        return filterRegistration;
    }

    @Bean
    public XssFilterService xssFilterService()
    {
        return new XssFilterService();
    }
}

```

第九步：拷贝antisamy-slashdot-1.4.4.xml文件和antisamy.xsd文件到资源目录中



这两个文件在这里没有什么用，就是为了复制

第十步：编写spring.factories文件



```
org.springframework.boot.autoconfigure.EnableAutoConfiguration=\
    mao.tools_xss.config.XssAuthConfiguration
```

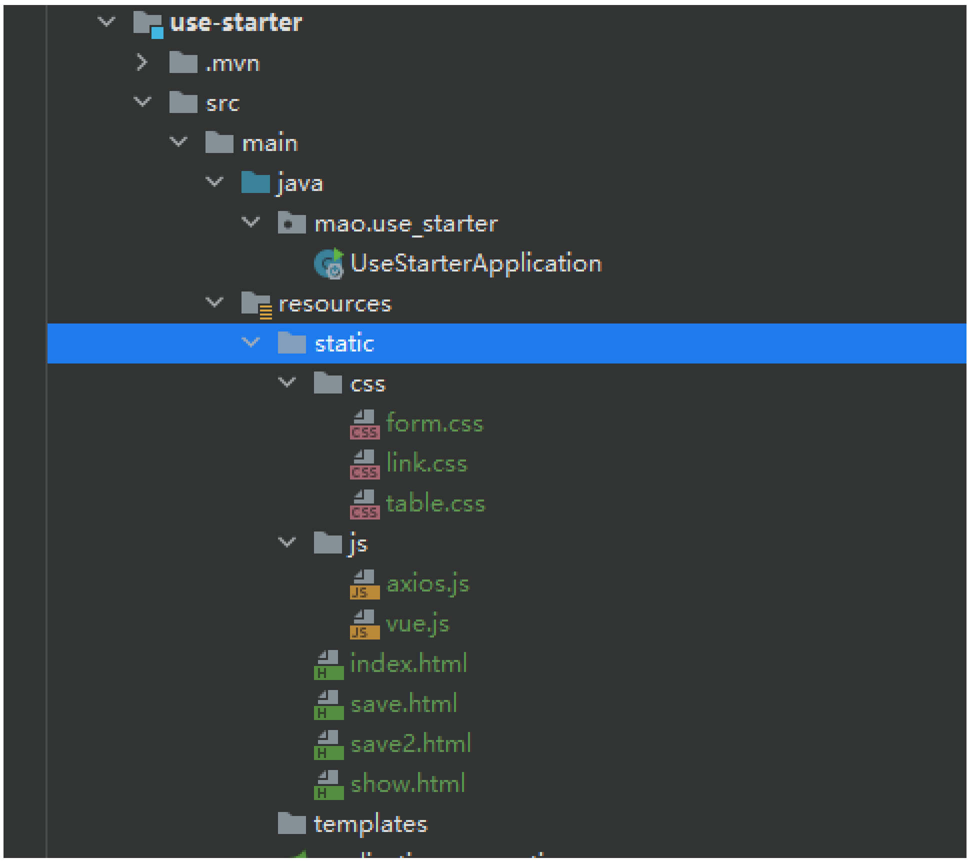
使用starter

第一步：导入tools-xss的依赖

```
<?xml version="1.0" encoding="UTF-8"?>
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <artifactId>AntiSamy_spring_boot_starter_demo</artifactId>
        <groupId>mao</groupId>
        <version>0.0.1-SNAPSHOT</version>
    </parent>

```

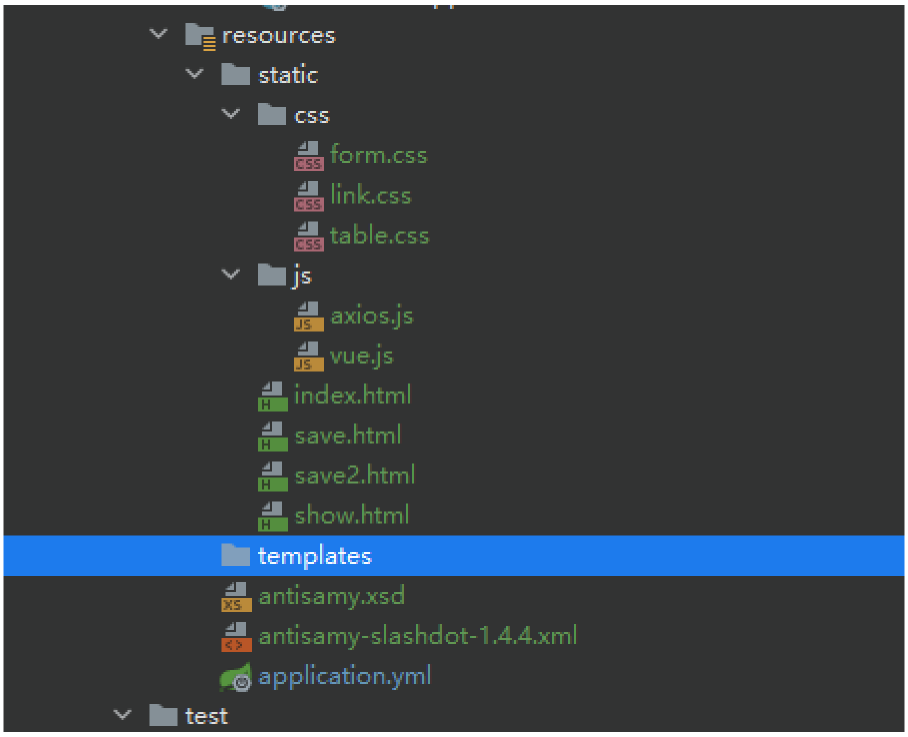
```xml
    <artifactId>use-starter</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>use-starter</name>
    <description>use-starter</description>

    <properties>

    </properties>

    <dependencies>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>mao</groupId>
            <artifactId>tools-xss</artifactId>
            <version>0.0.1-SNAPSHOT</version>
        </dependency>

    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```

第二步：导入之前项目的静态资源文件



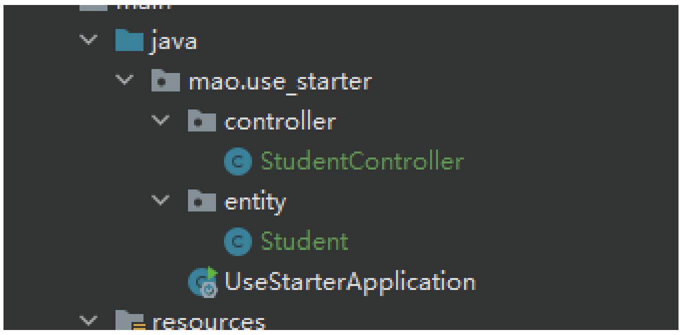
```
use-starter
  .mvn
  src
    main
      java
        mao.use_starter
          UseStarterApplication
      resources
        static
          css
            form.css
            link.css
            table.css
          js
            axios.js
            vue.js
          index.html
          save.html
          save2.html
          show.html
        templates
```

第三步：拷贝antisamy-slashdot-1.4.4.xml文件和antisamy.xsd文件到资源目录中



```
      resources
        static
          css
            form.css
            link.css
            table.css
          js
            axios.js
            vue.js
          index.html
          save.html
          save2.html
          show.html
        templates
        antisamy.xsd
        antisamy-slashdot-1.4.4.xml
        application.yml
    test
```

第四步：拷贝之前的controller和entity包到此项目中



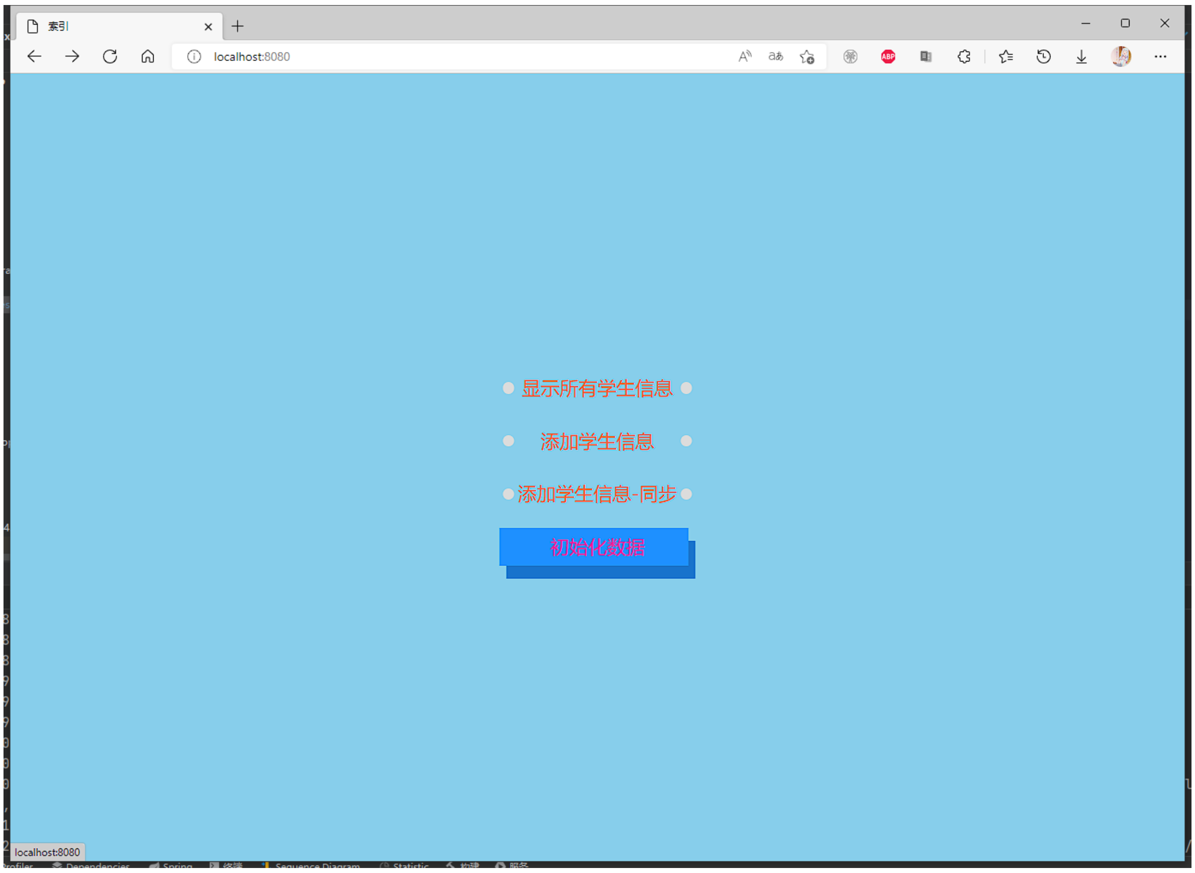
第五步：修改配置文件application.yml

```
  # 开启debug模式，输出调试信息，常用于检查系统运行状况
  #debug: true

  # 设置日志级别，root表示根节点，即整体应用日志级别
logging:
 # 日志输出到文件的文件名
  file:
     name: server.log
  # 字符集
  charset:
    file: UTF-8
  # 分文件
  logback:
    rollingpolicy:
      #最大文件大小
      max-file-size: 16KB
      # 文件格式
      file-name-pattern: logs/server_log/%d{yyyy/MM月/dd日/}%i.log
  # 设置日志组
  group:
  # 自定义组名，设置当前组中所包含的包
    mao_pro: mao
  level:
    root: info
    # 为对应组设置日志级别
    mao_pro: debug
    # 日志输出格式
# pattern:
  # console: "%d %clr(%p) --- [%16t] %clr(%-40.40c){cyan} : %m %n"
```

第六步：启动程序

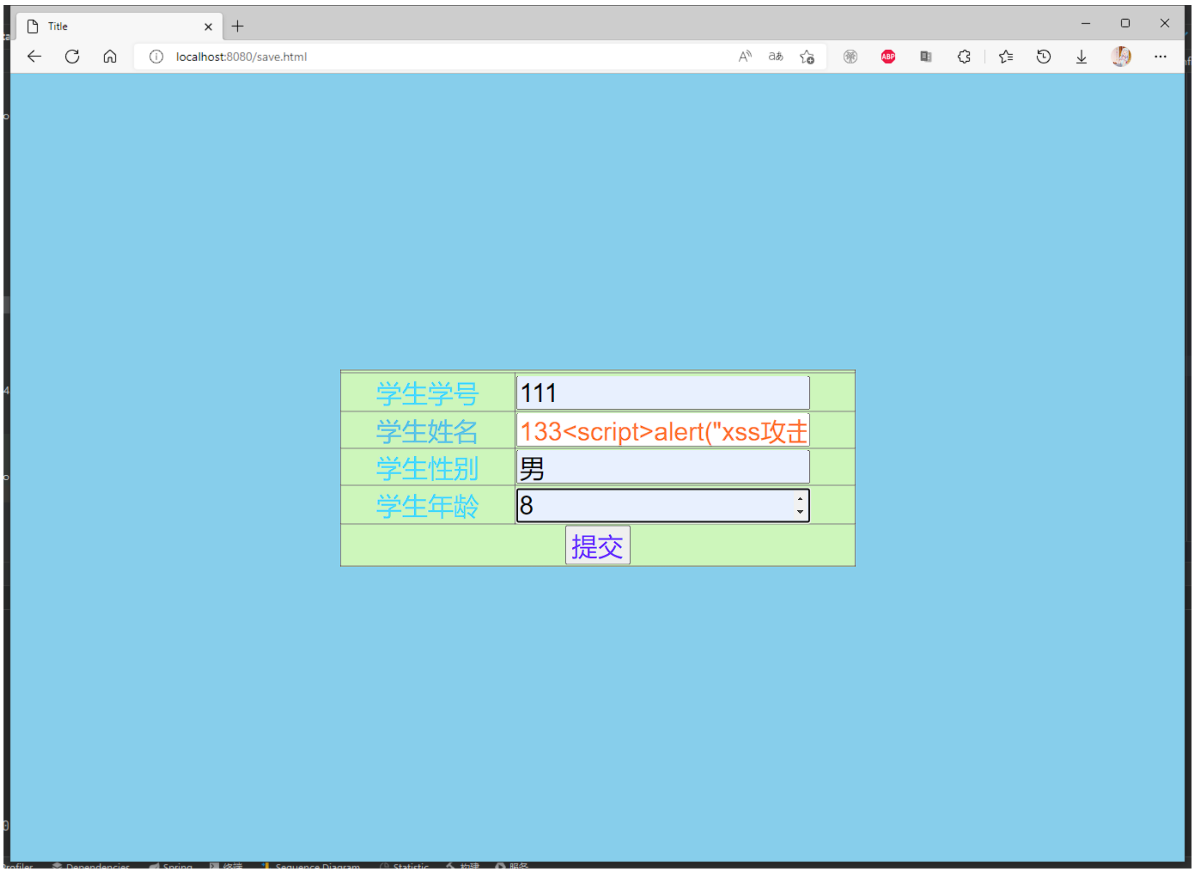
```

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::                (v2.7.1)

2022-10-30 21:39:24.175  INFO 18896 --- [           main] mao.use_starter.UseStarterApplication    : Starting UseStarterApplication using Java 16.0.2 on mao with PID 18896 (H:\程序\大四上期\AntiSamy_spring_boot_starter_demo\use-starter\target\classes started by mao in H:\程序\大四上期\AntiSamy_spring_boot_starter_demo)
2022-10-30 21:39:24.177 DEBUG 18896 --- [           main] mao.use_starter.UseStarterApplication    : Running with Spring Boot v2.7.1, Spring v5.3.21
2022-10-30 21:39:24.178  INFO 18896 --- [           main] mao.use_starter.UseStarterApplication    : No active profile set, falling back to 1 default profile: "default"
2022-10-30 21:39:24.830  INFO 18896 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
2022-10-30 21:39:24.842  INFO 18896 --- [           main] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2022-10-30 21:39:24.842  INFO 18896 --- [           main] org.apache.catalina.core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.64]
2022-10-30 21:39:24.924  INFO 18896 --- [           main] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplicationContext
2022-10-30 21:39:24.925  INFO 18896 --- [           main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 709 ms
2022-10-30 21:39:24.956 DEBUG 18896 --- [           main] mao.tools_xss.filter.XssFilter           : XSS fiter [XSSFilter] init start ...
2022-10-30 21:39:24.981 DEBUG 18896 --- [           main] mao.tools_xss.filter.XssFilter           : ignorePathList=null
2022-10-30 21:39:24.982 DEBUG 18896 --- [           main] mao.tools_xss.filter.XssFilter           : ignoreParamValueList=["samlp:LogoutRequest"]
2022-10-30 21:39:24.982 DEBUG 18896 --- [           main] mao.tools_xss.filter.XssFilter           : XSS fiter [XSSFilter] init end
2022-10-30 21:39:25.153  INFO 18896 --- [           main] o.s.b.a.w.s.WelcomePageHandlerMapping    : Adding welcome page: class path resource [static/index.html]
2022-10-30 21:39:25.264  INFO 18896 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
2022-10-30 21:39:25.275  INFO 18896 --- [           main] mao.use_starter.UseStarterApplication    : Started UseStarterApplication in 1.427 seconds (JVM running for 2.046)
```

第七步：访问

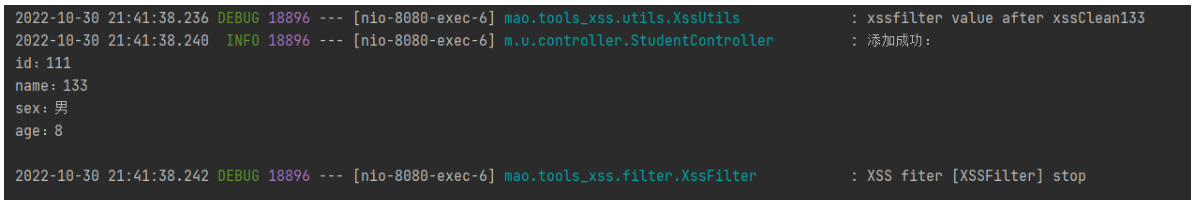


```
2022-10-30 21:39:45.985  INFO 18896 --- [nio-8080-exec-2] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
2022-10-30 21:39:45.989 DEBUG 18896 --- [nio-8080-exec-2] mao.tools_xss.filter.XssFilter           : XSS fiter [XSSFilter] starting
2022-10-30 21:39:45.989 DEBUG 18896 --- [nio-8080-exec-1] mao.tools_xss.filter.XssFilter           : XSS fiter [XSSFilter] starting
2022-10-30 21:39:45.990 DEBUG 18896 --- [nio-8080-exec-1] mao.tools_xss.filter.XssFilter           : has xssfiter path[/student/init] need XssFilter, go to XssRequestWrapper
2022-10-30 21:39:45.990 DEBUG 18896 --- [nio-8080-exec-2] mao.tools_xss.filter.XssFilter           : has xssfiter path[/] need XssFilter, go to XssRequestWrapper
2022-10-30 21:39:45.999 DEBUG 18896 --- [nio-8080-exec-2] mao.tools_xss.utils.XssUtils             :  start read XSS configfile [antisamy-slashdot-1.4.4.xml]
2022-10-30 21:39:46.002  INFO 18896 --- [nio-8080-exec-1] m.u.controller.StudentController         : 初始化完成
2022-10-30 21:39:46.007 DEBUG 18896 --- [nio-8080-exec-2] mao.tools_xss.utils.XssUtils             : read XSS configfile [antisamy-slashdot-1.4.4.xml] success
```

```
2022-10-30 21:39:46.007 DEBUG 18896 --- [nio-8080-exec-2] mao.tools_xss.utils.XssUtils             : raw value before xssClean: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
2022-10-30 21:39:46.016 DEBUG 18896 --- [nio-8080-exec-1] mao.tools_xss.filter.XssFilter           : XSS fiter [XSSFilter] stop
2022-10-30 21:39:46.028 DEBUG 18896 --- [nio-8080-exec-2] mao.tools_xss.utils.XssUtils             : xssfilter value after xssCleantext/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
2022-10-30 21:39:46.034 DEBUG 18896 --- [nio-8080-exec-2] mao.tools_xss.utils.XssUtils             : raw value before xssClean: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
2022-10-30 21:39:46.035 DEBUG 18896 --- [nio-8080-exec-2] mao.tools_xss.utils.XssUtils             : xssfilter value after xssCleantext/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
2022-10-30 21:39:46.038 DEBUG 18896 --- [nio-8080-exec-2] mao.tools_xss.filter.XssFilter           : XSS fiter [XSSFilter] stop
```

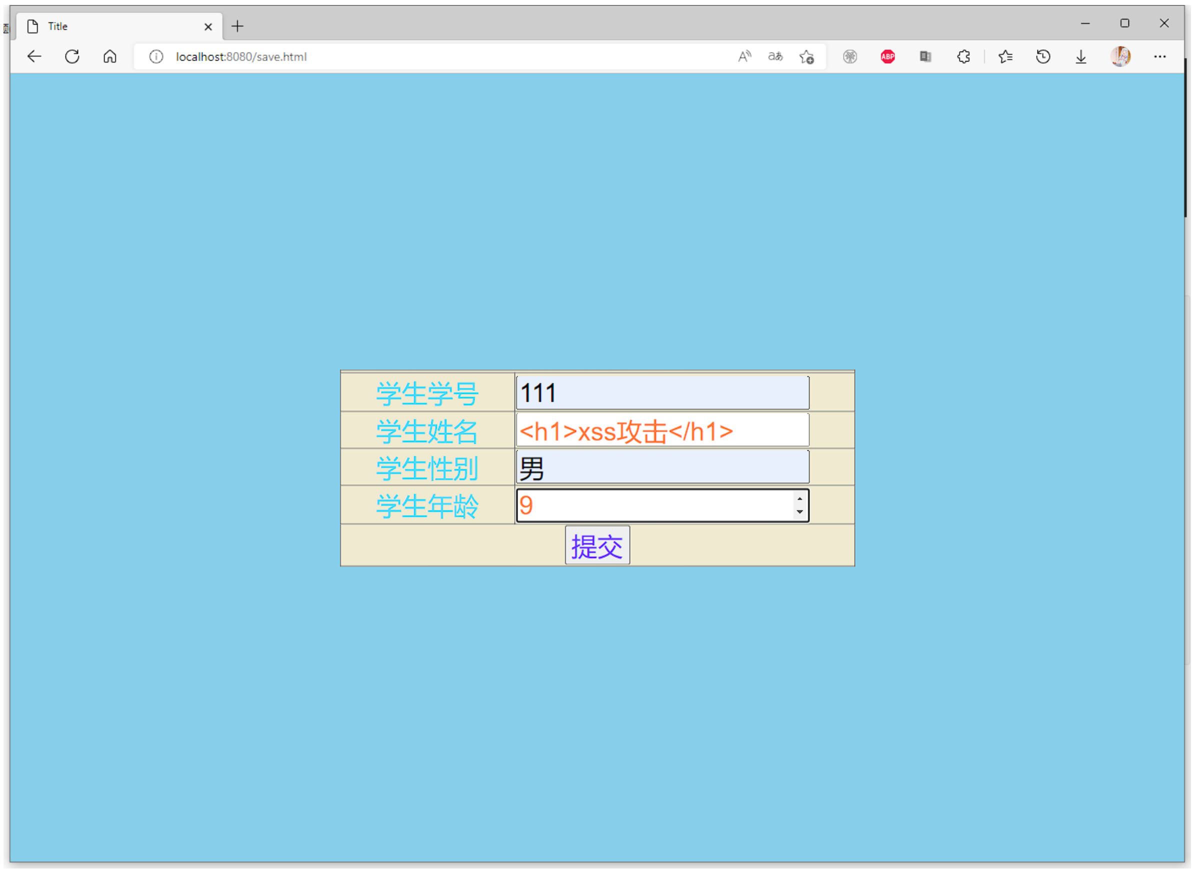


学生学号	111
学生姓名	133<script>alert("xss攻击
学生性别	男
学生年龄	8

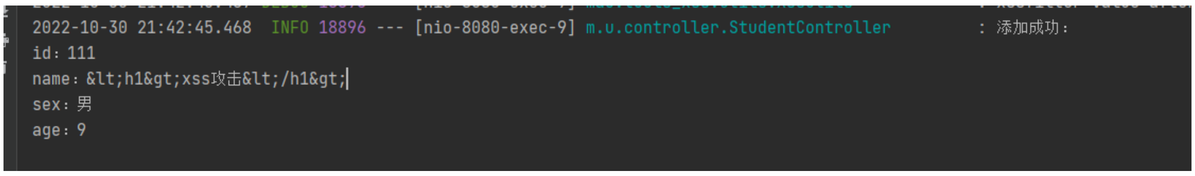


```
2022-10-30 21:41:38.236 DEBUG 18896 --- [nio-8080-exec-6] mao.tools_xss.utils.XssUtils             : xssfilter value after xssClean133
2022-10-30 21:41:38.240  INFO 18896 --- [nio-8080-exec-6] m.u.controller.StudentController         : 添加成功：
id：111
name：133
sex：男
age：8

2022-10-30 21:41:38.242 DEBUG 18896 --- [nio-8080-exec-6] mao.tools_xss.filter.XssFilter           : XSS fiter [XSSFilter] stop
```

```
2022-10-30 21:41:38.214 DEBUG 18896 --- [nio-8080-exec-6] mao.tools_xss.filter.XssFilter           : XSS fiter [XSSFilter] starting
2022-10-30 21:41:38.215 DEBUG 18896 --- [nio-8080-exec-6] mao.tools_xss.filter.XssFilter           : has xssfiter path[/student] need XssFilter, go to XssRequestWrapper
2022-10-30 21:41:38.233 DEBUG 18896 --- [nio-8080-exec-6] mao.tools_xss.utils.XssUtils             : raw value before xssClean: 133<script>alert("xss攻击")</script>
2022-10-30 21:41:38.235 DEBUG 18896 --- [nio-8080-exec-6] mao.tools_xss.utils.XssUtils             : 出于安全的原因，标记script不被允许。此标记不应该影响输入的显示。
2022-10-30 21:41:38.236 DEBUG 18896 --- [nio-8080-exec-6] mao.tools_xss.utils.XssUtils             : xssfilter value after xssClean133
2022-10-30 21:41:38.240  INFO 18896 --- [nio-8080-exec-6] m.u.controller.StudentController         : 添加成功：
id：111
name：133
sex：男
age：8

2022-10-30 21:41:38.242 DEBUG 18896 --- [nio-8080-exec-6] mao.tools_xss.filter.XssFilter           : XSS fiter [XSSFilter] stop
```

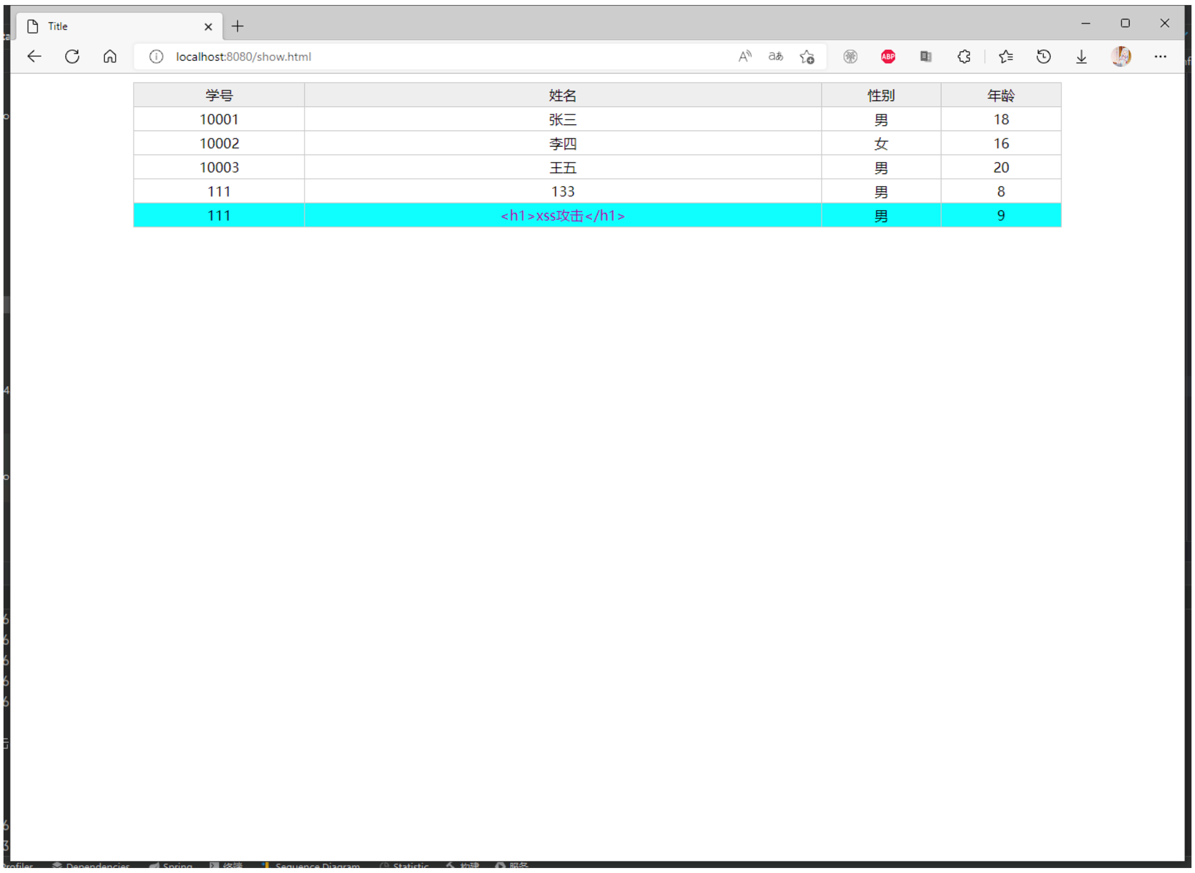


学生学号	111
学生姓名	<h1>xss攻击</h1>
学生性别	男
学生年龄	9
	提交

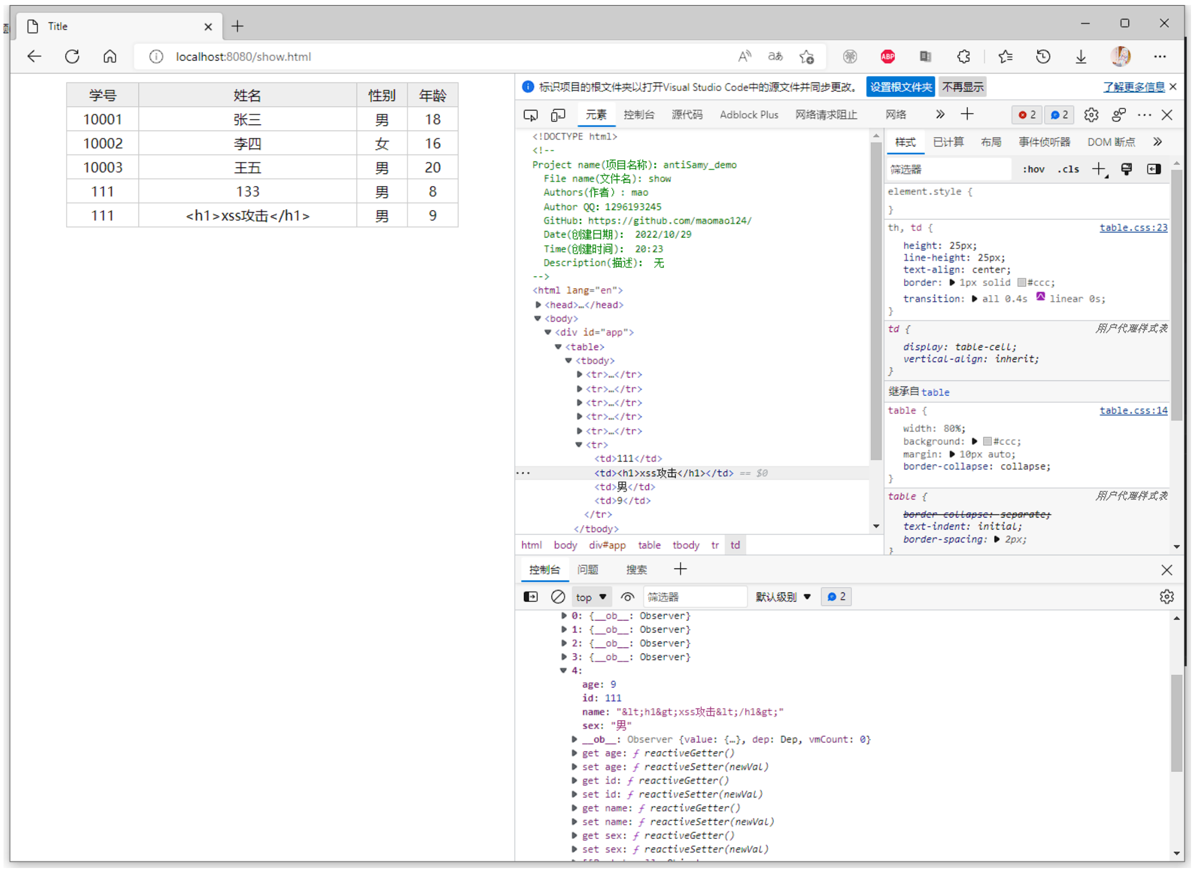


```
2022-10-30 21:42:45.468  INFO 18896 --- [nio-8080-exec-9] m.u.controller.StudentController         : 添加成功：
id：111
name：&lt;h1&gt;xss攻击&lt;/h1&gt;
sex：男
age：9
```

```
2022-10-30 21:42:45.463 DEBUG 18896 --- [nio-8080-exec-9] mao.tools_xss.filter.XssFilter           : XSS fiter [XSSFilter] starting
2022-10-30 21:42:45.464 DEBUG 18896 --- [nio-8080-exec-9] mao.tools_xss.filter.XssFilter           : has xssfiter path[/student] need XssFilter, go to XssRequestWrapper
2022-10-30 21:42:45.465 DEBUG 18896 --- [nio-8080-exec-9] mao.tools_xss.utils.XssUtils             : raw value before xssClean: <h1>xss攻击</h1>
2022-10-30 21:42:45.467 DEBUG 18896 --- [nio-8080-exec-9] mao.tools_xss.utils.XssUtils             : The h1 tag has been encoded for security reasons. The contents of the tag will remain in place.
2022-10-30 21:42:45.467 DEBUG 18896 --- [nio-8080-exec-9] mao.tools_xss.utils.XssUtils             : xssfilter value after xssClean&lt;h1&gt;xss攻击&lt;/h1&gt;
2022-10-30 21:42:45.468  INFO 18896 --- [nio-8080-exec-9] m.u.controller.StudentController         : 添加成功:
id: 111
name: &lt;h1&gt;xss攻击&lt;/h1&gt;
sex: 男
age: 9

2022-10-30 21:42:45.469 DEBUG 18896 --- [nio-8080-exec-9] mao.tools_xss.filter.XssFilter           : XSS fiter [XSSFilter] stop
```



学号	姓名	性别	年龄
10001	张三	男	18
10002	李四	女	16
10003	王五	男	20
111	133	男	8
111	<h1>xss攻击</h1>	男	9



localhost:8080/show.html

学号	姓名	性别	年龄
10001	张三	男	18
10002	李四	女	16
10003	王五	男	20
111	133	男	8
111	<h1>xss攻击</h1>	男	9

```
<!DOCTYPE html>
<!--
Project name(项目名称): antiSamy_demo
  File name(文件名): show
  Authors(作者) : mao
  Author QQ: 1296193245
  GitHub: https://github.com/maomao124/
  Date(创建日期):  2022/10/29
  Time(创建时间):  20:23
  Description(描述):  无
-->
<html lang="en">
 ▶<head>…</head>
 ▼<body>
   ▼<div id="app">
     ▼<table>
       ▼<tbody>
         ▶<tr>…</tr>
         ▶<tr>…</tr>
         ▶<tr>…</tr>
         ▶<tr>…</tr>
         ▶<tr>…</tr>
         ▼<tr>
            <td>111</td>
            <td><h1>xss攻击</h1></td> == $0
            <td>男</td>
            <td>9</td>
          </tr>
        </tbody>
```

```
element.style {
}
th, td {                                  table.css:23
    height: 25px;
    line-height: 25px;
    text-align: center;
    border: ▶ 1px solid #ccc;
    transition: ▶ all 0.4s linear 0s;
}
td {                                      用户代理样式表
    display: table-cell;
    vertical-align: inherit;
}
继承自 table
table {                                   table.css:14
    width: 80%;
    background: ▶ #ccc;
    margin: ▶ 10px auto;
    border-collapse: collapse;
}
table {                                   用户代理样式表
    ~~border-collapse: separate;~~
    text-indent: initial;
    border-spacing: ▶ 2px;
}
```

```
 ▶ 0: {__ob__: Observer}
 ▶ 1: {__ob__: Observer}
 ▶ 2: {__ob__: Observer}
 ▶ 3: {__ob__: Observer}
 ▼ 4:
     age: 9
     id: 111
     name: "&lt;h1&gt;xss攻击&lt;/h1&gt;"
     sex: "男"
   ▶ __ob__: Observer {value: {…}, dep: Dep, vmCount: 0}
   ▶ get age: ƒ reactiveGetter()
   ▶ set age: ƒ reactiveSetter(newVal)
   ▶ get id: ƒ reactiveGetter()
   ▶ set id: ƒ reactiveSetter(newVal)
   ▶ get name: ƒ reactiveGetter()
   ▶ set name: ƒ reactiveSetter(newVal)
   ▶ get sex: ƒ reactiveGetter()
   ▶ set sex: ƒ reactiveSetter(newVal)
```

---

end

---

by mao

2022 10 30
