

## AntiSamy

XSS介绍

AntiSamy介绍

AntiSamy入门案例

演示xss攻击

- 第一步：创建工程antiSamy\_demo
- 第二步：编写实体类Student
- 第三步：编写StudentController
- 第四步：编写样式表form.css
- 第五步：编写样式表link.css
- 第六步：编写样式表table.css
- 第七步：添加vue和axios库
- 第八步：编写index.html
- 第九步：编写show.html
- 第十步：编写save.html
- 第十一步：启动程序
- 第十二步：访问
- 第十三步：初始化数据
- 第十四步：查看所有学生的信息
- 第十五步：添加学生信息
- 第十六步：查询所有学生的信息

解决xss攻击

- 第一步：添加依赖
- 第二步：创建策略文件/resources/antisamy.xml
- 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
- 第四步：创建XssRequestWrapper类
- 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
- 第六步：复制antisamy-ebay.xml文件到/resources目录下
- 第七步：修改XssRequestWrapper类
- 第八步：启动程序
- 第九步：访问
- 第十步：更改StudentController
- 第十一步：编写save2.html
- 第十二步：修改index.html
- 第十三步：重启并访问
- 第十四步：查看控制台日志
- 第十五步：编写XssFilterService
- 第十六步：修改StudentController
- 第十七步：重启并访问
- 第十八步：编写类XssStringJsonDeserializer
- 第十九步：修改配置类AntiSamyConfig
- 第二十步：修改StudentController
- 第二十一部：重启并访问服务

自定义spring boot starter

开发starter

- 第一步：初始化项目
- 第二步：修改pom文件
- 第三步：编写工具类XssUtils
- 第四步：编写类XssRequestWrapper
- 第五步：编写类XssStringJsonDeserializer
- 第六步：编写过滤器类XssFilter
- 第七步：拷贝之前的service包到此项目中，并更改
- 第八步：编写配置类XssAuthConfiguration
- 第九步：拷贝antisamy-slashdot-1.4.4.xml文件和antisamy.xsd文件到资源目录中
- 第十步：编写spring.factories文件

使用starter

第一步：导入tools-xss的依赖

第二步：导入之前项目的静态资源文件

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

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

第六步：启动程序

第七步：访问

---

# 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。

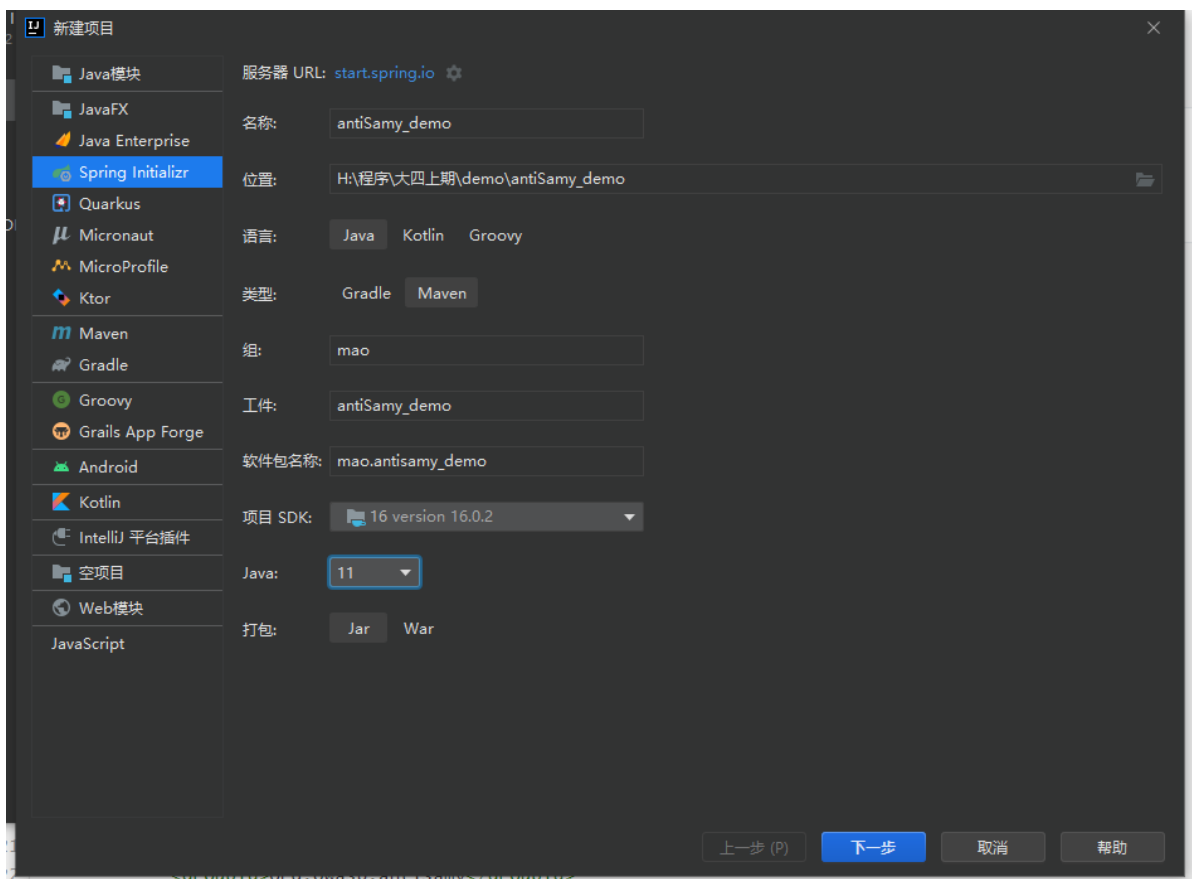
AntiSamy的maven坐标：

```
1 <dependency>
2   <groupId>org.owasp.antisamy</groupId>
3   <artifactId>antisamy</artifactId>
4   <version>1.5.7</version>
5 </dependency>
```

## AntiSamy入门案例

### 演示xss攻击

#### 第一步：创建工程antiSamy\_demo



## 第二步：编写实体类Student

```
1 package mao.antisamy_demo.entity;
2
3 /**
4  * Project name(项目名称): antiSamy_demo
5  * Package(包名): mao.antisamy_demo.entity
6  * Class(类名): Student
7  * Author(作者): mao
8  * Author QQ: 1296193245
9  * Github: https://github.com/maomao124/
10 * Date(创建日期): 2022/10/29
11 * Time(创建时间): 20:01
12 * Version(版本): 1.0
13 * Description(描述): 无
14 */
15
16
17 public class Student
18 {
19     /**
20      * id
21      */
22     private long id;
23     /**
24      * 名字
25      */
26     private String name;
27     /**
28      * 性别
29      */
30     private String sex;
31     /**
32      * 年龄
33      */
34     private int age;
35
36     /**
37      * Instantiates a new Student.
38      */
39     public Student()
40     {
41
42     }
43
44     /**
45      * Instantiates a new Student.
46      *
47      * @param id the id
48      * @param name the name
49      * @param sex the sex
50      * @param age the age
```

```

51     */
52     public Student(long id, String name, String sex, int age)
53     {
54         this.id = id;
55         this.name = name;
56         this.sex = sex;
57         this.age = age;
58     }
59
60     /**
61      * Gets id.
62      *
63      * @return the id
64      */
65     public long getId()
66     {
67         return id;
68     }
69
70     /**
71      * Sets id.
72      *
73      * @param id the id
74      */
75     public void setId(long id)
76     {
77         this.id = id;
78     }
79
80     /**
81      * Gets name.
82      *
83      * @return the name
84      */
85     public String getName()
86     {
87         return name;
88     }
89
90     /**
91      * Sets name.
92      *
93      * @param name the name
94      */
95     public void setName(String name)
96     {
97         this.name = name;
98     }
99
100    /**
101     * Gets sex.
102     *
103     * @return the sex
104     */
105    public String getSex()
106    {
107        return sex;
108    }

```

```

109
110     /**
111      * Sets sex.
112      *
113      * @param sex the sex
114      */
115     public void setSex(String sex)
116     {
117         this.sex = sex;
118     }
119
120     /**
121      * Gets age.
122      *
123      * @return the age
124      */
125     public int getAge()
126     {
127         return age;
128     }
129
130     /**
131      * Sets age.
132      *
133      * @param age the age
134      */
135     public void setAge(int age)
136     {
137         this.age = age;
138     }
139
140     @Override
141     @SuppressWarnings("all")
142     public String toString()
143     {
144         final StringBuilder stringbuilder = new StringBuilder();
145         stringbuilder.append("id: ").append(id).append('\n');
146         stringbuilder.append("name: ").append(name).append('\n');
147         stringbuilder.append("sex: ").append(sex).append('\n');
148         stringbuilder.append("age: ").append(age).append('\n');
149         return stringbuilder.toString();
150     }
151 }

```

### 第三步：编写StudentController

```

1 package mao.antisamy_demo.controller;
2
3 import mao.antisamy_demo.entity.Student;
4 import org.apache.juli.logging.Log;
5 import org.slf4j.Logger;

```

```

6  import org.slf4j.LoggerFactory;
7  import org.springframework.web.bind.annotation.*;
8
9  import java.util.ArrayList;
10 import java.util.Collections;
11 import java.util.List;
12
13 /**
14  * Project name(项目名称): antiSamy_demo
15  * Package(包名): mao.antisamy_demo.controller
16  * Class(类名): StudentController
17  * Author(作者): mao
18  * Author QQ: 1296193245
19  * GitHub: https://github.com/maomao124/
20  * Date(创建日期): 2022/10/29
21  * Time(创建时间): 20:05
22  * Version(版本): 1.0
23  * Description(描述): 无
24  */
25
26 @RestController
27 @RequestMapping("/student")
28 public class StudentController
29 {
30     private static final List<Student> list =
31     Collections.synchronizedList(new ArrayList<>());
32
33     private static final Logger log =
34     LoggerFactory.getLogger(StudentController.class);
35
36     @PostMapping("/init")
37     public synchronized void init()
38     {
39         Student student1 = new Student(10001, "张三", "男", 18);
40         Student student2 = new Student(10002, "李四", "女", 16);
41         Student student3 = new Student(10003, "王五", "男", 20);
42         list.clear();
43         list.add(student1);
44         list.add(student2);
45         list.add(student3);
46         log.info("初始化完成");
47     }
48
49     @PostMapping
50     public boolean save(@RequestBody Student student)
51     {
52         list.add(student);
53         log.info("添加成功: \n" + student);
54         return true;
55     }
56
57     @GetMapping
58     public List<Student> getAll()
59     {
60         log.info("查询所有: \n" + list);
61         return list;
62     }
63 }

```

#### 第四步：编写样式表form.css

```
1  /*
2    Project name(项目名称): antiSamy_demo
3    File name(文件名): form
4    Author(作者): mao
5    Author QQ: 1296193245
6    GitHub: https://github.com/maomao124/
7    Date(创建日期): 2022/10/29
8    Time(创建时间): 20:54
9    Version(版本): 1.0
10   Description(描述): 无
11   */
12
13
14  /*表单位置*/
15  div.form_position {
16      position: absolute;
17      top: 50%;
18      left: 50%;
19      transform: translate(-50%, -50%);
20  }
21
22  /*表单的边框*/
23  div.form {
24      border: 10px skyblue dotted;
25  }
26
27
28  /*表*/
29  table {
30      width: 600px;
31      border-collapse: collapse;
32      color: #4eda55;
33      background-color: #ffe4da;
34      transition: all 1s linear 0s;
35  }
36
37  table:hover {
38      background-color: #b4ffab;
39      /*width: 800px;*/
40      /*transition: all 1s linear 0s;*/
41  }
42
43
44  /*设置字体*/
45  td, input, input.input {
46      font-size: 30px;
47  }
```



```

48
49 input {
50     /*font-size: 24px;*/
51     /*width: 90%;*/
52     color: coral;
53 }
54
55 input.input {
56     width: 85%;
57     color: coral;
58     transition: all 0.5s linear 0s;
59 }
60
61 input.input:hover {
62     width: 98%;
63     transition: all 0.5s linear 0s;
64     background-color: #fcffee;
65 }
66
67
68 /*设置提示*/
69 .prompt {
70     text-align: center;
71     width: 200px;
72     transition: all 1s linear 0.2s;
73 }
74
75 .prompt:hover {
76     transition: all 1s linear 0.2s;
77     color: #ff2e2f;
78 }
79
80 /*提交按钮*/
81 input.submit {
82     color: #6739ff;
83 }
84
85 input.submit:hover {
86
87 }
88
89 /*最上面的字*/
90 div.text {
91     border: 10px violet dotted;
92     text-align: center;
93     font-size: 32px;
94     color: tomato;
95     background: bisque;
96 }
97
98 /*最上面的字的位置*/
99 div.text_position {
100     position: absolute;
101     top: 2%;
102     left: 50%;
103     transform: translate(-50%, 0%);
104 }

```

## 第五步：编写样式表link.css

```
1  /*
2   Project name(项目名称): antiSamy_demo
3   File name(文件名): link
4   Author(作者): mao
5   Author QQ: 1296193245
6   Github: https://github.com/maomao124/
7   Date(创建日期): 2022/10/29
8   Time(创建时间): 20:17
9   Version(版本): 1.0
10  Description(描述): 无
11  */
12
13
14  a {
15      color: tomato;
16      text-decoration: none;
17      display: flex;
18      justify-content: center;
19      align-items: center;
20
21      padding: 0;
22      list-style-type: none;
23      font-size: 22px;
24      width: 10em;
25      height: 2em;
26      text-align: center;
27      line-height: 2em;
28      font-family: sans-serif;
29      text-transform: capitalize;
30      position: relative;
31      margin: 0.8em;
32  }
33
34  /* 添加了左右两个圆点 */
35  a::before, a::after {
36      content: '';
37      position: absolute;
38      width: 0.6em;
39      height: 0.6em;
40      background-color: gainsboro;
41      top: calc(50% - 0.3em);
42      border-radius: 50%;
43      transition: 0.5s cubic-bezier(0.5, -0.5, 0.25, 1.5);
44  }
45
46  a::before {
47      left: 0;
48      z-index: -1;
```

```

49 }
50
51 a::after {
52     right: 0;
53     z-index: -2;
54 }
55
56 /* 添加悬浮效果 */
57 a:hover {
58     color: deeppink;
59 }
60
61 /* 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 */
62 a:hover::before, a:hover::after {
63     width: 100%;
64     height: 100%;
65     border-radius: 0;
66     background-color: dodgerblue;
67 }
68
69 a:hover::before {
70     top: 0;
71     left: -0.2em;
72 }
73
74 a:hover::after {
75     right: -0.2em;
76     filter: brightness(0.8);
77 }

```

## 第六步：编写样式表table.css

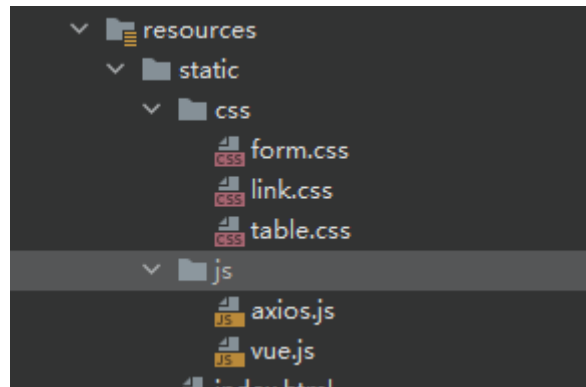
```

1  /*
2   Project name(项目名称): antiSamy_demo
3   File name(文件名): table
4   Author(作者): mao
5   Author QQ: 1296193245
6   Github: https://github.com/maomao124/
7   Date(创建日期): 2022/10/29
8   Time(创建时间): 20:33
9   Version(版本): 1.0
10  Description(描述): 无
11  */
12
13
14  table {
15      width: 80%;
16      background: #ccc;
17      margin: 10px auto;
18      /*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。*/
19      /*collapse - 相邻单元格具有共享边框（折叠边框表格渲染模型）。*/

```

```
20     border-collapse: collapse;
21 }
22
23 th, td {
24     height: 25px;
25     line-height: 25px;
26     text-align: center;
27     border: 1px solid #ccc;
28     transition: all 0.4s linear 0s;
29 }
30
31 th {
32     background: #eee;
33     font-weight: normal;
34 }
35
36 tr {
37     background: #fff;
38     transition: all 2s linear 0s;
39 }
40
41 tr:hover {
42     background: aqua;
43     transition: all 0.4s linear 0s;
44 }
45
46 td:hover {
47     color: magenta;
48     transition: all 0.4s linear 0s;
49 }
50
51 td a {
52     color: #06f;
53     text-decoration: none;
54 }
55
56 td a:hover {
57     color: #06f;
58     text-decoration: underline;
59     /*transition: all 1s linear 0s;*/
60 }
61
62 caption {
63     color: #28ffc7;
64     font-size: 28px;
65 }
```

## 第七步：添加vue和axios库



cdn:

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

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

## 第八步：编写index.html

```
1 <!DOCTYPE html>
2
3 <!--
4 Project name(项目名称): antiSamy_demo
5 File name(文件名): index
6 Authors(作者): mao
7 Author QQ: 1296193245
8 GitHub: https://github.com/maomao124/
9 Date(创建日期): 2022/10/29
10 Time(创建时间): 20:03
11 Description(描述): 无
12 -->
13
14 <html lang="en">
15 <head>
16   <meta charset="UTF-8">
17   <title>索引</title>
18   <link rel="stylesheet" href="/css/link.css">
19   <script src="/js/axios.js"></script>
20
```

```

21
22     <style>
23         body {
24             background-color: skyblue;
25         }
26
27         div.a {
28             position: absolute;
29             top: 50%;
30             left: 50%;
31             transform: translate(-50%, -50%);
32         }
33     </style>
34 </head>
35 <body>
36
37 <div class="a">
38
39     <a href="/show.html">显示所有学生信息</a>
40     <a href="/save.html">添加学生信息</a>
41     <a href="" onclick="init()">初始化数据</a>
42
43 </div>
44
45 <script>
46
47     function init()
48     {
49         //axios发起ajax请求
50         axios({
51             //请求的方式:
52             method: "post",
53             //请求的url:
54             url: "/student/init",
55             //url参数:
56             params:
57                 {},
58             //头信息:
59             headers:
60                 {},
61             //请求体参数:
62             data:
63                 {},
64         }).then(response =>
65         {
66             console.log(response);
67             alert("初始化数据成功");
68         }).catch(error =>
69         {
70             //console.log(error);
71             alert("网络异常! ");
72         })
73     }
74
75 </script>
76
77 </body>
78 </html>

```

## 第九步：编写show.html

```
1  <!DOCTYPE html>
2
3  <!--
4  Project name(项目名称): antiSamy_demo
5  File name(文件名): show
6  Authors(作者): mao
7  Author QQ: 1296193245
8  Github: https://github.com/maomao124/
9  Date(创建日期): 2022/10/29
10 Time(创建时间): 20:23
11 Description(描述): 无
12 -->
13
14 <html lang="en">
15 <head>
16   <meta charset="UTF-8">
17   <title>Title</title>
18   <link rel="stylesheet" href="/css/table.css">
19   <script src="/js/axios.js"></script>
20   <script src="/js/vue.js"></script>
21 </head>
22 <body>
23
24 <div id="app">
25
26   <table>
27     <tr>
28       <th>学号</th>
29       <th>姓名</th>
30       <th>性别</th>
31       <th>年龄</th>
32     </tr>
33     <tr v-for="student in studentList">
34       <td v-html="student.id"></td>
35       <td v-html="student.name"></td>
36       <td v-html="student.sex"></td>
37       <td v-html="student.age"></td>
38     </tr>
39   </table>
40
41 </div>
42
43 </body>
44
45 <script>
46
47   var app = new Vue({
48     el: "#app",
49     data: {
```

```

50         studentList: null
51     },
52     method: {},
53     mounted: function ()
54     {
55         //axios发起ajax请求
56         axios({
57             //请求的方式:
58             method: "get",
59             //请求的url:
60             url: "/student",
61             //url参数:
62             params:
63                 {},
64             //头信息:
65             headers:
66                 {},
67             //请求体参数:
68             data:
69                 {},
70         }).then(response =>
71         {
72             console.log(response);
73             this.studentList = response.data;
74
75         }).catch(error =>
76         {
77             //console.log(error);
78             alert("网络异常!");
79         })
80     }
81 })
82
83 </script>
84 </html>

```

## 第十步：编写save.html

```

1  <!DOCTYPE html>
2
3  <!--
4  Project name(项目名称): antiSamy_demo
5      File name(文件名): save
6      Authors(作者): mao
7      Author QQ: 1296193245
8      GitHub: https://github.com/maomao124/
9      Date(创建日期): 2022/10/29
10     Time(创建时间): 20:23
11     Description(描述): 无
12 -->
13

```



```

14 <html lang="en">
15 <head>
16   <meta charset="UTF-8">
17   <title>Title</title>
18   <link rel="stylesheet" href="css/form.css">
19   <script src="js/axios.js"></script>
20   <script src="js/vue.js"></script>
21   <style>
22     body {
23       background-color: skyblue;
24     }
25   </style>
26 </head>
27 <body>
28
29 <div id="app">
30
31
32   <div class="form_position">
33     <div class="animated bounceInDown">
34       <div class="form">
35         <form action="" method="post">
36           <table border="1">
37             <tr>
38               <td colspan="2" align="center">
39                 </td>
40             </tr>
41             <tr>
42               <td class="prompt">学生学号</td>
43               <td>
44                 <label>
45                   <input v-model="student.id"
class="input" type="number" name="id">
46                 </label>
47               </td>
48             </tr>
49             <tr>
50               <td class="prompt">学生姓名</td>
51               <td>
52                 <label>
53                   <input v-model="student.name"
class="input" type="text" name="name">
54                 </label>
55               </td>
56             </tr>
57             <tr>
58               <td class="prompt">学生性别</td>
59               <td>
60                 <label>
61                   <input v-model="student.sex"
class="input" type="text" name="sex">
62                 </label>
63               </td>
64             </tr>
65             <tr>
66               <td class="prompt">学生年龄</td>
67               <td>
68                 <label>

```

```

69         <input v-model="student.age"
class="input" type="number" name="age">
70         </label>
71     </td>
72 </tr>
73 <tr>
74     <td colspan="2" align="center">
75         <input @click="f()" class="submit"
type="button" value="提交"/>
76     </td>
77 </tr>
78 </table>
79 </form>
80 </div>
81 </div>
82 </div>
83
84 </div>
85
86 <!--<script>alert("xss攻击")</script>-->
87 <!--<h1>xss攻击</h1>-->
88
89 <script>
90
91     var app = new Vue(
92     {
93         el: "#app",
94         data: {
95             student: {
96                 id: null,
97                 name: null,
98                 sex: null,
99                 age: null,
100             }
101         },
102         methods: {
103             f: function ()
104             {
105                 console.log("提交")
106                 console.log(this.student)
107                 var that = this;
108                 //axios发起ajax请求
109                 axios({
110                     //请求的方式:
111                     method: "post",
112                     //请求的url:
113                     url: "/student",
114                     //url参数:
115                     params:
116                     {},
117                     //头信息:
118                     headers:
119                     {},
120                     //请求体参数:
121                     data:
122                     {
123                         id: that.student.id,
124                         name: that.student.name,

```

```

125         sex: that.student.sex,
126         age: that.student.age,
127     },
128     }).then(response =>
129     {
130         console.log(response);
131         if (response.data === true)
132         {
133             alert("请求成功")
134         }
135         else
136         {
137             alert("失败")
138         }
139     })
140     .catch(error =>
141     {
142         //console.log(error);
143         alert("网络异常! ");
144     })
145     }
146     }
147     }
148     )
149
150 </script>
151
152 </body>
153 </html>

```

## 第十一步：启动程序

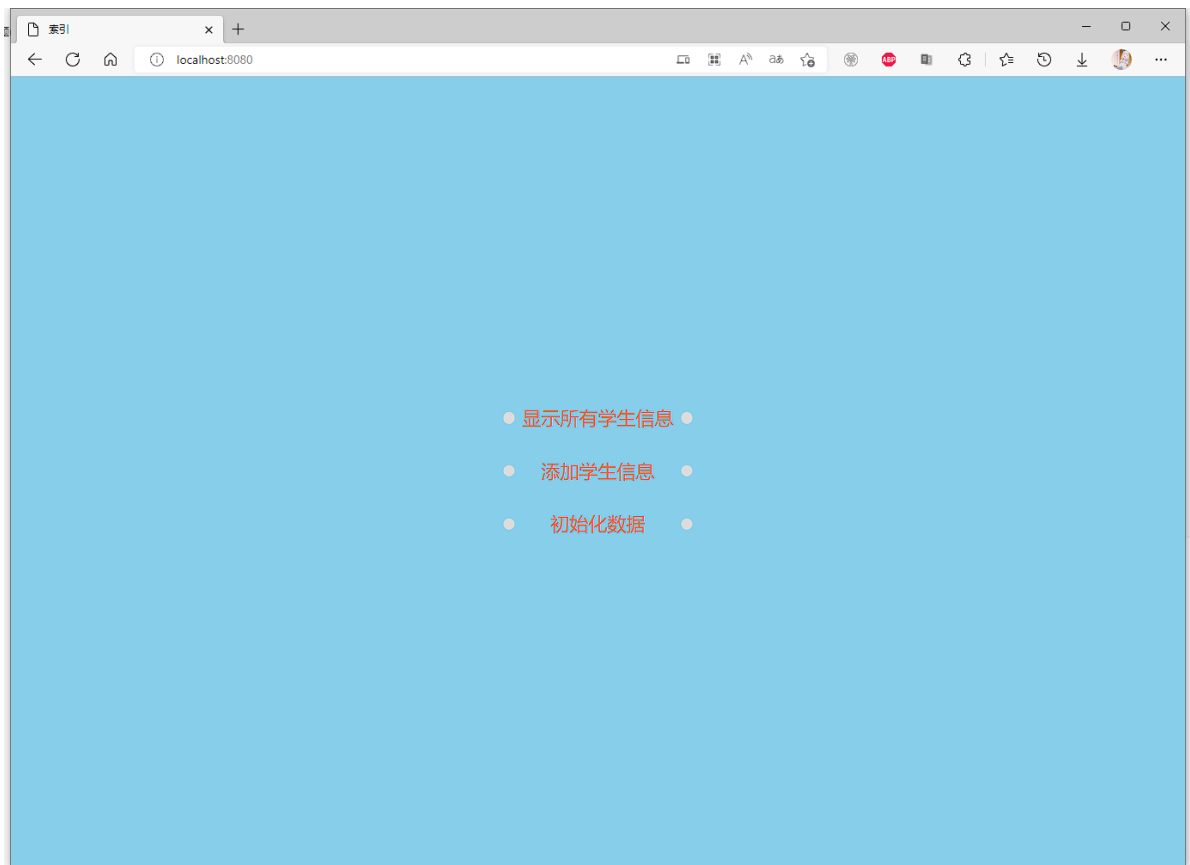
```

1
2      .   _ _ _       -         _ _ _ _
3     /\ / _-'_-- --( )- _ _ _ \ \ \ \
4    ( ( )\_ | '_| '_||'_ \V _` | \ \ \ \
5     \V __)|_| ||||| ||| (| | ) ) ) )
6      ' |__| ._|_| |_|_| |_ \| , | / / / /
7     =====|_|=====|___/=/_/_/_/
8     :: Spring Boot ::                                (v2.7.1)
9
10 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)
```

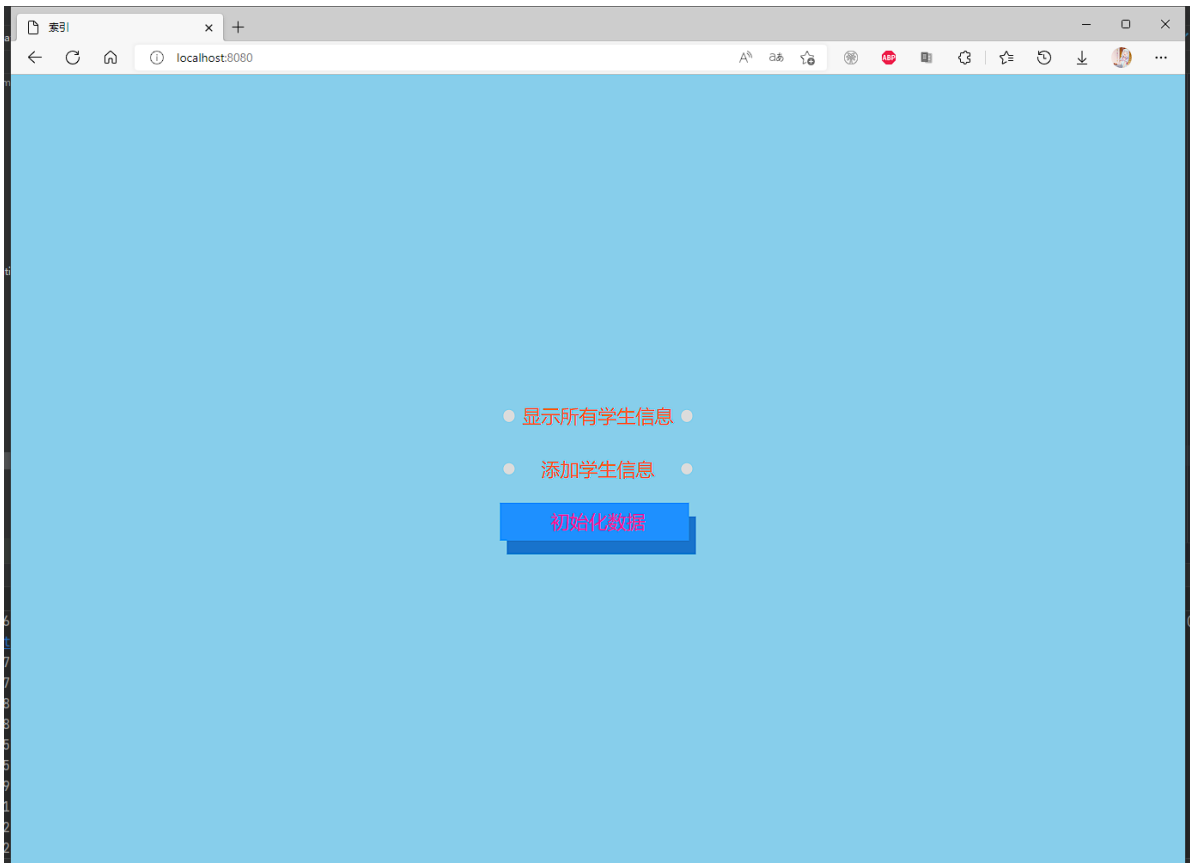
```
11 2022-10-30 13:14:22.271 INFO 17656 --- [          main]
    m.antisamy_demo.AntiSamyDemoApplication : No active profile set, falling
    back to 1 default profile: "default"
12 2022-10-30 13:14:22.970 INFO 17656 --- [          main]
    o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s):
    8080 (http)
13 2022-10-30 13:14:22.980 INFO 17656 --- [          main]
    o.apache.catalina.core.StandardService : Starting service [Tomcat]
14 2022-10-30 13:14:22.980 INFO 17656 --- [          main]
    org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
    Tomcat/9.0.64]
15 2022-10-30 13:14:23.057 INFO 17656 --- [          main] o.a.c.c.C.
    [Tomcat].[localhost].[/] : Initializing Spring embedded
    WebApplicationContext
16 2022-10-30 13:14:23.057 INFO 17656 --- [          main]
    w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
    initialization completed in 746 ms
17 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]
18 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 ''
19 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

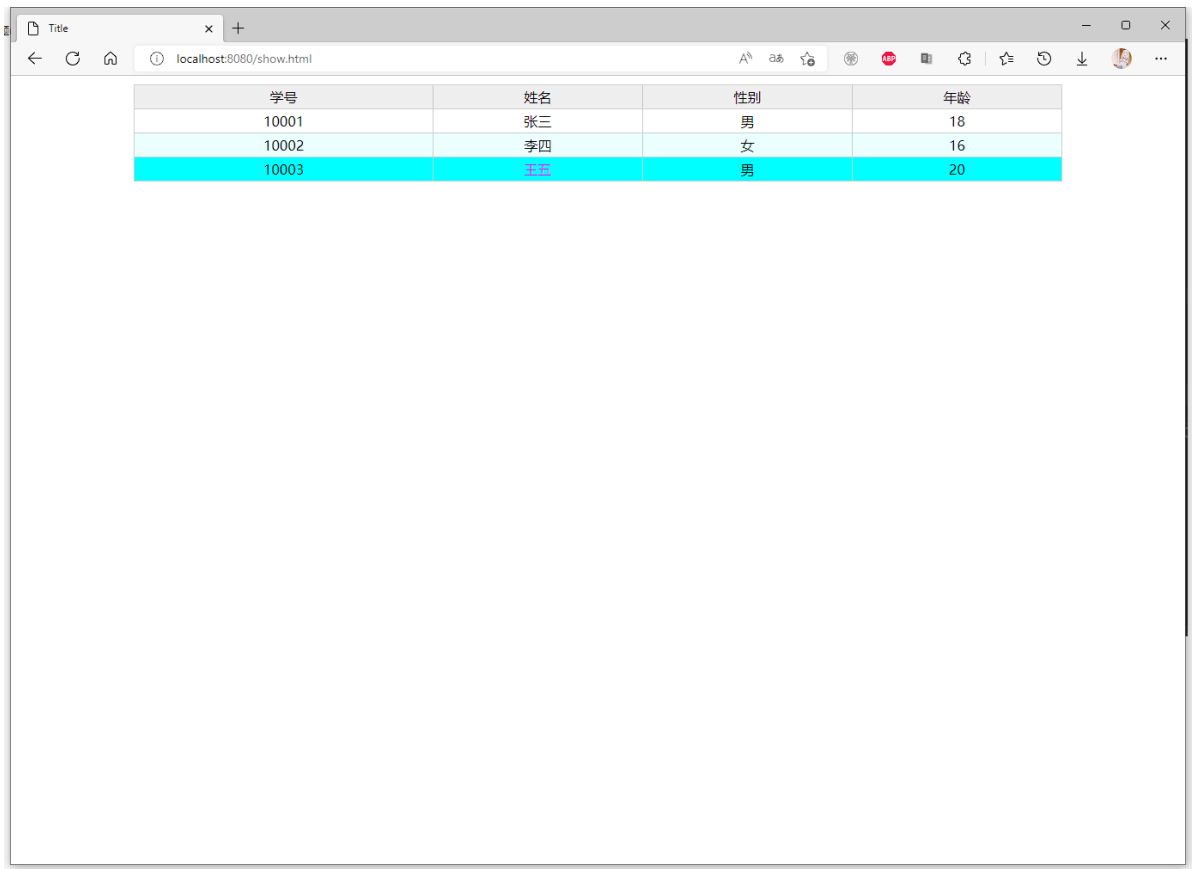
<http://localhost:8080/>



### 第十三步：初始化数据



第十四步：查看所有学生的信息



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/show.html'. The browser's title bar says 'Title'. The main content area displays a table with four columns: '学号' (Student ID), '姓名' (Name), '性别' (Gender), and '年龄' (Age). The table contains three rows of data. The first row has a light blue background, the second row has a light green background, and the third row has a light red background. The third row's '姓名' cell contains a red text '王五'.

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

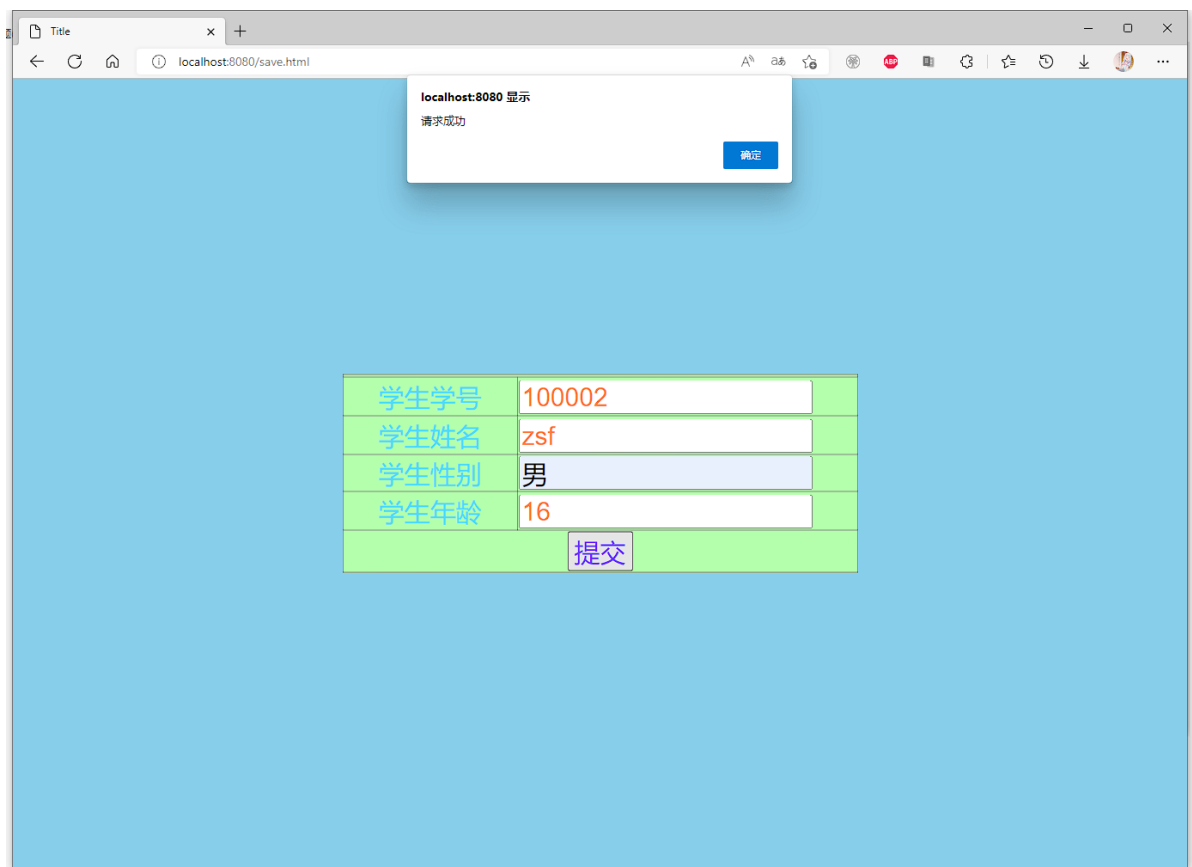
第十五步：添加学生信息

localhost:8080/save.html

学生学号	<input type="text"/>
学生姓名	<input type="text"/>
学生性别	<input type="text"/>
学生年龄	<input type="text"/>
<input type="button" value="提交"/>	

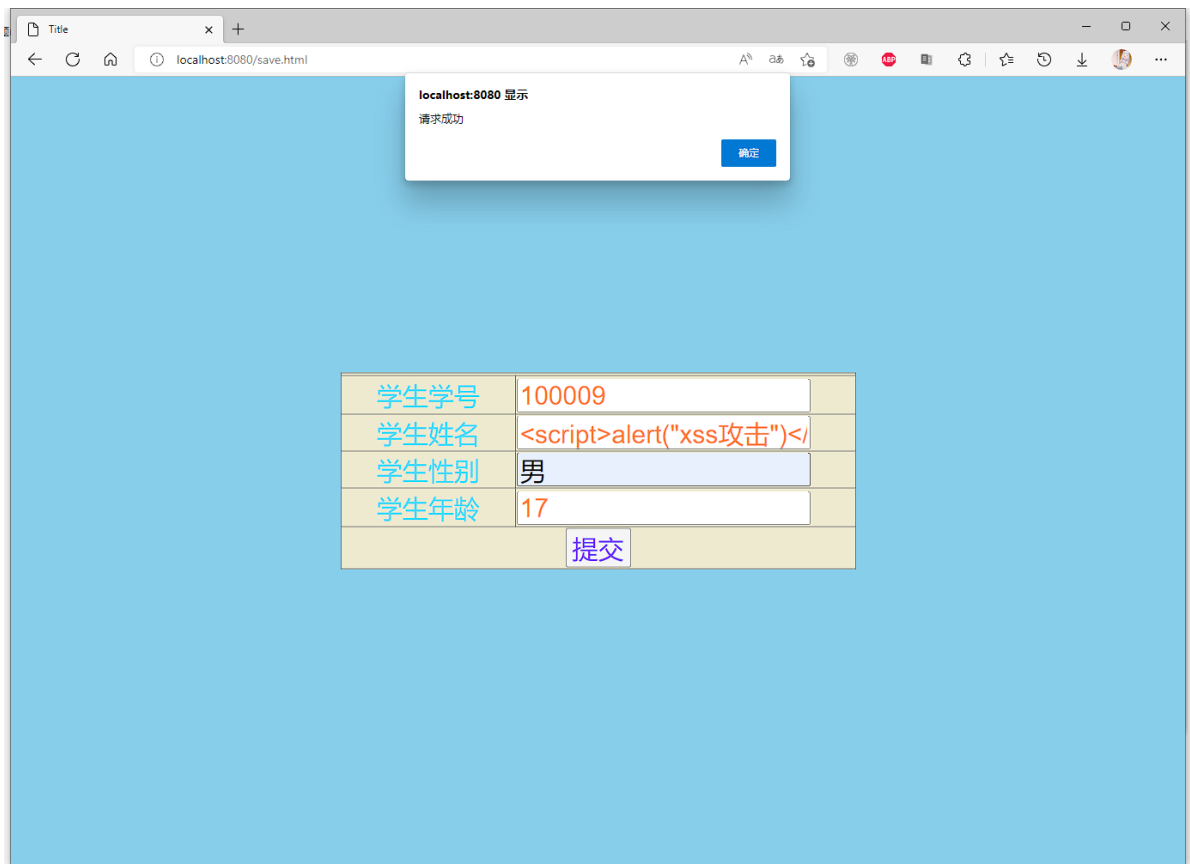
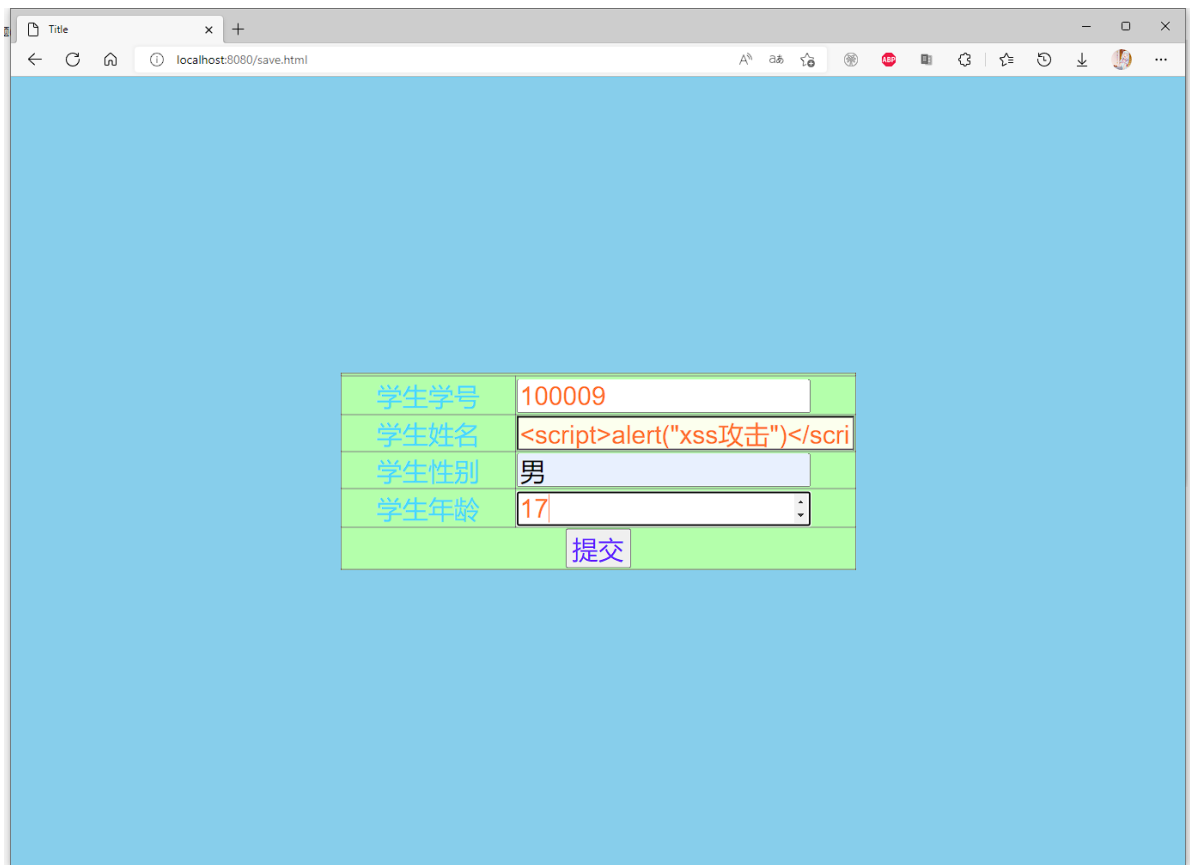
localhost:8080/save.html

学生学号	<input type="text" value="100002"/>
学生姓名	<input type="text" value="zsf"/>
学生性别	<input type="text" value="男"/>
学生年龄	<input type="text" value="16"/>
<input type="button" value="提交"/>	



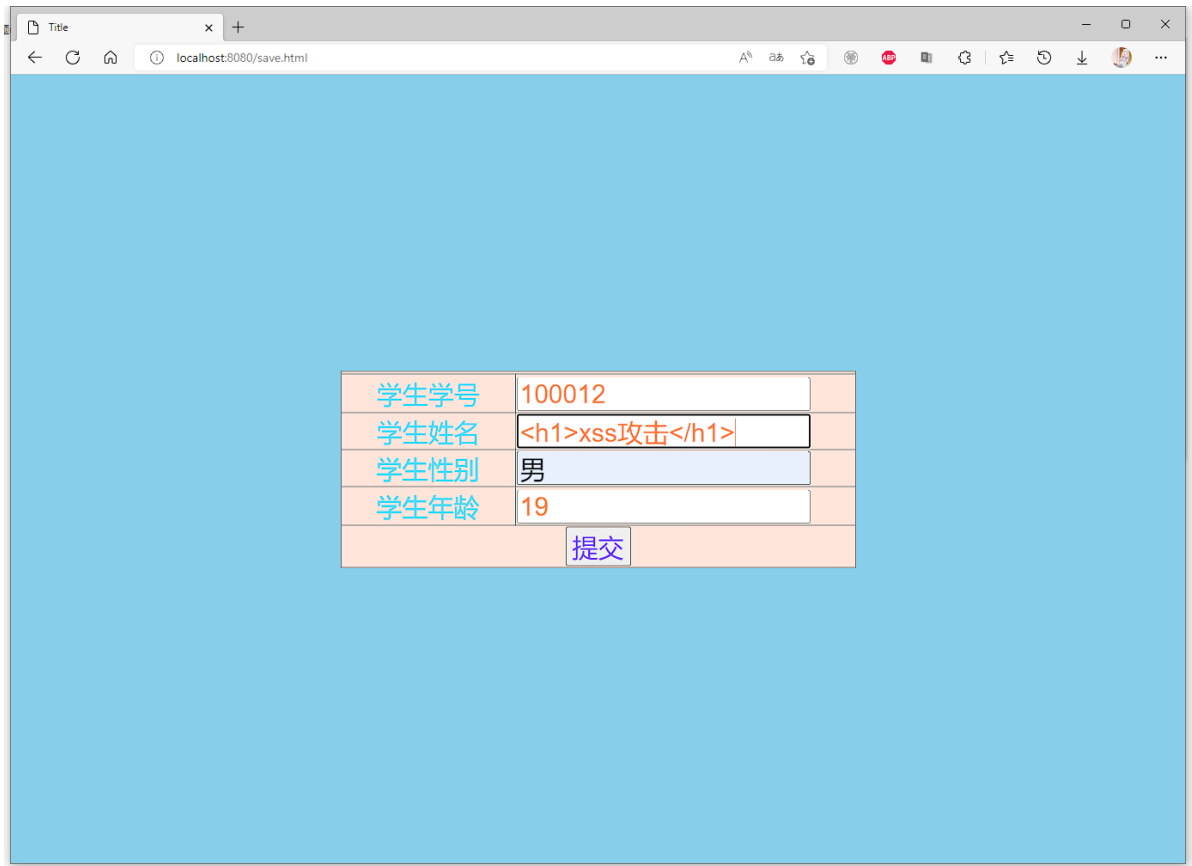
```
]
2022-10-30 13:18:32.477 INFO 17656 --- [nio-8080-exec-1] m.a.controller.StudentController : 添加成功:
id: 100002
name: zsf
sex: 男
age: 16
```

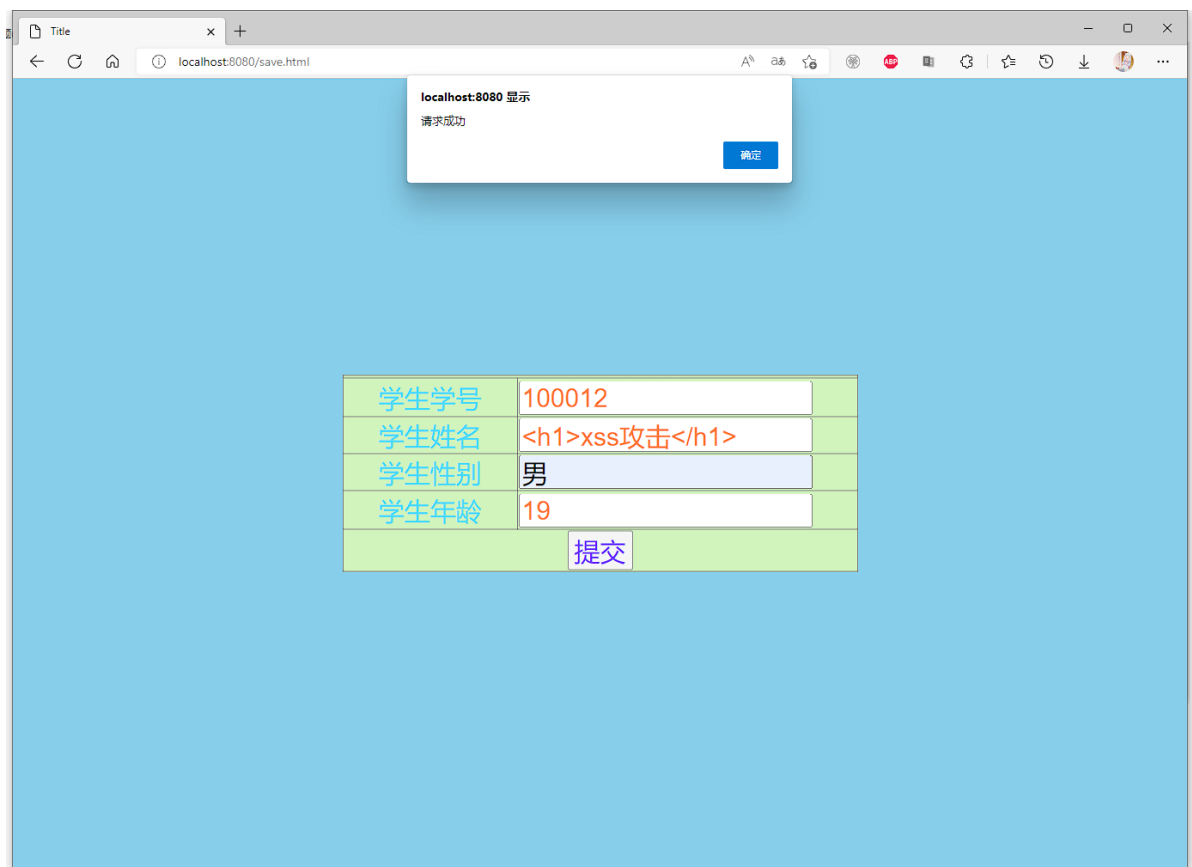




```
age: 10

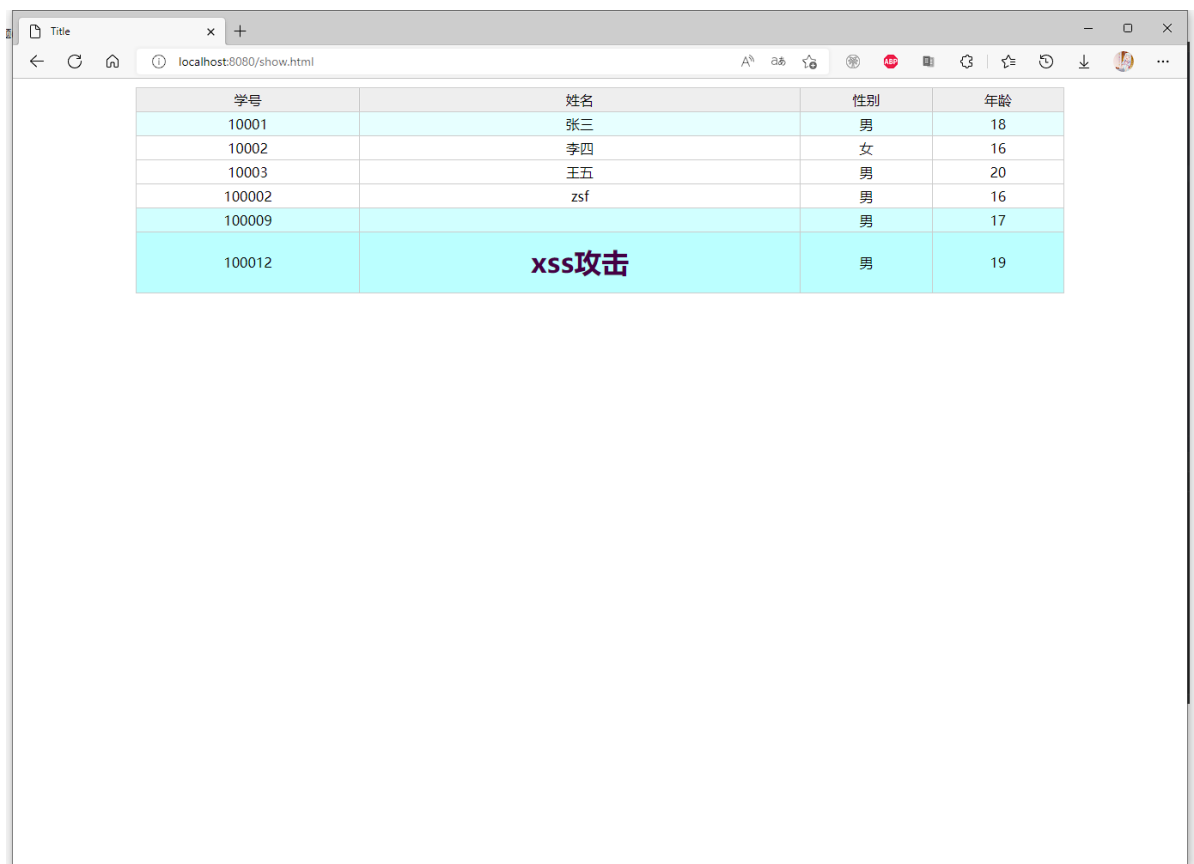
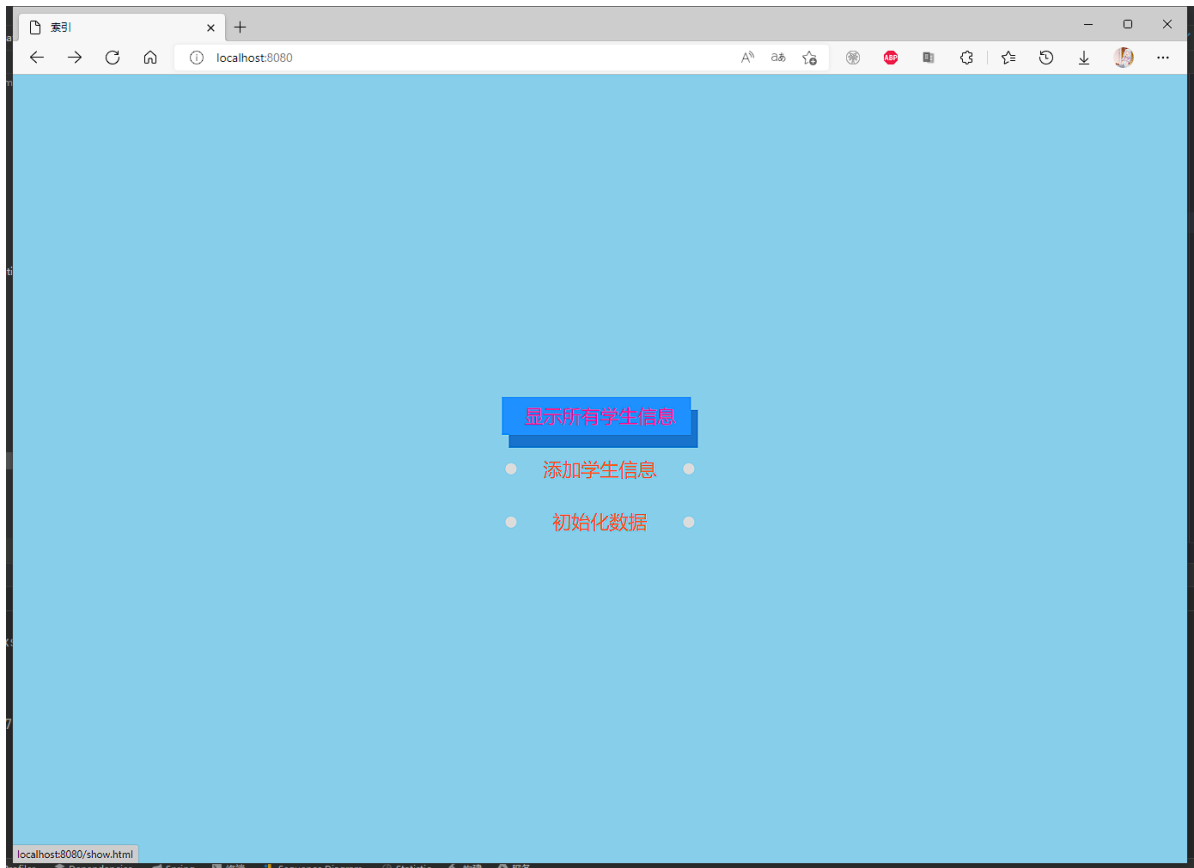
2022-10-30 13:20:06.888 INFO 17656 --- [nio-8080-exec-3] m.a.controller.StudentController : 添加成功:
id: 100009
name: <script>alert("xss攻击")</script>
sex: 男
age: 17
```





```
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



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

20 ms 40 ms 60 ms 80 ms 100 ms 120 ms 140 ms

名称 标头 预览 响应 发起程序 计时 Cookie

show.html  
axios.js  
vue.js  
student

↑: 0.00 KB ↓: 0.36 KB  
CPU: 3% 内存: 59%

```
▼ [{id: 10001, name: "张三", sex: "男", age: 18}, {id: 10002, name: "李四", sex: "女", age: 16}, {id: 10003, name: "王五", sex: "男", age: 20}, {id: 100002, name: "zsf", sex: "男", age: 16}, {id: 100009, name: "<script>alert('xss攻击')</script>", sex: "男", age: 17}, {id: 100009, name: "<script>alert('xss攻击')</script>", sex: "男", age: 17}, {id: 100012, name: "<h1>xss攻击</h1>", sex: "男", age: 19}, {id: 100012, name: "<h1>xss攻击</h1>", sex: "男", age: 19}]\n0: {id: 10001, name: "张三", sex: "男", age: 18}\n1: {id: 10002, name: "李四", sex: "女", age: 16}\n2: {id: 10003, name: "王五", sex: "男", age: 20}\n3: {id: 100002, name: "zsf", sex: "男", age: 16}\n4: {id: 100009, name: "<script>alert('xss攻击')</script>", sex: "男", age: 17}\n  age: 17\n  id: 100009\n  name: "<script>alert('xss攻击')</script>"\n  sex: "男"\n5: {id: 100012, name: "<h1>xss攻击</h1>", sex: "男", age: 19}\n  age: 19\n  id: 100012\n  name: "<h1>xss攻击</h1>"\n  sex: "男"
```

4 次请求 已传输 755 B 565 kB 资源

控制台 问题 搜索 +

top 筛选器 默认级别 5

Download the Vue Devtools extension for a better development experience: <https://github.com/vuejs/vue-devtools> vue.js:11291

You are running Vue in development mode. Make sure to turn on production mode when deploying for production. See more tips at <https://vuejs.org/guide/deployment.html> vue.js:11301

show.html:72

```
▼ {data: Array(6), status: 200, statusText: '', headers: {...}, config: {...}, ...}\n  config: {transitional: {...}, transformRequest: Array(1), transformResponse: Array(1), timeout: 0, adapter: ...}\n  data: Array(6)\n    0: {__ob__: Observer}\n    1: {__ob__: Observer}\n    2: {__ob__: Observer}\n    3: {__ob__: Observer}\n    4: {__ob__: Observer}\n    5: {__ob__: Observer}
```



## 解决xss攻击

### 第一步：添加依赖

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

```

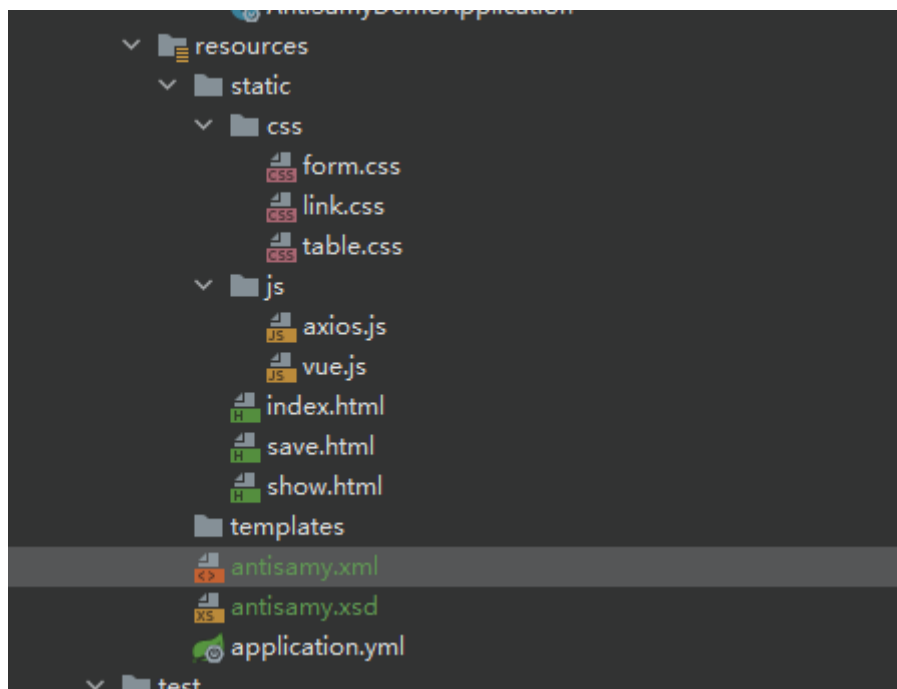
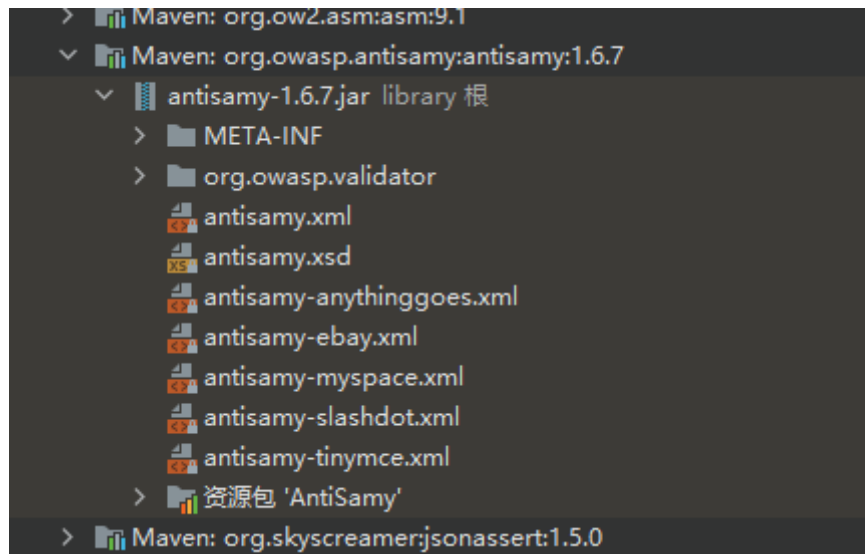
6      <groupId>org.springframework.boot</groupId>
7      <artifactId>spring-boot-starter-parent</artifactId>
8      <version>2.7.1</version>
9      <relativePath/> <!-- lookup parent from repository -->
10     </parent>
11     <groupId>mao</groupId>
12     <artifactId>antiSamy_demo</artifactId>
13     <version>0.0.1-SNAPSHOT</version>
14     <name>antiSamy_demo</name>
15     <description>antiSamy_demo</description>
16     <properties>
17         <java.version>11</java.version>
18     </properties>
19     <dependencies>
20         <dependency>
21             <groupId>org.springframework.boot</groupId>
22             <artifactId>spring-boot-starter-web</artifactId>
23         </dependency>
24
25         <dependency>
26             <groupId>org.springframework.boot</groupId>
27             <artifactId>spring-boot-starter-test</artifactId>
28             <scope>test</scope>
29         </dependency>
30
31         <!--解决xss攻击-->
32         <dependency>
33             <groupId>org.owasp.antisamy</groupId>
34             <artifactId>antisamy</artifactId>
35             <version>1.5.7</version>
36         </dependency>
37
38     </dependencies>
39
40     <build>
41         <plugins>
42             <plugin>
43                 <groupId>org.springframework.boot</groupId>
44                 <artifactId>spring-boot-maven-plugin</artifactId>
45             </plugin>
46         </plugins>
47     </build>
48
49 </project>
50

```

## 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，包含几个常用的策略文件



### 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

```
1 package mao.antisamy_demo.filter;
2
3 import javax.servlet.*;
4 import javax.servlet.http.HttpServletRequest;
5 import java.io.IOException;
6
7 /**
8  * Project name(项目名称): antiSamy_demo
9  * Package(包名): mao.antisamy_demo.filter
10 * Class(类名): XssFilter
11 * Author(作者): mao
12 * Author QQ: 1296193245
13 * GitHub: https://github.com/maomao124/
14 * Date(创建日期): 2022/10/30
15 * Time(创建时间): 13:44
16 * Version(版本): 1.0
17 * Description(描述): 无
18 */
19
20 public class XssFilter implements Filter
21 {
22
23     @Override
24     public void doFilter(ServletRequest servletRequest, ServletResponse
servletResponse, FilterChain filterChain)
```

```

25         throws IOException, ServletException
26     {
27         HttpServletRequest request = (HttpServletRequest) servletRequest;
28         //传入重写后的Request
29         filterChain.doFilter(new XssRequestWrapper(request),
servletResponse);
30     }
31 }

```

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

```

1  package mao.antisamy_demo.wrapper;
2
3  import org.owasp.validator.html.*;
4  import org.slf4j.Logger;
5  import org.slf4j.LoggerFactory;
6
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletRequestWrapper;
9  import java.io.InputStream;
10
11
12  /**
13   * Project name(项目名称): antiSamy_demo
14   * Package(包名): mao.antisamy_demo.wrapper
15   * Class(类名): XssRequestWrapper
16   * Author(作者): mao
17   * Author QQ: 1296193245
18   * GitHub: https://github.com/maomao124/
19   * Date(创建日期): 2022/10/30
20   * Time(创建时间): 13:46
21   * Version(版本): 1.0
22   * Description(描述): 无
23   */
24
25  public class XssRequestWrapper extends HttpServletRequestWrapper
26  {
27      /**
28       * 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

```

```

29     */
30     @SuppressWarnings("all")
31     private static final String antiSamyPath = "antisamy.xml";
32
33     public static Policy policy = null;
34
35     private static final Logger log =
36     LoggerFactory.getLogger(XssRequestWrapper.class);
37
38     static
39     {
40         // 指定策略文件
41         try
42         {
43             InputStream inputStream =
44             XssRequestWrapper.class.getClassLoader().getResourceAsStream(antiSamyPath);
45             assert inputStream != null;
46             policy = Policy.getInstance(inputStream);
47         }
48         catch (PolicyException e)
49         {
50             e.printStackTrace();
51         }
52     }
53
54     /**
55     * AntiSamy过滤数据
56     *
57     * @param taintedHTML 需要进行过滤的数据
58     * @return 返回过滤后的数据
59     */
60     private String xssClean(String taintedHTML)
61     {
62         try
63         {
64             // 使用AntiSamy进行过滤
65             AntiSamy antiSamy = new AntiSamy();
66             CleanResults cleanResults = antiSamy.scan(taintedHTML, policy);
67             taintedHTML = cleanResults.getCleanHTML();
68         }
69         catch (ScanException | PolicyException e)
70         {
71             e.printStackTrace();
72         }
73         return taintedHTML;
74     }
75
76     public XssRequestWrapper(HttpServletRequest request)
77     {
78         super(request);
79     }
80
81     @Override
82     public String[] getParameterValues(String name)
83     {
84         String[] values = super.getParameterValues(name);
85         if (values == null)
86         {

```

```

85         return null;
86     }
87     int len = values.length;
88     String[] newArray = new String[len];
89     for (int j = 0; j < len; j++)
90     {
91         log.info("Antisamy过滤清理, 清理之前的参数值: " + values[j]);
92         // 过滤清理
93         newArray[j] = xssClean(values[j]);
94         log.info("Antisamy过滤清理, 清理之后的参数值: " + newArray[j]);
95     }
96     return newArray;
97 }
98 }

```

## 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

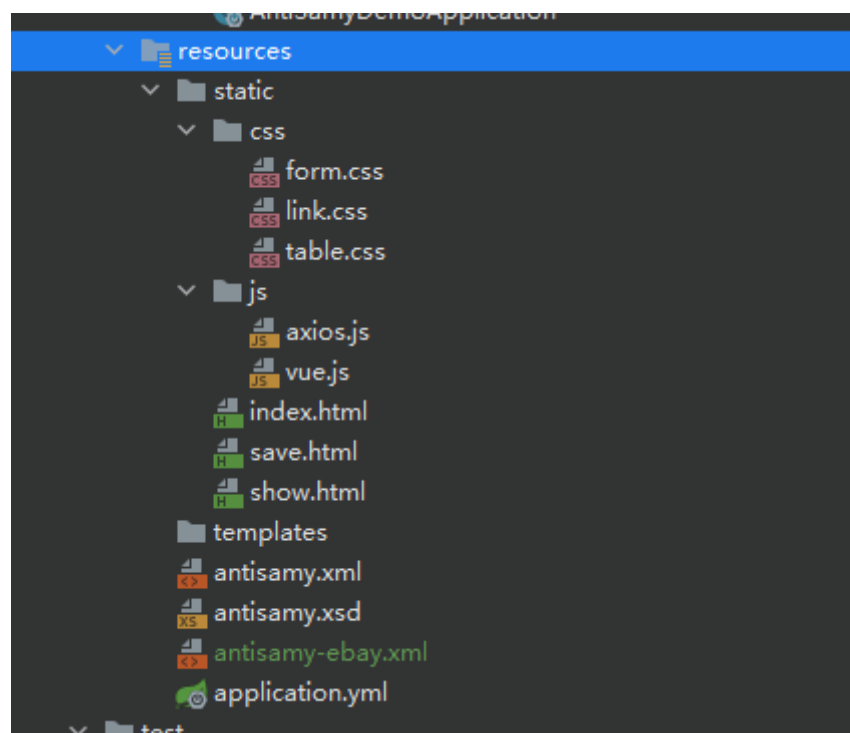
```

1  package mao.antisamy_demo.config;
2
3  import mao.antisamy_demo.filter.XssFilter;
4  import org.springframework.boot.web.servlet.FilterRegistrationBean;
5  import org.springframework.context.annotation.Bean;
6  import org.springframework.context.annotation.Configuration;
7
8  /**
9   * Project name(项目名称): antiSamy_demo
10  * Package(包名): mao.antisamy_demo.config
11  * Class(类名): AntiSamyConfig
12  * Author(作者): mao
13  * Author QQ: 1296193245
14  * GitHub: https://github.com/maomao124/
15  * Date(创建日期): 2022/10/30
16  * Time(创建时间): 13:53
17  * Version(版本): 1.0
18  * Description(描述): 无
19  */
20
21 @Configuration
22 public class AntiSamyConfig
23 {
24     @Bean
25     public FilterRegistrationBean<XssFilter> filterRegistrationBean()
26     {
27         FilterRegistrationBean<XssFilter> filterRegistrationBean = new
28         FilterRegistrationBean<>(new XssFilter());
29         filterRegistrationBean.addUrlPatterns("/");
30         filterRegistrationBean.setOrder(1);
31         return filterRegistrationBean;
32     }
33 }

```

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

```
1 package mao.antisamy_demo.wrapper;  
2  
3 import org.owasp.validator.html.*;  
4 import org.slf4j.Logger;  
5 import org.slf4j.LoggerFactory;  
6  
7 import javax.servlet.http.HttpServletRequest;  
8 import javax.servlet.http.HttpServletRequestWrapper;  
9 import java.io.InputStream;
```

```

10 import java.util.Map;
11
12
13 /**
14  * Project name(项目名称): antiSamy_demo
15  * Package(包名): mao.antisamy_demo.wrapper
16  * Class(类名): XssRequestWrapper
17  * Author(作者) : mao
18  * Author QQ: 1296193245
19  * GitHub: https://github.com/maomao124/
20  * Date(创建日期): 2022/10/30
21  * Time(创建时间): 13:46
22  * Version(版本): 1.0
23  * Description(描述): 无
24  */
25
26 public class XssRequestWrapper extends HttpServletRequestWrapper
27 {
28     /**
29      * 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
30      */
31     @SuppressWarnings("all")
32     private static final String antiSamyPath = "antisamy-ebay.xml";
33
34     public static Policy policy = null;
35
36     private static final Logger log =
37     LoggerFactory.getLogger(XssRequestWrapper.class);
38
39     static
40     {
41         // 指定策略文件
42         try
43         {
44             InputStream inputStream =
45             XssRequestWrapper.class.getClassLoader().getResourceAsStream(antiSamyPath);
46             assert inputStream != null;
47             policy = Policy.getInstance(inputStream);
48         }
49         catch (PolicyException e)
50         {
51             e.printStackTrace();
52         }
53     }
54
55     /**
56      * AntiSamy过滤数据
57      *
58      * @param taintedHTML 需要进行过滤的数据
59      * @return 返回过滤后的数据
60      */
61     private String xssClean(String taintedHTML)
62     {
63         try
64         {
65             // 使用AntiSamy进行过滤
66             AntiSamy antiSamy = new AntiSamy();
67             CleanResults cr = antiSamy.scan(taintedHTML, policy);

```

```

66         taintedHTML = cr.getCleanHTML();
67     }
68     catch (ScanException | PolicyException e)
69     {
70         e.printStackTrace();
71     }
72     return taintedHTML;
73 }
74
75 public XssRequestWrapper(HttpServletRequest request)
76 {
77     super(request);
78 }
79
80 @Override
81 public String[] getParameterValues(String name)
82 {
83     String[] values = super.getParameterValues(name);
84     if (values == null)
85     {
86         return null;
87     }
88     int len = values.length;
89     String[] newArray = new String[len];
90     for (int j = 0; j < len; j++)
91     {
92         // 过滤清理
93         newArray[j] = xssClean(values[j]);
94         log.debug("Antisamy过滤清理, 清理之前的参数值: " + values[j]);
95         log.debug("Antisamy过滤清理, 清理之后的参数值: " + newArray[j]);
96     }
97     return newArray;
98 }
99
100 @Override
101 public String getParameter(String paramString)
102 {
103     String str = super.getParameter(paramString);
104     if (str == null)
105     {
106         return null;
107     }
108     return xssClean(str);
109 }
110
111
112 @Override
113 public String getHeader(String paramString)
114 {
115     String str = super.getHeader(paramString);
116     if (str == null)
117     {
118         return null;
119     }
120     return xssClean(str);
121 }
122
123 @Override

```

```
124 public Map<String, String[]> getParameterMap()  
125 {  
126     Map<String, String[]> requestMap = super.getParameterMap();  
127     for (Map.Entry<String, String[]> stringEntry :  
requestMap.entrySet())  
128     {  
129         String[] values = stringEntry.getValue();  
130         for (int i = 0; i < values.length; i++)  
131         {  
132             values[i] = xssClean(values[i]);  
133         }  
134     }  
135     return requestMap;  
136 }  
137 }
```

## 第八步：启动程序

[illegible]



```
17 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]
18 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 ''
19 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	<input type="text" value="111"/>
学生姓名	<input type="text" value="12&lt;script&gt;alert('xss攻击')&lt;/script&gt;"/>
学生性别	<input type="text" value="男"/>
学生年龄	<input type="text" value="8"/>
<input type="button" value="提交"/>	

```
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

```
1 package mao.antisamy_demo.controller;
2
3 import mao.antisamy_demo.entity.Student;
4 import mao.antisamy_demo.wrapper.XssRequestWrapper;
5 import org.apache.juli.logging.Log;
6 import org.slf4j.Logger;
7 import org.slf4j.LoggerFactory;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.web.bind.annotation.*;
10
11 import java.util.ArrayList;
12 import java.util.Collections;
13 import java.util.List;
14
15 /**
16  * Project name(项目名称): antiSamy_demo
17  * Package(包名): mao.antisamy_demo.controller
18  * Class(类名): StudentController
19  * Author(作者): mao
20  * Author QQ: 1296193245
21  * GitHub: https://github.com/maomao124/
22  * Date(创建日期): 2022/10/29
23  * Time(创建时间): 20:05
24  * Version(版本): 1.0
25  * Description(描述): 无
26  */
27
28 @RestController
29 @RequestMapping("/student")
30 public class StudentController
31 {
32     private static final List<Student> list =
33     Collections.synchronizedList(new ArrayList<>());
34
35     private static final Logger log =
36     LoggerFactory.getLogger(StudentController.class);
37
38     @PostMapping("/init")
39     public synchronized void init()
40     {
41         Student student1 = new Student(10001, "张三", "男", 18);
42         Student student2 = new Student(10002, "李四", "女", 16);
43         Student student3 = new Student(10003, "王五", "男", 20);
44         list.clear();
45     }
46 }
```

```

44         list.add(student1);
45         list.add(student2);
46         list.add(student3);
47         log.info("初始化完成");
48     }
49
50     @PostMapping
51     public boolean save(@RequestBody Student student)
52     {
53         list.add(student);
54         log.info("添加成功: \n" + student);
55         return true;
56     }
57
58     @PostMapping("/sync")
59     public boolean saveSync(Student student)
60     {
61         list.add(student);
62         log.info("添加成功: \n" + student);
63         return true;
64     }
65
66     @GetMapping
67     public List<Student> getAll()
68     {
69         log.info("查询所有: \n" + list);
70         return list;
71     }
72
73 }

```

## 第十一步：编写save2.html

```

1  <!DOCTYPE html>
2
3  <!--
4  Project name(项目名称): antiSamy_demo
5  File name(文件名): save
6  Authors(作者): mao
7  Author QQ: 1296193245
8  GitHub: https://github.com/maomao124/
9  Date(创建日期): 2022/10/29
10 Time(创建时间): 20:23
11 Description(描述): 无
12 -->
13
14 <html lang="en">
15 <head>

```

```

16     <meta charset="UTF-8">
17     <title>Title</title>
18     <link rel="stylesheet" href="css/form.css">
19     <script src="js/axios.js"></script>
20     <script src="js/vue.js"></script>
21     <style>
22         body {
23             background-color: skyblue;
24         }
25     </style>
26 </head>
27 <body>
28
29 <div id="app">
30
31
32     <div class="form_position">
33         <div class="animated bounceInDown">
34             <div class="form">
35                 <form action="/student/sync" method="post">
36                     <table border="1">
37                         <tr>
38                             <td colspan="2" align="center">
39                                 </td>
40                             </tr>
41                         <tr>
42                             <td class="prompt">学生学号</td>
43                             <td>
44                                 <label>
45                                     <input v-model="student.id"
46                                     class="input" type="number" name="id">
47                                     </label>
48                                 </td>
49                             </tr>
50                         <tr>
51                             <td class="prompt">学生姓名</td>
52                             <td>
53                                 <label>
54                                     <input v-model="student.name"
55                                     class="input" type="text" name="name">
56                                     </label>
57                                 </td>
58                             </tr>
59                         <tr>
60                             <td class="prompt">学生性别</td>
61                             <td>
62                                 <label>
63                                     <input v-model="student.sex"
64                                     class="input" type="text" name="sex">
65                                     </label>
66                                 </td>
67                             </tr>
68                         <tr>
69                             <td class="prompt">学生年龄</td>
70                             <td>
71                                 <label>
72                                     <input v-model="student.age"
73                                     class="input" type="number" name="age">

```

```

70         </label>
71     </td>
72 </tr>
73 </tr>
74     <td colspan="2" align="center">
75         <input @click="f()" class="submit"
type="submit" value="提交"/>
76     </td>
77 </tr>
78 </table>
79 </form>
80 </div>
81 </div>
82 </div>
83
84 </div>
85
86 <!--<script>alert("xss攻击")</script>-->
87 <!--<h1>xss攻击</h1>-->
88
89 <script>
90
91     // var app = new vue(
92     //     {
93     //         el: "#app",
94     //         data: {
95     //             student: {
96     //                 id: null,
97     //                 name: null,
98     //                 sex: null,
99     //                 age: null,
100            //     }
101        //     },
102        //     methods: {
103        //         f: function ()
104        //         {
105        //             console.log("提交")
106        //             console.log(this.student)
107        //             var that = this;
108        //             //axios发起ajax请求
109        //             axios({
110        //                 //请求的方式:
111        //                 method: "post",
112        //                 //请求的url:
113        //                 url: "/student",
114        //                 //url参数:
115        //                 params:
116        //                 {
117        //                     //头信息:
118        //                     headers:
119        //                     {
120        //                         //请求体参数:
121        //                         data:
122        //                         {
123        //                             id: that.student.id,
124        //                             name: that.student.name,
125        //                             sex: that.student.sex,
126        //                             age: that.student.age,

```

```

127         //         },
128         //         }).then(response =>
129         //         {
130         //             console.log(response);
131         //             if (response.data === true)
132         //             {
133         //                 alert("请求成功")
134         //             }
135         //             else
136         //             {
137         //                 alert("失败")
138         //             }
139         //         })
140         //         }).catch(error =>
141         //         {
142         //             //console.log(error);
143         //             alert("网络异常! ");
144         //         })
145         //     }
146     }
147 }
148 // )
149
150 </script>
151
152 </body>
153 </html>

```

## 第十二步：修改index.html

```

1  <!DOCTYPE html>
2
3  <!--
4  Project name(项目名称): antiSamy_demo
5  File name(文件名): index
6  Authors(作者): mao
7  Author QQ: 1296193245
8  Github: https://github.com/maomao124/
9  Date(创建日期): 2022/10/29
10 Time(创建时间): 20:03
11 Description(描述): 无
12 -->
13
14 <html lang="en">
15 <head>
16     <meta charset="UTF-8">
17     <title>索引</title>
18     <link rel="stylesheet" href="/css/link.css">
19     <script src="/js/axios.js"></script>
20

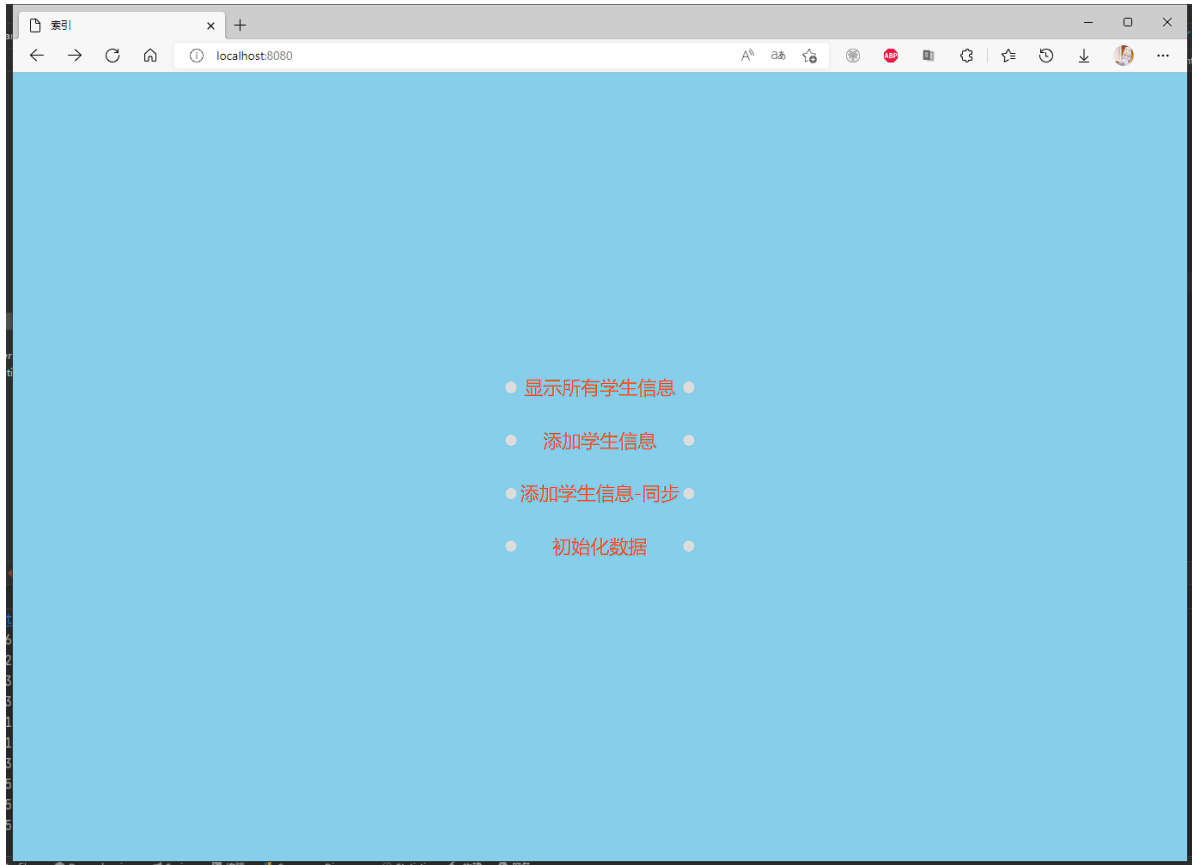
```

```

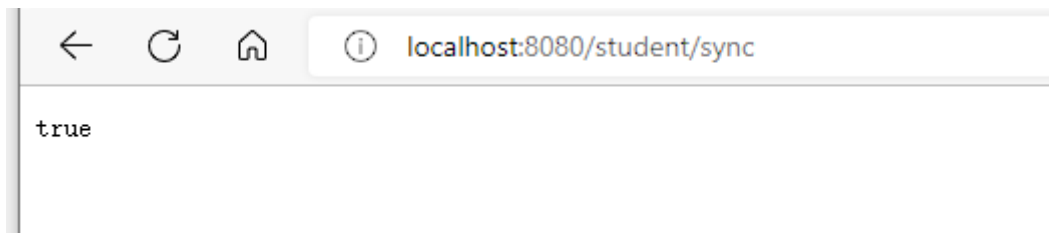
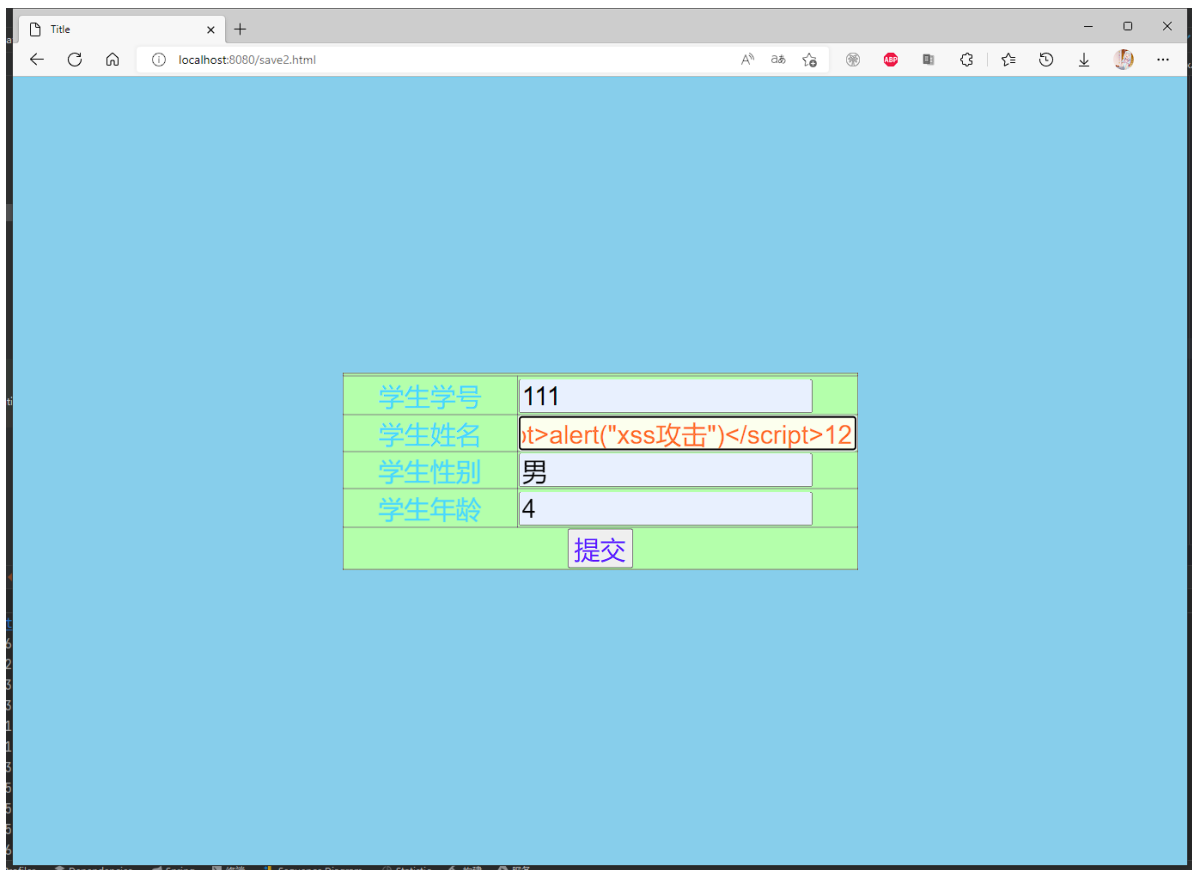
21
22     <style>
23         body {
24             background-color: skyblue;
25         }
26
27         div.a {
28             position: absolute;
29             top: 50%;
30             left: 50%;
31             transform: translate(-50%, -50%);
32         }
33     </style>
34 </head>
35 <body>
36
37 <div class="a">
38
39     <a href="/show.html">显示所有学生信息</a>
40     <a href="/save.html">添加学生信息</a>
41     <a href="/save2.html">添加学生信息-同步</a>
42     <a href="" onclick="init()">初始化数据</a>
43
44 </div>
45
46 <script>
47
48     function init()
49     {
50         //axios发起ajax请求
51         axios({
52             //请求的方式:
53             method: "post",
54             //请求的url:
55             url: "/student/init",
56             //url参数:
57             params:
58                 {},
59             //头信息:
60             headers:
61                 {},
62             //请求体参数:
63             data:
64                 {},
65         }).then(response =>
66         {
67             console.log(response);
68             alert("初始化数据成功");
69         }).catch(error =>
70         {
71             //console.log(error);
72             alert("网络异常! ");
73         })
74     }
75
76 </script>
77
78 </body>

```

### 第十三步：重启并访问



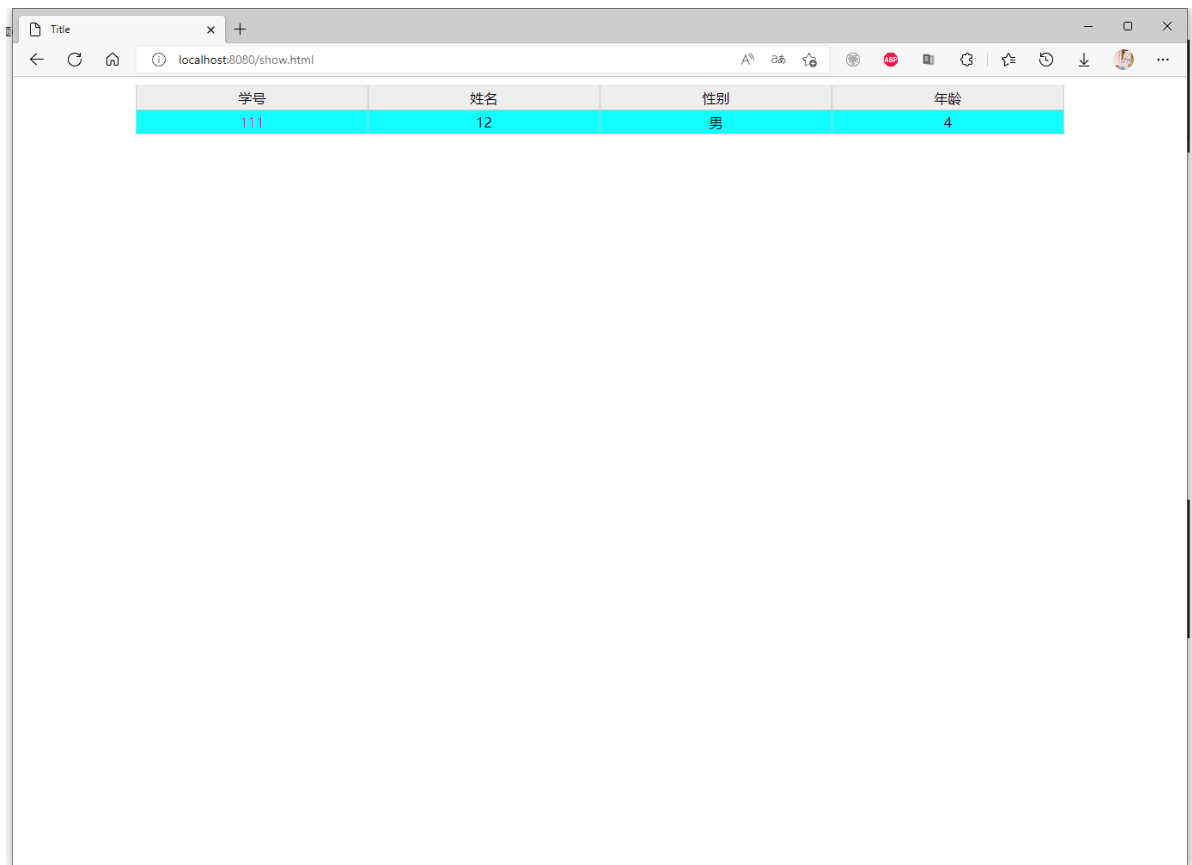




#### 第十四步：查看控制台日志

```
2022-10-30 15:51:59.578 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



以下是解决方案2

## 第十五步：编写XssFilterService

```
1 package mao.antisamy_demo.service;
2
3 import mao.antisamy_demo.wrapper.XssRequestWrapper;
4 import org.owasp.validator.html.*;
5 import org.slf4j.Logger;
6 import org.slf4j.LoggerFactory;
7 import org.springframework.stereotype.Service;
8
9 import java.io.InputStream;
10
11 /**
12  * Project name(项目名称): antisamy_demo
13  * Package(包名): mao.antisamy_demo.service
14  * Class(类名): XssFilterService
15  * Author(作者): mao
16  * Author QQ: 1296193245
17  * Github: https://github.com/maomao124/
```

```

18  * Date(创建日期): 2022/10/30
19  * Time(创建时间): 15:43
20  * Version(版本): 1.0
21  * Description(描述): 无
22  */
23
24  @Service
25  public class XssFilterService
26  {
27      /**
28       * 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
29       */
30      @SuppressWarnings("all")
31      private static final String antiSamyPath = "antisamy-ebay.xml";
32
33      public static Policy policy = null;
34
35      private static final Logger log =
36      LoggerFactory.getLogger(XssRequestWrapper.class);
37
38      static
39      {
40          // 指定策略文件
41          try
42          {
43              InputStream inputStream =
44              XssRequestWrapper.class.getClassLoader().getResourceAsStream(antiSamyPath);
45              assert inputStream != null;
46              policy = Policy.getInstance(inputStream);
47          }
48          catch (PolicyException e)
49          {
50              e.printStackTrace();
51          }
52      }
53
54      /**
55       * AntiSamy过滤数据
56       *
57       * @param taintedHTML 需要进行过滤的数据
58       * @return 返回过滤后的数据
59       */
60      public String xssClean(String taintedHTML)
61      {
62          try
63          {
64              // 使用AntiSamy进行过滤
65              AntiSamy antiSamy = new AntiSamy();
66              CleanResults cleanResults = antiSamy.scan(taintedHTML, policy);
67              taintedHTML = cleanResults.getCleanHTML();
68          }
69          catch (ScanException | PolicyException e)
70          {
71              e.printStackTrace();
72          }
73          return taintedHTML;
74      }
75  }

```

## 第十六步：修改StudentController

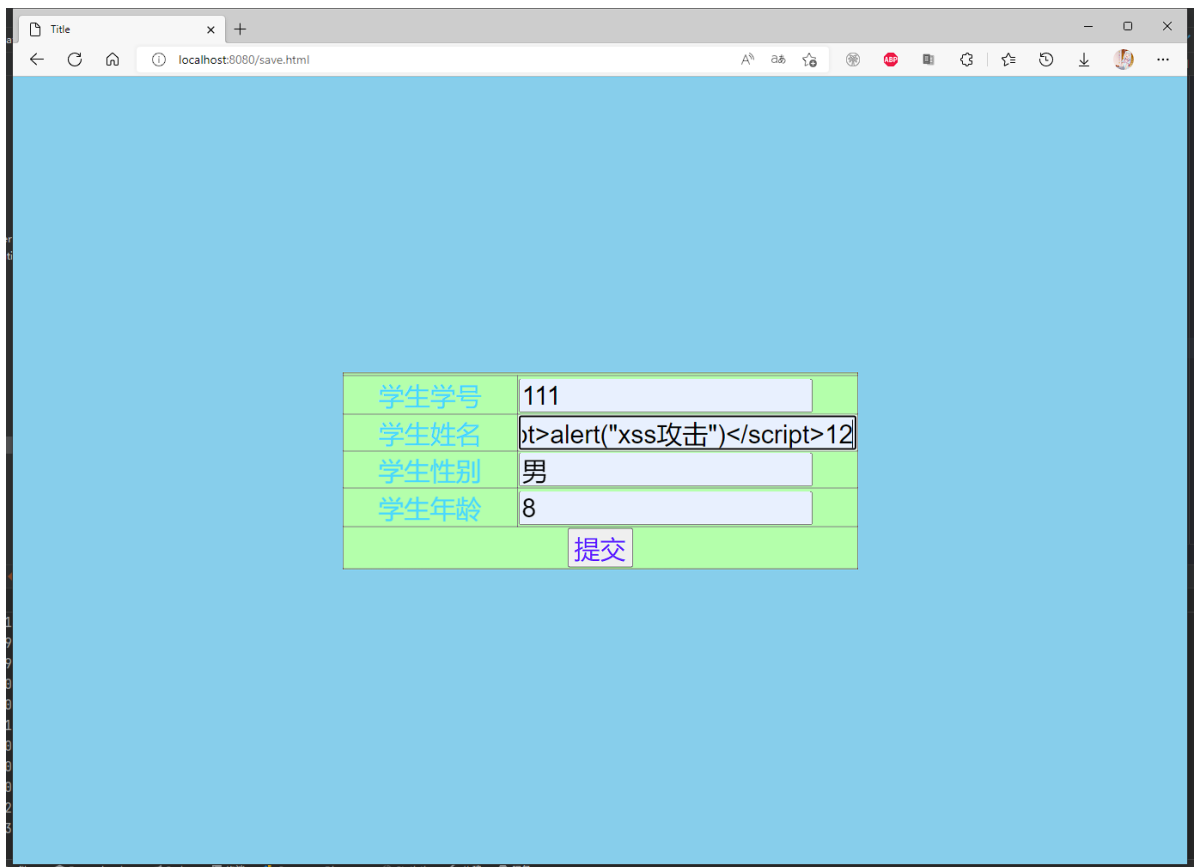
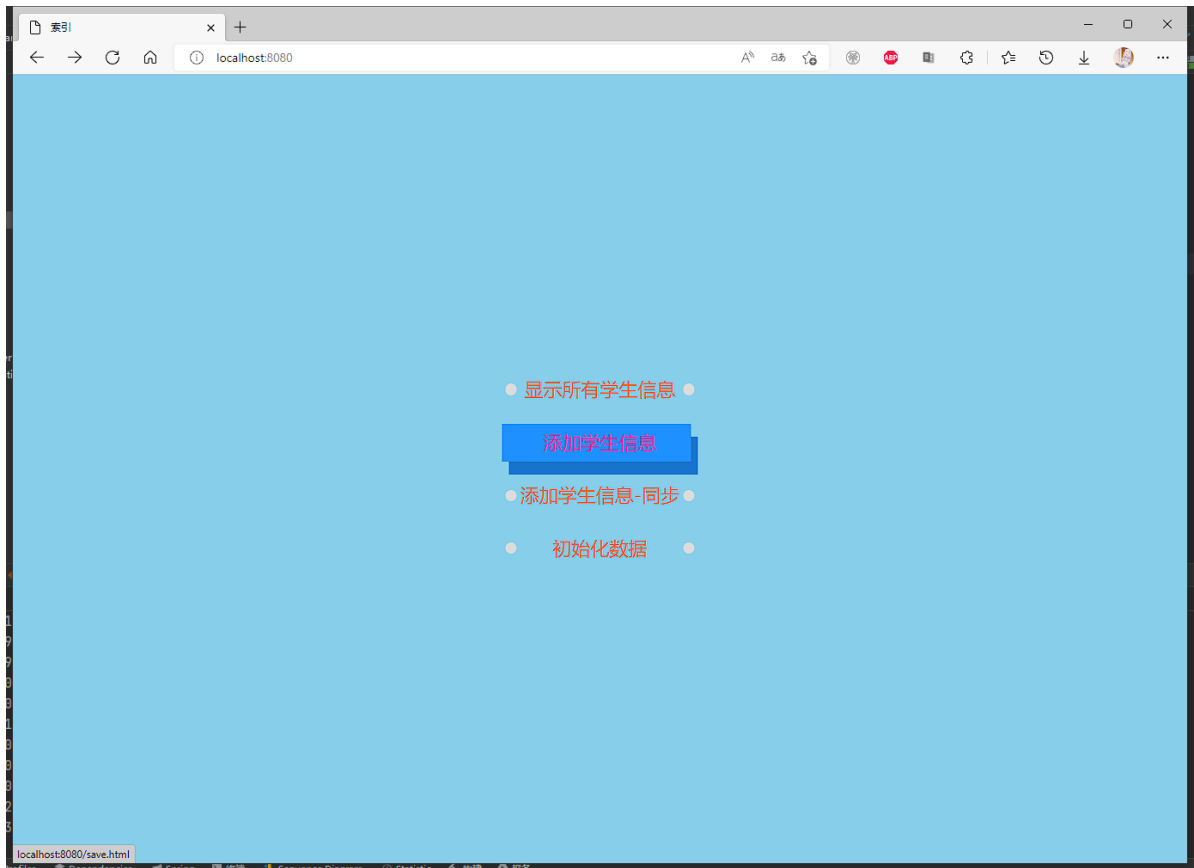
```
1 package mao.antisamy_demo.controller;
2
3 import mao.antisamy_demo.entity.Student;
4 import mao.antisamy_demo.service.XssFilterService;
5 import mao.antisamy_demo.wrapper.XssRequestWrapper;
6 import org.apache.juli.logging.Log;
7 import org.slf4j.Logger;
8 import org.slf4j.LoggerFactory;
9 import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.web.bind.annotation.*;
11
12 import java.util.ArrayList;
13 import java.util.Collections;
14 import java.util.List;
15
16 /**
17  * Project name(项目名称): antiSamy_demo
18  * Package(包名): mao.antisamy_demo.controller
19  * Class(类名): StudentController
20  * Author(作者): mao
21  * Author QQ: 1296193245
22  * GitHub: https://github.com/maomao124/
23  * Date(创建日期): 2022/10/29
24  * Time(创建时间): 20:05
25  * Version(版本): 1.0
26  * Description(描述): 无
27  */
28
29 @RestController
30 @RequestMapping("/student")
31 public class StudentController
32 {
33     private static final List<Student> list =
34         Collections.synchronizedList(new ArrayList<>());
35
36     private static final Logger log =
37         LoggerFactory.getLogger(StudentController.class);
38
39     @Autowired
40     private XssFilterService xssFilterService;
41
42     @PostMapping("/init")
43     public synchronized void init()
44     {
45         Student student1 = new Student(10001, "张三", "男", 18);
46         Student student2 = new Student(10002, "李四", "女", 16);
47         Student student3 = new Student(10003, "王五", "男", 20);
48         list.clear();
49         list.add(student1);
```

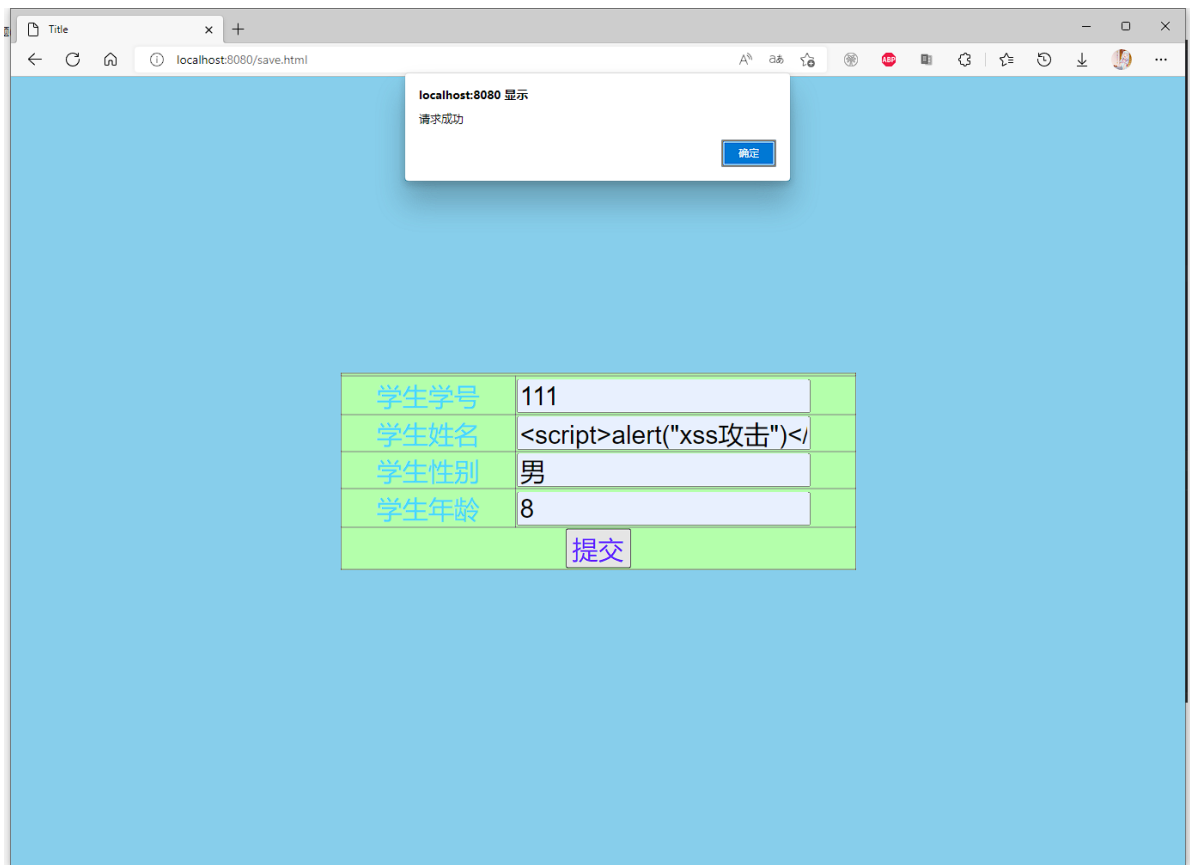
```

48         list.add(student2);
49         list.add(student3);
50         log.info("初始化完成");
51     }
52
53     // @PostMapping
54     // public boolean save(@RequestBody Student student)
55     // {
56     //     list.add(student);
57     //     log.info("添加成功: \n" + student);
58     //     return true;
59     // }
60
61
62     @PostMapping
63     public boolean save(@RequestBody Student student)
64     {
65         student.setName(xssFilterService.xssClean(student.getName()));
66         student.setSex(xssFilterService.xssClean(student.getSex()));
67
68         list.add(student);
69         log.info("添加成功: \n" + student);
70         return true;
71     }
72
73
74     @PostMapping("/sync")
75     public boolean saveSync(Student student)
76     {
77         list.add(student);
78         log.info("添加成功: \n" + student);
79         return true;
80     }
81
82     @GetMapping
83     public List<Student> getAll()
84     {
85         log.info("查询所有: \n" + list);
86         return list;
87     }
88
89 }

```

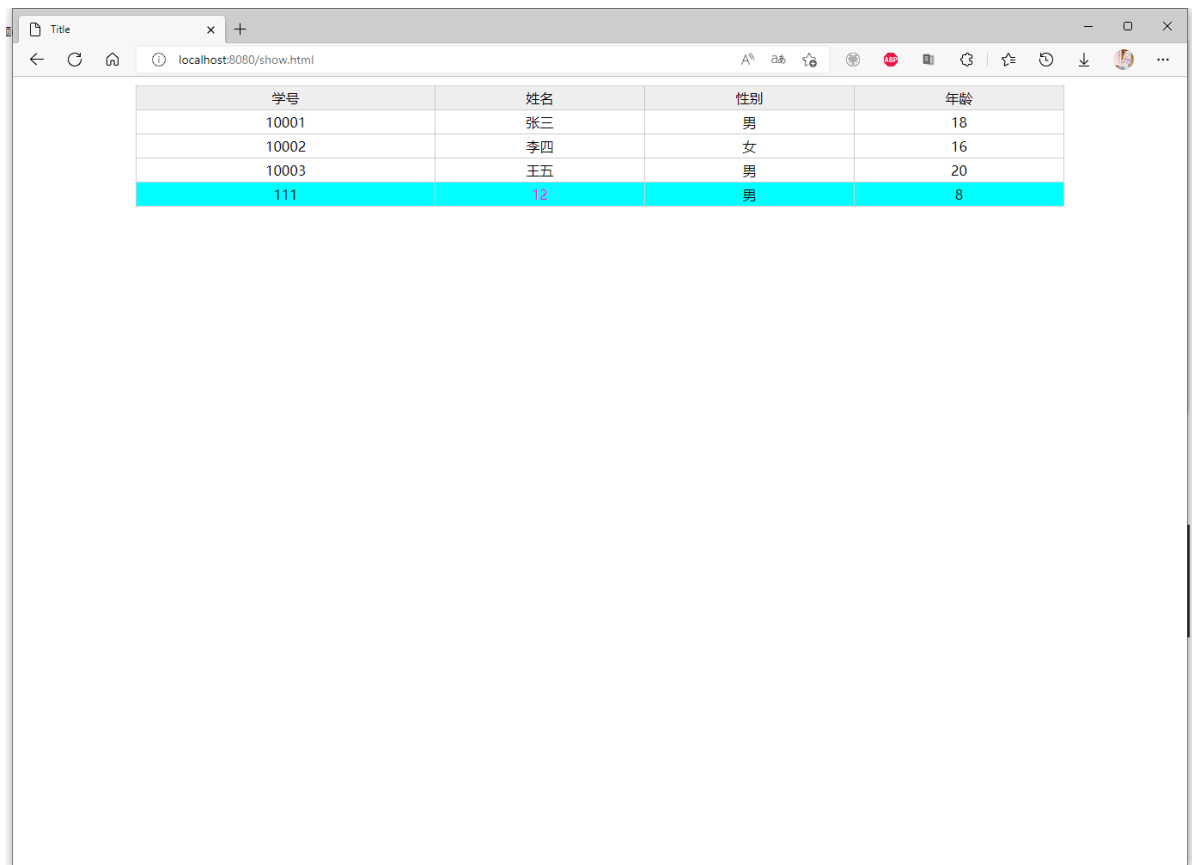
## 第十七步：重启并访问





```
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



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

以下是解决方案3

## 第十八步：编写类XssStringJsonDeserializer

```
1 package mao.antisamy_demo.converter;
2
3 import com.fasterxml.jackson.core.JsonParser;
4 import com.fasterxml.jackson.core.JsonProcessingException;
5 import com.fasterxml.jackson.core.JsonToken;
6 import com.fasterxml.jackson.databind.DeserializationContext;
7 import com.fasterxml.jackson.databind.JsonDeserializer;
8 import mao.antisamy_demo.service.XssFilterService;
9
10 import java.io.IOException;
11 import java.util.ArrayList;
12 import java.util.List;
13
14 /**
15  * Project name(项目名称): AntiSamy_spring_boot_starter_demo
16  * Package(包名): mao.tools_xss.converter
17  * Class(类名): XssStringJsonDeserializer
```



```

18  * Author(作者): mao
19  * Author QQ: 1296193245
20  * GitHub: https://github.com/maomao124/
21  * Date(创建日期): 2022/10/30
22  * Time(创建时间): 20:14
23  * Version(版本): 1.0
24  * Description(描述): 过滤跨站脚本的 反序列化工具
25  */
26
27  public class XssStringJsonDeserializer extends JsonSerializer<String>
28  {
29
30      private final XssFilterService xssFilterService;
31
32      public XssStringJsonDeserializer(XssFilterService xssFilterService)
33      {
34          this.xssFilterService = xssFilterService;
35      }
36
37      @Override
38      public String deserialize(JsonParser p, DeserializationContext dc)
39      throws IOException, JsonProcessingException
40      {
41          if (p.hasToken(JsonToken.VALUE_STRING))
42          {
43              String value = p.getValueAsString();
44
45              if (value == null || "".equals(value))
46              {
47                  return value;
48              }
49
50              List<String> list = new ArrayList<>();
51              list.add("<script>");
52              list.add("</script>");
53              list.add("<iframe>");
54              list.add("</iframe>");
55              list.add("<noscript>");
56              list.add("</noscript>");
57              list.add("<frameset>");
58              list.add("</frameset>");
59              list.add("<frame>");
60              list.add("</frame>");
61              list.add("<noframes>");
62              list.add("</noframes>");
63              boolean flag = list.stream().anyMatch(value::contains);
64              if (flag)
65              {
66                  return xssFilterService.xssClean(value);
67              }
68              return value;
69          }
70          return null;
71      }

```

## 第十九步：修改配置类AntiSamyConfig

```
1 package mao.antisamy_demo.config;
2
3 import mao.antisamy_demo.converter.XssStringJsonDeserializer;
4 import mao.antisamy_demo.filter.XssFilter;
5 import mao.antisamy_demo.service.XssFilterService;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import
org.springframework.boot.autoconfigure.jackson.Jackson2ObjectMapperBuilderCu
stomizer;
8 import org.springframework.boot.web.servlet.FilterRegistrationBean;
9 import org.springframework.context.annotation.Bean;
10 import org.springframework.context.annotation.Configuration;
11
12 import javax.servlet.DispatcherType;
13
14 /**
15  * Project name(项目名称): antiSamy_demo
16  * Package(包名): mao.antisamy_demo.config
17  * Class(类名): AntiSamyConfig
18  * Author(作者): mao
19  * Author QQ: 1296193245
20  * GitHub: https://github.com/maomao124/
21  * Date(创建日期): 2022/10/30
22  * Time(创建时间): 13:53
23  * Version(版本): 1.0
24  * Description(描述): 无
25  */
26
27
28 @Configuration
29 public class AntiSamyConfig
30 {
31     @Bean
32     public FilterRegistrationBean<XssFilter> filterRegistrationBean()
33     {
34         FilterRegistrationBean<XssFilter> filterRegistration = new
FilterRegistrationBean<>(new XssFilter());
35         filterRegistration.addUrlPatterns("/");
36         filterRegistration.setDispatcherTypes(DispatcherType.REQUEST);
37         filterRegistration.setName("xssFilter");
38         filterRegistration.setOrder(1);
39
40         return filterRegistration;
41     }
42
43     /**
44      * 配置跨站攻击 反序列化处理器
45      *
46      * @return Jackson2ObjectMapperBuilderCustomizer
47      */
48     @Bean
```

```

49     public Jackson2ObjectMapperBuilderCustomizer
jackson2ObjectMapperBuilderCustomizer2(@Autowired XssFilterService
xssFilterService)
50     {
51         return builder -> builder.deserializerByType(String.class, new
XssStringJsonDeserializer(xssFilterService));
52     }
53
54 }

```

## 第二十步：修改StudentController

```

1  package mao.antisamy_demo.controller;
2
3  import mao.antisamy_demo.entity.Student;
4  import mao.antisamy_demo.service.XssFilterService;
5  import mao.antisamy_demo.wrapper.XssRequestWrapper;
6  import org.apache.juli.logging.Log;
7  import org.slf4j.Logger;
8  import org.slf4j.LoggerFactory;
9  import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.web.bind.annotation.*;
11
12 import java.util.ArrayList;
13 import java.util.Collections;
14 import java.util.List;
15
16 /**
17  * Project name(项目名称): antisamy_demo
18  * Package(包名): mao.antisamy_demo.controller
19  * Class(类名): StudentController
20  * Author(作者): mao
21  * Author QQ: 1296193245
22  * Github: https://github.com/maomao124/
23  * Date(创建日期): 2022/10/29
24  * Time(创建时间): 20:05
25  * Version(版本): 1.0
26  * Description(描述): 无
27  */
28
29 @RestController
30 @RequestMapping("/student")
31 public class StudentController
32 {
33     private static final List<Student> list =
Collections.synchronizedList(new ArrayList<>());
34
35     private static final Logger log =
LoggerFactory.getLogger(StudentController.class);
36

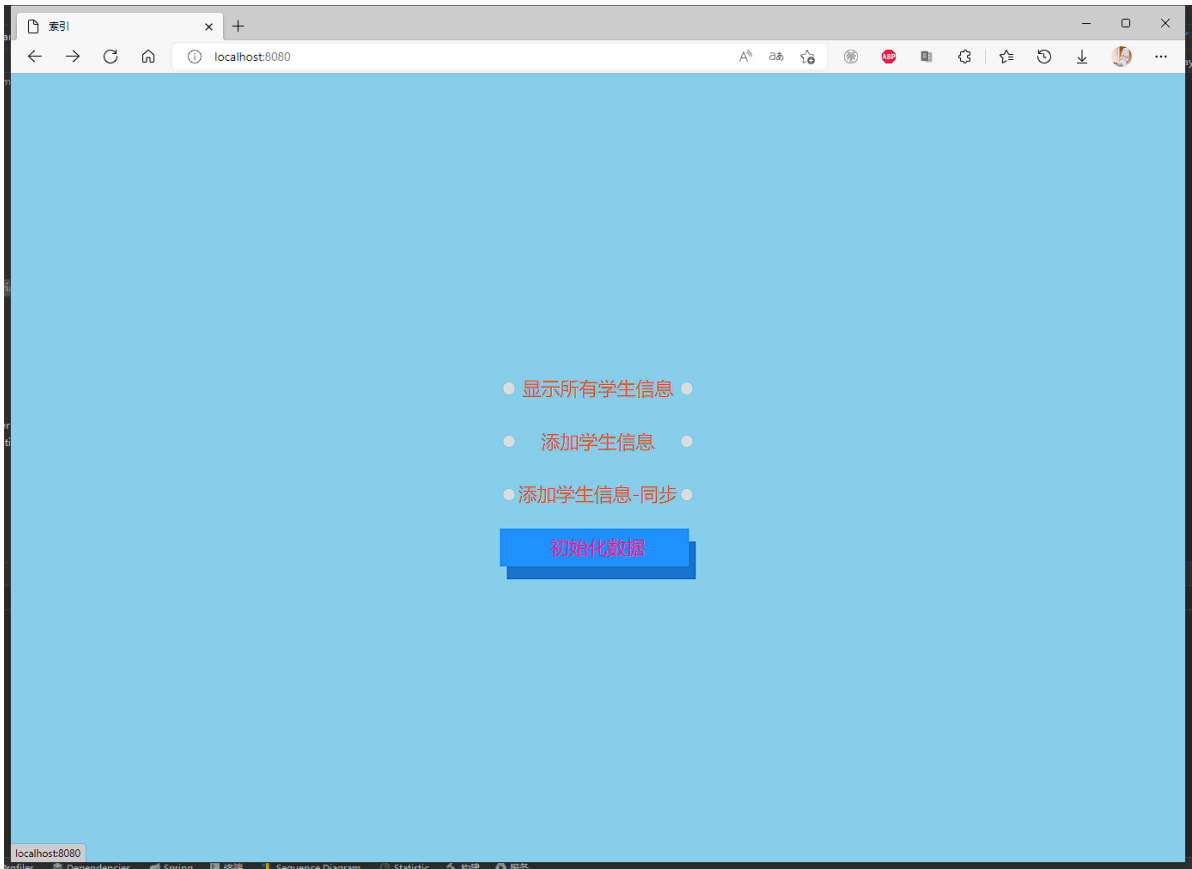
```

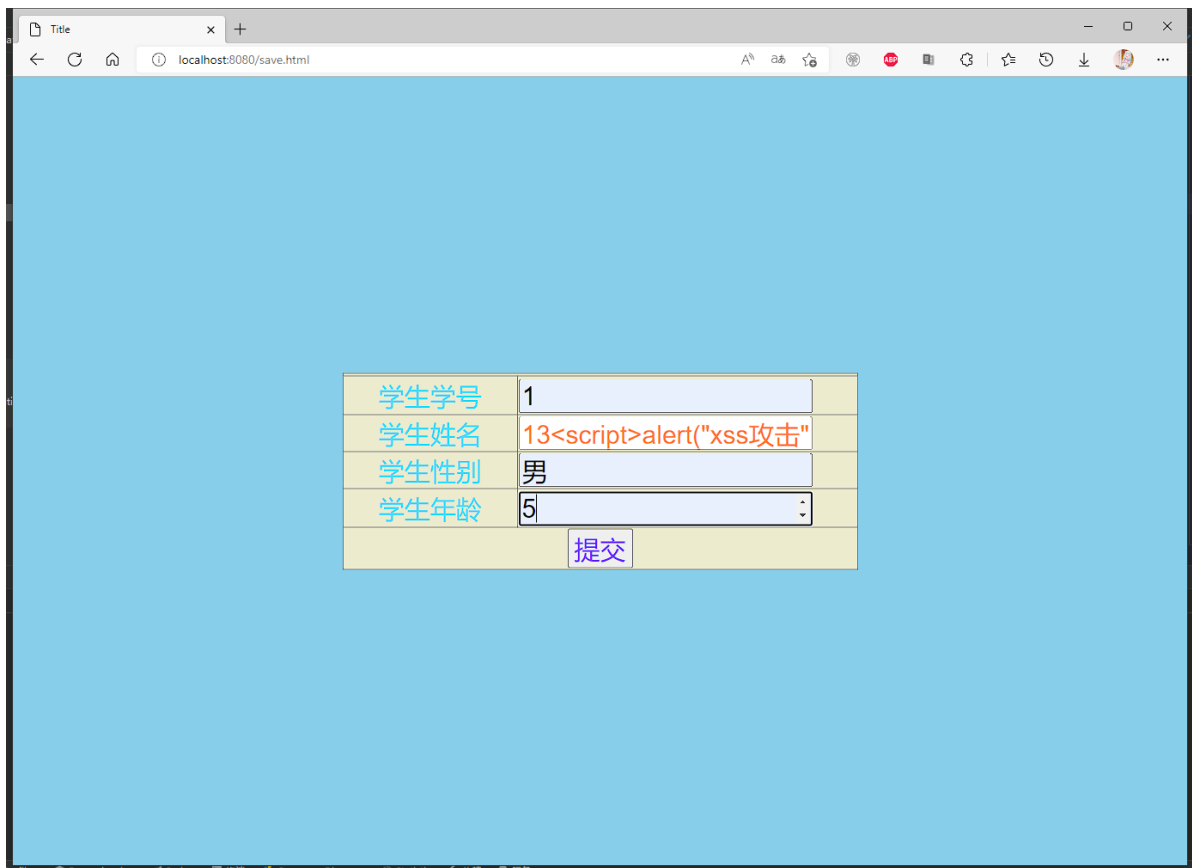
```

37     @Autowired
38     private XssFilterService xssFilterService;
39
40     @PostMapping("/init")
41     public synchronized void init()
42     {
43         Student student1 = new Student(10001, "张三", "男", 18);
44         Student student2 = new Student(10002, "李四", "女", 16);
45         Student student3 = new Student(10003, "王五", "男", 20);
46         list.clear();
47         list.add(student1);
48         list.add(student2);
49         list.add(student3);
50         log.info("初始化完成");
51     }
52
53     @PostMapping
54     public boolean save(@RequestBody Student student)
55     {
56         list.add(student);
57         log.info("添加成功: \n" + student);
58         return true;
59     }
60
61
62     // @PostMapping
63     // public boolean save(@RequestBody Student student)
64     // {
65     //     student.setName(xssFilterService.xssClean(student.getName()));
66     //     student.setSex(xssFilterService.xssClean(student.getSex()));
67     //
68     //     list.add(student);
69     //     log.info("添加成功: \n" + student);
70     //     return true;
71     // }
72
73
74     @PostMapping("/sync")
75     public boolean savesync(Student student)
76     {
77         list.add(student);
78         log.info("添加成功: \n" + student);
79         return true;
80     }
81
82     @GetMapping
83     public List<Student> getAll()
84     {
85         log.info("查询所有: \n" + list);
86         return list;
87     }
88
89 }

```

## 第二十一部：重启并访问服务





```
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

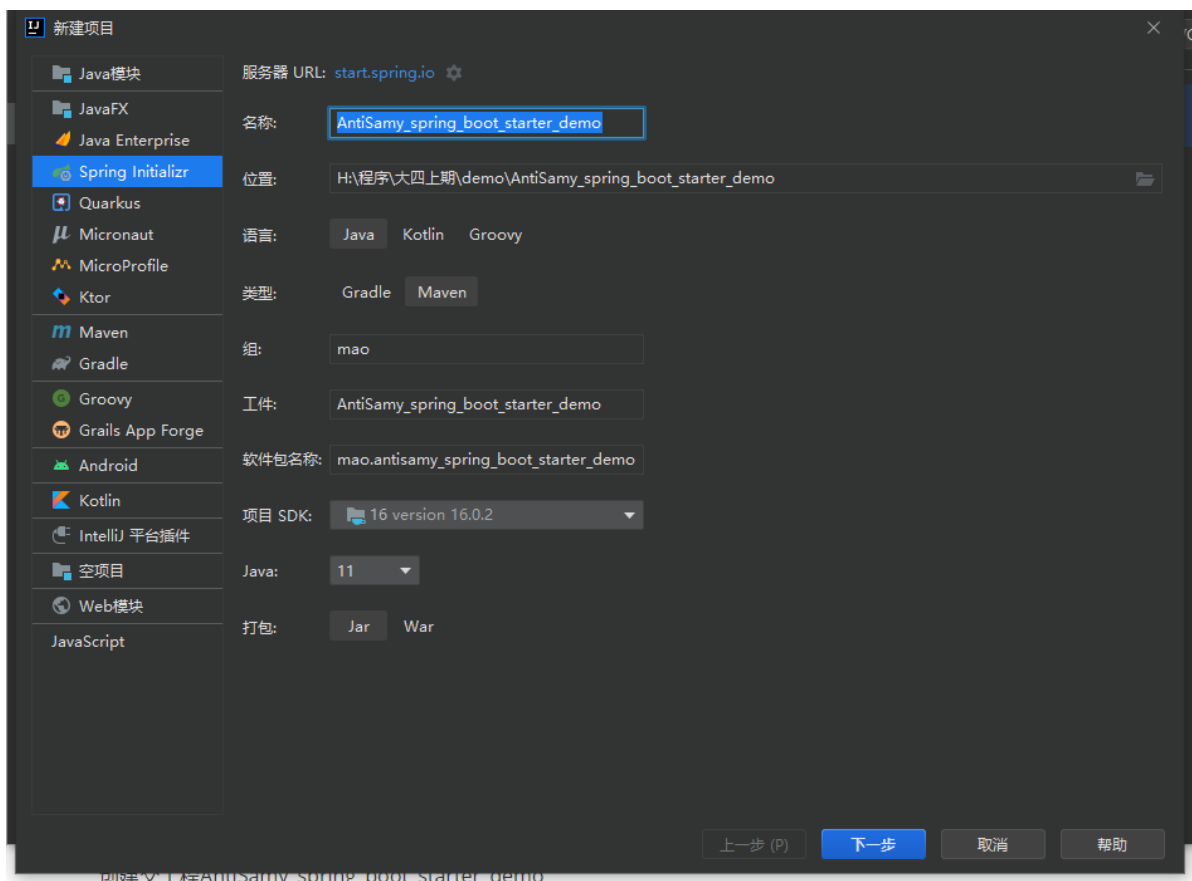
# 自定义spring boot starter

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

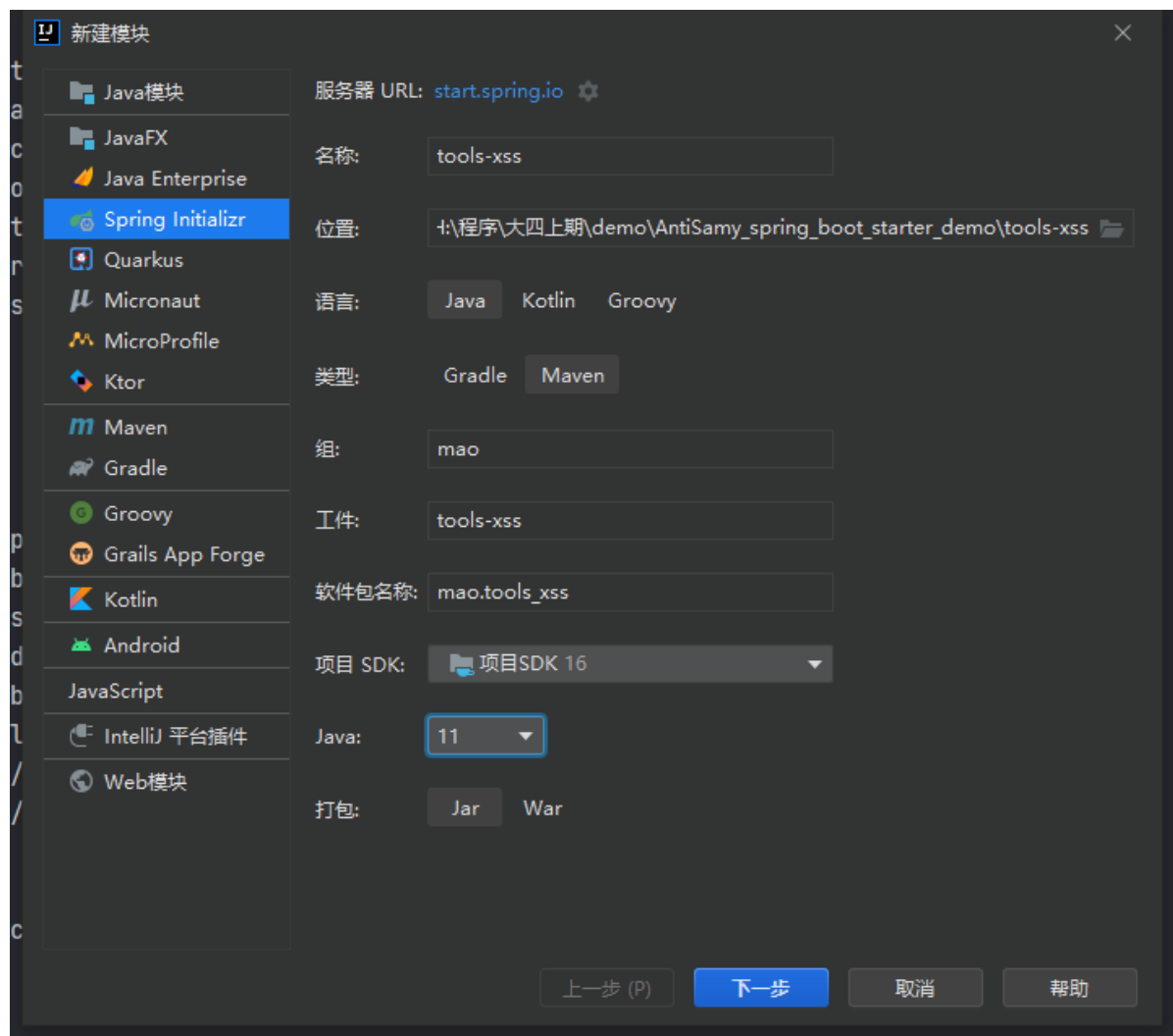
## 开发starter

### 第一步：初始化项目

创建父工程AntiSamy\_spring\_boot\_starter\_demo

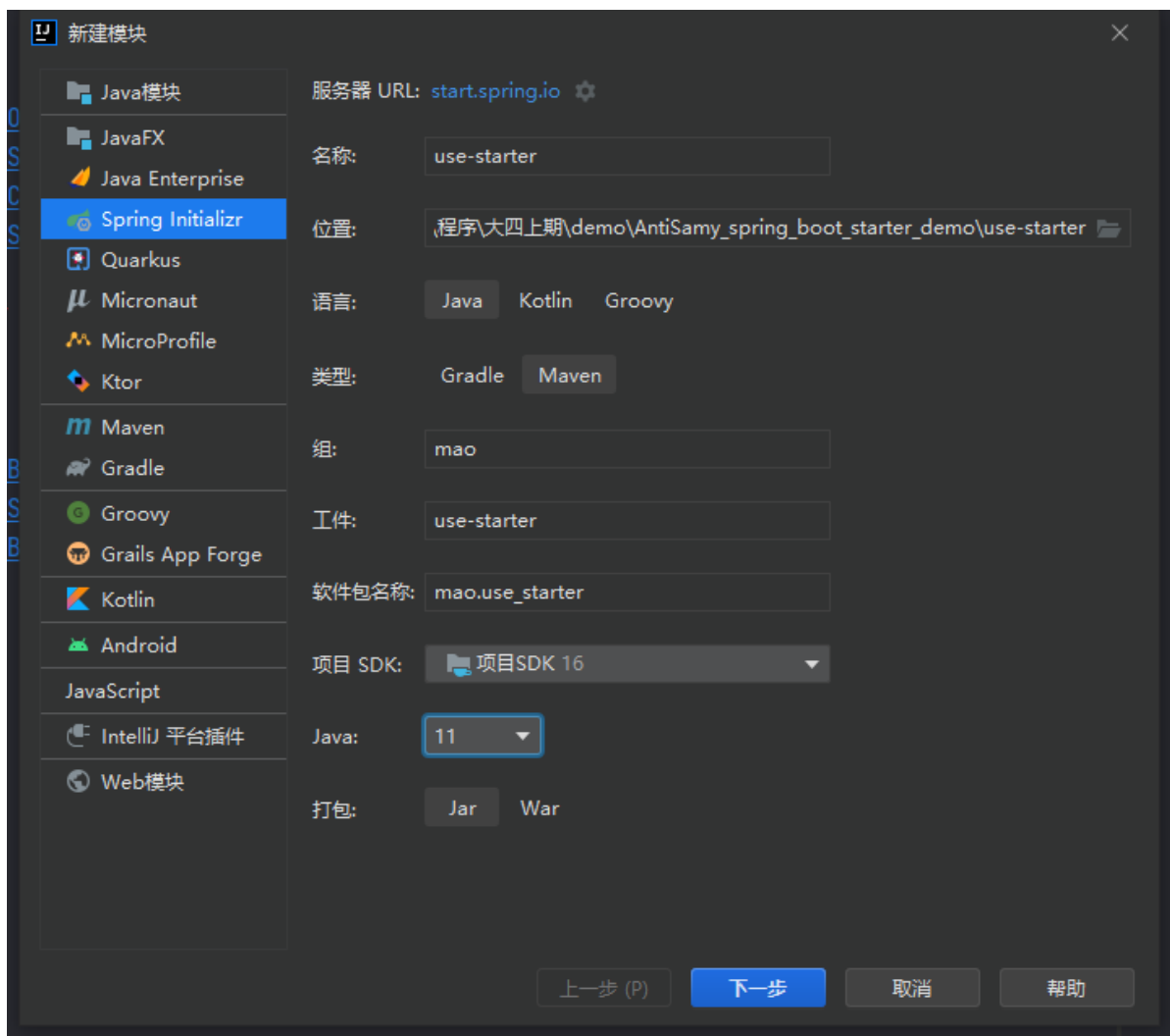


创建子工程tools-xss



创建子工程use-starter





## 第二步：修改pom文件

父工程AntiSamy\_spring\_boot\_starter\_demo的pom文件：

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
4     <modelVersion>4.0.0</modelVersion>
5     <parent>
6         <artifactId>AntiSamy_spring_boot_starter_demo</artifactId>
7         <groupId>mao</groupId>
8         <version>0.0.1-SNAPSHOT</version>
9     </parent>
10
11     <artifactId>use-starter</artifactId>
12     <version>0.0.1-SNAPSHOT</version>
13     <name>use-starter</name>
14     <description>use-starter</description>
15
```

```

16     <properties>
17
18     </properties>
19
20     <dependencies>
21
22         <dependency>
23             <groupId>org.springframework.boot</groupId>
24             <artifactId>spring-boot-starter-web</artifactId>
25         </dependency>
26
27         <dependency>
28             <groupId>org.springframework.boot</groupId>
29             <artifactId>spring-boot-starter-test</artifactId>
30             <scope>test</scope>
31         </dependency>
32
33     </dependencies>
34
35     <build>
36         <plugins>
37             <plugin>
38                 <groupId>org.springframework.boot</groupId>
39                 <artifactId>spring-boot-maven-plugin</artifactId>
40             </plugin>
41         </plugins>
42     </build>
43
44 </project>

```

子工程tools-xss的pom文件:

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5  https://maven.apache.org/xsd/maven-4.0.0.xsd">
6      <modelVersion>4.0.0</modelVersion>
7      <parent>
8          <artifactId>AntiSamy_spring_boot_starter_demo</artifactId>
9          <groupId>mao</groupId>
10         <version>0.0.1-SNAPSHOT</version>
11     </parent>
12
13     <artifactId>tools-xss</artifactId>
14     <version>0.0.1-SNAPSHOT</version>
15     <name>tools-xss</name>
16     <description>tools-xss</description>
17
18     <properties>
19
20     </properties>

```

```

19
20 <dependencies>
21
22     <dependency>
23         <groupId>org.springframework.boot</groupId>
24         <artifactId>spring-boot-starter-web</artifactId>
25     </dependency>
26
27     <!--spring boot starter开发依赖-->
28     <dependency>
29         <groupId>org.springframework.boot</groupId>
30         <artifactId>spring-boot-starter</artifactId>
31     </dependency>
32
33     <dependency>
34         <groupId>org.springframework.boot</groupId>
35         <artifactId>spring-boot-autoconfigure</artifactId>
36     </dependency>
37
38     <dependency>
39         <groupId>org.springframework.boot</groupId>
40         <artifactId>spring-boot-configuration-processor</artifactId>
41     </dependency>
42
43     <!--解决xss攻击-->
44     <dependency>
45         <groupId>org.owasp.antisamy</groupId>
46         <artifactId>antisamy</artifactId>
47         <version>1.5.7</version>
48     </dependency>
49
50     <!--阿里巴巴的FastJson json解析-->
51     <dependency>
52         <groupId>com.alibaba</groupId>
53         <artifactId>fastjson</artifactId>
54         <version>1.2.79</version>
55     </dependency>
56
57 </dependencies>
58
59 <build>
60     <plugins>
61         <plugin>
62             <groupId>org.springframework.boot</groupId>
63             <artifactId>spring-boot-maven-plugin</artifactId>
64             <configuration>
65                 <skip>true</skip>
66             </configuration>
67         </plugin>
68     </plugins>
69 </build>
70
71 </project>
72

```

子工程use-starter的pom文件:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <project xmlns="http://maven.apache.org/POM/4.0.0"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5       https://maven.apache.org/xsd/maven-4.0.0.xsd">
6   <modelVersion>4.0.0</modelVersion>
7   <parent>
8     <artifactId>AntiSamy_spring_boot_starter_demo</artifactId>
9     <groupId>mao</groupId>
10    <version>0.0.1-SNAPSHOT</version>
11  </parent>
12  <artifactId>use-starter</artifactId>
13  <version>0.0.1-SNAPSHOT</version>
14  <name>use-starter</name>
15  <description>use-starter</description>
16  <properties>
17  </properties>
18  <dependencies>
19
20    <dependency>
21      <groupId>org.springframework.boot</groupId>
22      <artifactId>spring-boot-starter-web</artifactId>
23    </dependency>
24
25    <dependency>
26      <groupId>org.springframework.boot</groupId>
27      <artifactId>spring-boot-starter-test</artifactId>
28      <scope>test</scope>
29    </dependency>
30
31  </dependencies>
32
33  <build>
34    <plugins>
35      <plugin>
36        <groupId>org.springframework.boot</groupId>
37        <artifactId>spring-boot-maven-plugin</artifactId>
38      </plugin>
39    </plugins>
40  </build>
41
42 </project>
```

### 第三步：编写工具类XssUtils

```
1 package mao.tools_xss.utils;
2
3
4 import org.owasp.validator.html.*;
5 import org.slf4j.Logger;
6 import org.slf4j.LoggerFactory;
7
8 import java.io.IOException;
9 import java.io.InputStream;
10 import java.util.List;
11
12 /**
13  * Project name(项目名称): AntiSamy_spring_boot_starter_demo
14  * Package(包名): mao.tools_xss.utils
15  * Class(类名): XssUtils
16  * Author(作者): mao
17  * Author QQ: 1296193245
18  * GitHub: https://github.com/maomao124/
19  * Date(创建日期): 2022/10/30
20  * Time(创建时间): 20:03
21  * Version(版本): 1.0
22  * Description(描述): XSS 工具类, 用于过滤特殊字符
23  */
24
25 public class XssUtils
26 {
27     private static final Logger log =
28     LoggerFactory.getLogger(XssUtils.class);
29
30     private static final String ANTISAMY_SLASHDOT_XML = "antisamy-slashdot-
31     1.4.4.xml";
32     private static Policy policy = null;
33
34     static
35     {
36         log.debug(" start read xss configfile [" + ANTISAMY_SLASHDOT_XML +
37         "]);
38         InputStream inputStream =
39         XssUtils.class.getClassLoader().getResourceAsStream(ANTISAMY_SLASHDOT_XML);
40         try
41         {
42             policy = Policy.getInstance(inputStream);
43             log.debug("read xss configfile [" + ANTISAMY_SLASHDOT_XML + "]
44             success");
45         }
46         catch (PolicyException e)
47         {
48             log.error("read xss configfile [" + ANTISAMY_SLASHDOT_XML + "]
49             fail , reason:", e);
50         }
51         finally
52         {
53             if (inputStream != null)
```

```

48         {
49             try
50             {
51                 inputStream.close();
52             }
53             catch (IOException e)
54             {
55                 log.error("close XSS configfile [" +
ANTISAMY_SLASHDOT_XML + "] fail , reason:", e);
56             }
57         }
58     }
59 }
60
61 /**
62  * 跨站攻击语句过滤 方法
63  *
64  * @param paramValue      待过滤的参数
65  * @param ignoreParamValueList 忽略过滤的参数列表
66  * @return String
67  */
68 public static String xssClean(String paramValue, List<String>
ignoreParamValueList)
69 {
70     AntiSamy antiSamy = new AntiSamy();
71     try
72     {
73         log.debug("raw value before xssClean: " + paramValue);
74         if (isIgnoreParamValue(paramValue, ignoreParamValueList))
75         {
76             log.debug("ignore the xssClean,keep the raw paramValue: " +
paramValue);
77             return paramValue;
78         }
79         else
80         {
81             final CleanResults cleanResults = antiSamy.scan(paramValue,
policy);
82             cleanResults.getErrorMessage().forEach(log::debug);
83             String str = cleanResults.getCleanHTML();
84             /*String str =
StringEscapeUtils.escapeHtml(cleanResults.getCleanHTML());
85             str = str.replaceAll((antiSamy.scan("&nbsp;",
policy)).getCleanHTML(), "");
86             str = StringEscapeUtils.unescapeHtml(str);*/
87             /*str = str.replaceAll(""", "\"");
88             str = str.replaceAll("&", "&");
89             str = str.replaceAll("'", "'");
90             str = str.replaceAll("'", "' ");
91
92             str = str.replaceAll("<", "<");
93             str = str.replaceAll(">", ">");*/
94             log.debug("xssfilter value after xssClean" + str);
95
96             return str;
97         }
98     }
99     catch (ScanException e)

```

```

100         {
101             log.error("scan failed armter is [" + paramValue + "]", e);
102         }
103         catch (PolicyException e)
104         {
105             log.error("antisamy convert failed armter is [" + paramValue +
106                 "]", e);
107         }
108         return paramValue;
109     }
110     /**
111     * 忽略参数值
112     *
113     * @param paramValue      参数值
114     * @param ignoreParamValueList 忽略参数值列表
115     * @return boolean
116     */
117     private static boolean isIgnoreParamValue(String paramValue,
118         List<String> ignoreParamValueList)
119     {
120         if (paramValue == null || paramValue.equals(""))
121         {
122             return true;
123         }
124         if (ignoreParamValueList == null || ignoreParamValueList.size() ==
125             0)
126         {
127             return false;
128         }
129         else
130         {
131             for (String ignoreParamValue : ignoreParamValueList)
132             {
133                 if (paramValue.contains(ignoreParamValue))
134                 {
135                     return true;
136                 }
137             }
138             return false;
139         }
140     }

```

#### 第四步：编写类XssRequestWrapper

```

1 package mao.tools_xss.wrapper;
2
3 import mao.tools_xss.utils.XssUtils;
4 import org.slf4j.Logger;

```

```

5  import org.slf4j.LoggerFactory;
6
7  import javax.servlet.http.HttpServletRequest;
8  import javax.servlet.http.HttpServletRequestWrapper;
9  import java.util.List;
10 import java.util.Map;
11
12 /**
13  * Project name(项目名称): AntiSamy_spring_boot_starter_demo
14  * Package(包名): mao.tools.xss.wrapper
15  * Class(类名): XssRequestWrapper
16  * Author(作者): mao
17  * Author QQ: 1296193245
18  * GitHub: https://github.com/maomao124/
19  * Date(创建日期): 2022/10/30
20  * Time(创建时间): 20:01
21  * Version(版本): 1.0
22  * Description(描述): 无
23  */
24
25 public class XssRequestWrapper extends HttpServletRequestWrapper
26 {
27
28     private static final Logger log =
29         LoggerFactory.getLogger(XssRequestWrapper.class);
30
31     private final List<String> ignoreParamValueList;
32
33     public XssRequestWrapper(HttpServletRequest request, List<String>
34         ignoreParamValueList)
35     {
36         super(request);
37         this.ignoreParamValueList = ignoreParamValueList;
38     }
39
40     @Override
41     public Map<String, String[]> getParameterMap()
42     {
43         Map<String, String[]> requestMap = super.getParameterMap();
44         for (Map.Entry<String, String[]> me : requestMap.entrySet())
45         {
46             log.debug(me.getKey() + ":");
47             String[] values = me.getValue();
48             for (int i = 0; i < values.length; i++)
49             {
50                 log.debug(values[i]);
51                 values[i] = xssUtils.xssClean(values[i],
52                     this.ignoreParamValueList);
53             }
54             return requestMap;
55         }
56     }
57
58     @Override
59     public String[] getParameterValues(String paramString)
60     {
61         String[] arrayOfString1 = super.getParameterValues(paramString);
62         if (arrayOfString1 == null)

```



```

60         {
61             return null;
62         }
63         int i = arrayOfString1.length;
64         String[] arrayOfString2 = new String[i];
65         for (int j = 0; j < i; j++)
66         {
67             arrayOfString2[j] = XssUtils.xssClean(arrayOfString1[j],
this.ignoreParamValueList);
68         }
69         return arrayOfString2;
70     }
71
72     @Override
73     public String getParameter(String paramString)
74     {
75         String str = super.getParameter(paramString);
76         if (str == null)
77         {
78             return null;
79         }
80         return XssUtils.xssClean(str, this.ignoreParamValueList);
81     }
82
83     @Override
84     public String getHeader(String paramString)
85     {
86         String str = super.getHeader(paramString);
87         if (str == null)
88         {
89             return null;
90         }
91         return XssUtils.xssClean(str, this.ignoreParamValueList);
92     }
93 }

```

## 第五步：编写类XssStringJsonDeserializer

```

1  package mao.tools_xss.converter;
2
3  import com.fasterxml.jackson.core.JsonParser;
4  import com.fasterxml.jackson.core.JsonProcessingException;
5  import com.fasterxml.jackson.core.JsonToken;
6  import com.fasterxml.jackson.databind.DeserializationContext;
7  import com.fasterxml.jackson.databind.JsonDeserializer;
8
9  import com.fasterxml.jackson.databind.JsonDeserializer;
10 import mao.tools_xss.utils.XssUtils;
11
12 import java.io.IOException;

```

```

13 import java.util.ArrayList;
14 import java.util.List;
15
16 /**
17  * Project name(项目名称): AntiSamy_spring_boot_starter_demo
18  * Package(包名): mao.tools_xss.converter
19  * Class(类名): XssStringJsonDeserializer
20  * Author(作者): mao
21  * Author QQ: 1296193245
22  * GitHub: https://github.com/maomao124/
23  * Date(创建日期): 2022/10/30
24  * Time(创建时间): 20:14
25  * Version(版本): 1.0
26  * Description(描述): 过滤跨站脚本的 反序列化工具
27  */
28
29 public class XssStringJsonDeserializer extends JsonSerializer<String>
30 {
31
32     private static final List<String> list = new ArrayList<>();
33
34     static
35     {
36         list.add("<script>");
37         list.add("</script>");
38         list.add("<iframe>");
39         list.add("</iframe>");
40         list.add("<noscript>");
41         list.add("</noscript>");
42         list.add("<frameset>");
43         list.add("</frameset>");
44         list.add("<frame>");
45         list.add("</frame>");
46         list.add("<noframes>");
47         list.add("</noframes>");
48         list.add("<h1>");
49         list.add("</h1>");
50         list.add("<h2>");
51         list.add("</h2>");
52         list.add("<h3>");
53         list.add("</h3>");
54         list.add("<h4>");
55         list.add("</h4>");
56         list.add("<h5>");
57         list.add("</h5>");
58         list.add("<h6>");
59         list.add("</h6>");
60         list.add("<img>");
61         list.add("</img>");
62         list.add("<table>");
63         list.add("</table>");
64         list.add("<form>");
65         list.add("</form>");
66     }
67
68
69     @Override

```

```

70     public String deserialize(JsonParser p, DeserializationContext dc)
       throws IOException, JsonProcessingException
71     {
72         if (p.hasToken(JsonToken.VALUE_STRING))
73         {
74             String value = p.getValueAsString();
75
76             if (value == null || "".equals(value))
77             {
78                 return value;
79             }
80
81             boolean flag = list.stream().anyMatch(value::contains);
82             if (flag)
83             {
84                 return XssUtils.xssClean(value, null);
85             }
86             return value;
87         }
88         return null;
89     }
90 }
91

```

## 第六步：编写过滤器类XssFilter

```

1  package mao.tools_xss.filter;
2
3  import com.alibaba.fastjson.JSON;
4  import mao.tools_xss.wrapper.XssRequestWrapper;
5  import org.slf4j.Logger;
6  import org.slf4j.LoggerFactory;
7
8
9  import javax.servlet.Filter;
10 import javax.servlet.FilterChain;
11 import javax.servlet.FilterConfig;
12 import javax.servlet.ServletException;
13 import javax.servlet.ServletRequest;
14 import javax.servlet.ServletResponse;
15 import javax.servlet.http.HttpServletRequest;
16 import java.io.IOException;
17 import java.util.ArrayList;
18 import java.util.Arrays;
19 import java.util.List;
20
21 /**
22  * Project name(项目名称): AntiSamy_spring_boot_starter_demo
23  * Package(包名): mao.tools_xss.filter
24  * Class(类名): XssFilter
25  * Author(作者): mao

```

```

26  * Author QQ: 1296193245
27  * GitHub: https://github.com/maomao124/
28  * Date(创建日期): 2022/10/30
29  * Time(创建时间): 20:18
30  * Version(版本): 1.0
31  * Description(描述): 无
32  */
33
34  public class XssFilter implements Filter
35  {
36
37      private static final Logger log =
38      LoggerFactory.getLogger(XssFilter.class);
39
40      /**
41       * 可放行的请求路径
42       */
43      private static final String IGNORE_PATH = "ignorePath";
44
45      /**
46       * 可放行的参数值
47       */
48      private static final String IGNORE_PARAM_VALUE = "ignoreParamValue";
49
50      /**
51       * 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)
52       */
53      private static final String CAS_LOGOUT_RESPONSE_TAG =
54      "samlp:LogoutRequest";
55
56      /**
57       * 可放行的请求路径列表
58       */
59      private List<String> ignorePathList;
60
61      /**
62       * 可放行的参数值列表
63       */
64      private List<String> ignoreParamValueList;
65
66      @Override
67      public void init(FilterConfig filterConfig) throws ServletException
68      {
69          log.debug("XSS filter [XssFilter] init start ...");
70          String ignorePaths = filterConfig.getInitParameter(IGNORE_PATH);
71          String ignoreParamValues =
72          filterConfig.getInitParameter(IGNORE_PARAM_VALUE);
73          if (!(ignorePaths == null || ignorePaths.equals("")))
74          {
75              String[] ignorePathArr = ignorePaths.split(",");
76              ignorePathList = Arrays.asList(ignorePathArr);
77          }
78          if (!(ignoreParamValues == null || ignoreParamValues.equals("")))
79          {
80              String[] ignoreParamValueArr = ignoreParamValues.split(",");
81              ignoreParamValueList = Arrays.asList(ignoreParamValueArr);
82              //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
83              行)
84          }
85          if (!ignoreParamValueList.contains(CAS_LOGOUT_RESPONSE_TAG))
86          {
87              ignoreParamValueList.add(CAS_LOGOUT_RESPONSE_TAG);
88          }
89      }
90  }

```

```

80     }
81     else
82     {
83         //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)
84         ignoreParamValueList = new ArrayList<>();
85         ignoreParamValueList.add(CAS_LOGOUT_RESPONSE_TAG);
86     }
87     log.debug("ignorePathList=" + JSON.toJSONString(ignorePathList));
88     log.debug("ignoreParamValueList=" +
JSON.toJSONString(ignoreParamValueList));
89     log.debug("XSS fiter [XSSFilter] init end");
90 }
91
92 @Override
93 public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain)
94     throws IOException, ServletException
95 {
96     log.debug("XSS fiter [XSSFilter] starting");
97     // 判断uri是否包含项目名称
98     String uriPath = ((HttpServletRequest) request).getRequestURI();
99     if (isIgnorePath(uriPath))
100     {
101         log.debug("ignore xssfilter,path[" + uriPath + "] pass through
XssFilter, go ahead...");
102         chain.doFilter(request, response);
103         return;
104     }
105     else
106     {
107         log.debug("has xssfilter path[" + uriPath + "] need XssFilter,
go to XssRequestWrapper");
108         //传入重写后的Request
109         chain.doFilter(new XssRequestWrapper((HttpServletRequest)
request, ignoreParamValueList), response);
110     }
111     log.debug("XSS fiter [XSSFilter] stop");
112 }
113
114 @Override
115 public void destroy()
116 {
117     log.debug("XSS fiter [XSSFilter] destroy");
118 }
119
120 /**
121  * 是否为忽略的请求路径
122  *
123  * @param servletPath servlet路径
124  * @return boolean
125  */
126 private boolean isIgnorePath(String servletPath)
127 {
128     if (servletPath == null || servletPath.equals(""))
129     {
130         return true;
131     }

```

```

132         if (ignorePathList == null || ignorePathList.size() == 0)
133         {
134             return false;
135         }
136         else
137         {
138             for (String ignorePath : ignorePathList)
139             {
140                 if (!(ignorePath == null || ignorePath.equals("")) &&
servletPath.contains(ignorePath.trim()))
141                 {
142                     return true;
143                 }
144             }
145         }
146         return false;
147     }
148 }

```

## 第七步：拷贝之前的service包到此项目中，并更改

```

1  package mao.tools_xss.service;
2
3  import org.owasp.validator.html.*;
4  import org.slf4j.Logger;
5  import org.slf4j.LoggerFactory;
6
7
8  import java.io.InputStream;
9
10 /**
11  * Project name(项目名称): antiSamy_demo
12  * Package(包名): mao.antisamy_demo.service
13  * Class(类名): XssFilterService
14  * Author(作者): mao
15  * Author QQ: 1296193245
16  * GitHub: https://github.com/maomao124/
17  * Date(创建日期): 2022/10/30
18  * Time(创建时间): 15:43
19  * Version(版本): 1.0
20  * Description(描述): 无
21  */
22
23 public class XssFilterService
24 {
25     /**
26     * 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
27     */
28     @SuppressWarnings("all")
29     private static final String antiSamyPath = "antisamy-slashdot-
1.4.4.xml";

```

```

30
31     public static Policy policy = null;
32
33     private static final Logger log =
LoggerFactory.getLogger(XssFilterService.class);
34
35     static
36     {
37         // 指定策略文件
38         try
39         {
40             InputStream inputStream =
XssFilterService.class.getClassLoader().getResourceAsStream(antiSamyPath);
41             assert inputStream != null;
42             policy = Policy.getInstance(inputStream);
43         }
44         catch (PolicyException e)
45         {
46             e.printStackTrace();
47         }
48     }
49
50     /**
51      * AntiSamy过滤数据
52      *
53      * @param taintedHTML 需要进行过滤的数据
54      * @return 返回过滤后的数据
55      */
56     public String xssClean(String taintedHTML)
57     {
58         try
59         {
60             // 使用AntiSamy进行过滤
61             AntiSamy antiSamy = new AntiSamy();
62             CleanResults cleanResults = antiSamy.scan(taintedHTML, policy);
63             taintedHTML = cleanResults.getCleanHTML();
64         }
65         catch (ScanException | PolicyException e)
66         {
67             e.printStackTrace();
68         }
69         return taintedHTML;
70     }
71 }
72

```

## 第八步：编写配置类XssAuthConfiguration

```
1 package mao.tools_xss.config;
2
3 import mao.tools_xss.converter.XssStringJsonDeserializer;
4 import mao.tools_xss.filter.XssFilter;
5 import mao.tools_xss.service.XssFilterService;
6 import
org.springframework.boot.autoconfigure.jackson.Jackson2ObjectMapperBuilderCu
stomizer;
7 import org.springframework.boot.web.servlet.FilterRegistrationBean;
8 import org.springframework.context.annotation.Bean;
9 import org.springframework.context.annotation.Configuration;
10
11 import java.util.HashMap;
12 import java.util.Map;
13 import java.util.StringJoiner;
14
15 /**
16  * Project name(项目名称): AntiSamy_spring_boot_starter_demo
17  * Package(包名): mao.tools_xss.config
18  * Class(类名): XssAuthConfiguration
19  * Author(作者): mao
20  * Author QQ: 1296193245
21  * GitHub: https://github.com/maomao124/
22  * Date(创建日期): 2022/10/30
23  * Time(创建时间): 20:31
24  * Version(版本): 1.0
25  * Description(描述): XSS 跨站攻击自动配置
26  */
27
28 @Configuration
29 public class XssAuthConfiguration
30 {
31     /**
32      * 配置跨站攻击 反序列化处理器
33      *
34      * @return Jackson2ObjectMapperBuilderCustomizer
35      */
36     @Bean
37     public Jackson2ObjectMapperBuilderCustomizer
jackson2ObjectMapperBuilderCustomizer2()
38     {
39         return builder -> builder.deserializerByType(String.class, new
XssStringJsonDeserializer());
40     }
41
42     /**
43      * 配置跨站攻击过滤器
44      *
45      * @return FilterRegistrationBean
46      */
47     @Bean
48     public FilterRegistrationBean<XssFilter> filterRegistrationBean()
```

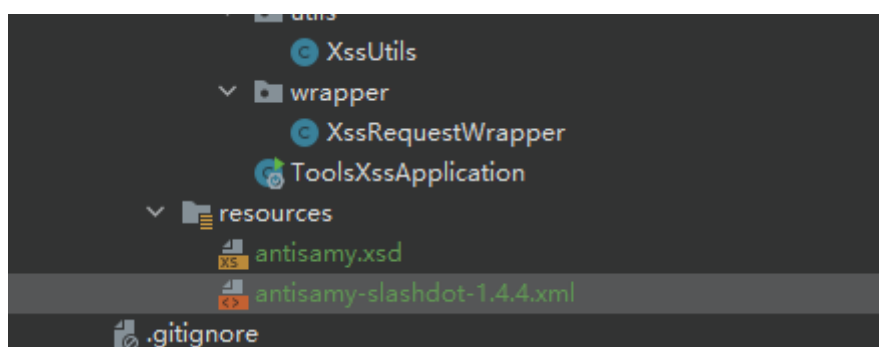


```

50     {
51         //可以拓展
52
53         FilterRegistrationBean<XssFilter> filterRegistration = new
FilterRegistrationBean<>(new XssFilter());
54         filterRegistration.addUrlPatterns("/");
55         filterRegistration.setOrder(1);
56
57         Map<String, String> initParameters = new HashMap<>(2);
58         String excludes = new StringJoiner(",")
59             .add("/favicon.ico")
60             .add("/doc.html")
61             .add("/swagger-ui.html")
62             .add("/csrf")
63             .add("/webjars/*")
64             .add("/v2/*")
65             .add("/swagger-resources/*")
66             .add("/resources/*")
67             .add("/static/*")
68             .add("/public/*")
69             .add("/classpath:/*")
70             .add("/actuator/*")
71             .toString();
72         initParameters.put("excludes", excludes);
73         initParameters.put("isIncludeRichText", "true");
74         filterRegistration.setInitParameters(initParameters);
75         return filterRegistration;
76     }
77
78     @Bean
79     public XssFilterService xssFilterService()
80     {
81         return new XssFilterService();
82     }
83 }
84

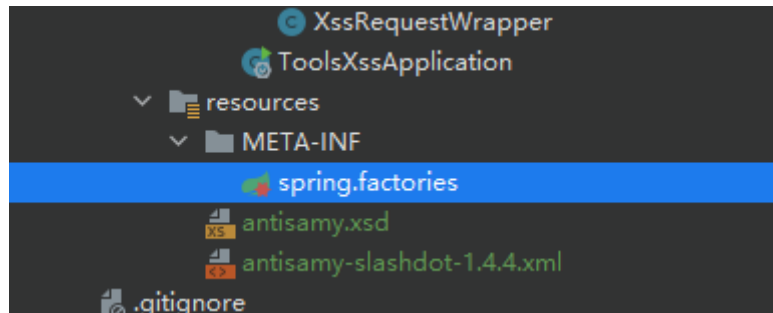
```

**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



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

## 使用starter

### 第一步：导入tools-xss的依赖

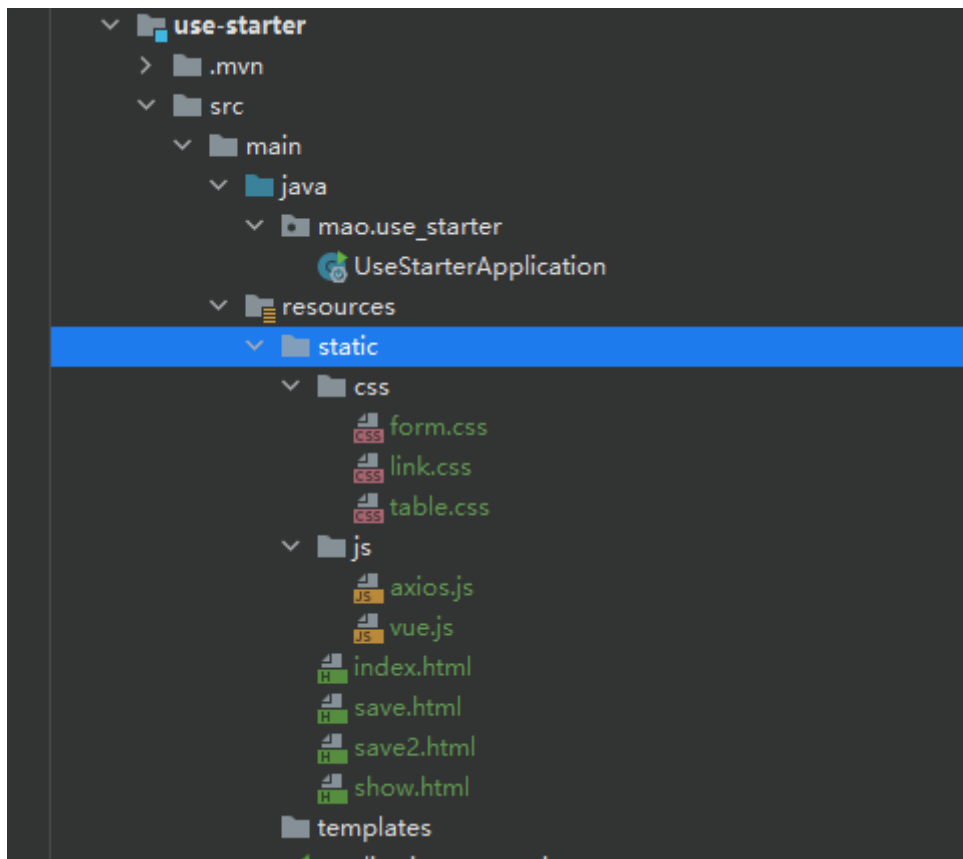
```
1 | <?xml version="1.0" encoding="UTF-8"?>
2 | <project xmlns="http://maven.apache.org/POM/4.0.0"
3 |   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4 |     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5 |       https://maven.apache.org/xsd/maven-4.0.0.xsd">
6 |   <modelVersion>4.0.0</modelVersion>
7 |   <parent>
8 |     <artifactId>AntiSamy_spring_boot_starter_demo</artifactId>
9 |     <groupId>mao</groupId>
10 |     <version>0.0.1-SNAPSHOT</version>
11 |   </parent>
```

```

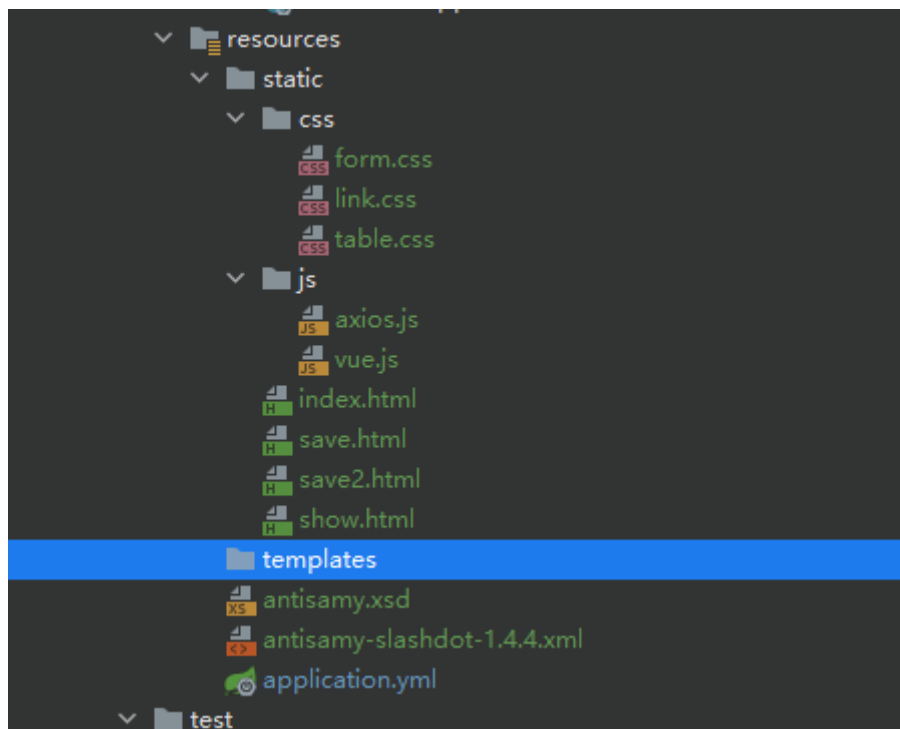
11     <artifactId>use-starter</artifactId>
12     <version>0.0.1-SNAPSHOT</version>
13     <name>use-starter</name>
14     <description>use-starter</description>
15
16     <properties>
17
18     </properties>
19
20     <dependencies>
21
22         <dependency>
23             <groupId>org.springframework.boot</groupId>
24             <artifactId>spring-boot-starter-web</artifactId>
25         </dependency>
26
27         <dependency>
28             <groupId>org.springframework.boot</groupId>
29             <artifactId>spring-boot-starter-test</artifactId>
30             <scope>test</scope>
31         </dependency>
32
33         <dependency>
34             <groupId>mao</groupId>
35             <artifactId>tools-xss</artifactId>
36             <version>0.0.1-SNAPSHOT</version>
37         </dependency>
38
39     </dependencies>
40
41     <build>
42         <plugins>
43             <plugin>
44                 <groupId>org.springframework.boot</groupId>
45                 <artifactId>spring-boot-maven-plugin</artifactId>
46             </plugin>
47         </plugins>
48     </build>
49
50 </project>

```

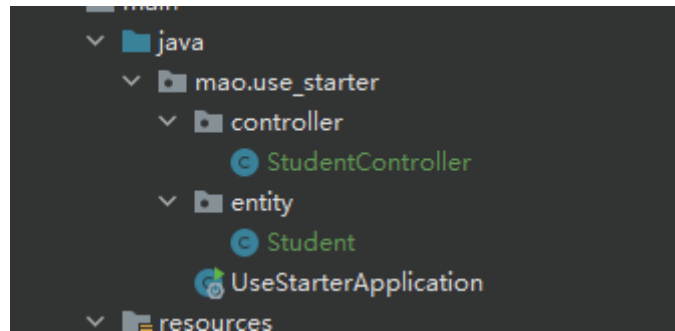
## 第二步：导入之前项目的静态资源文件



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



#### 第四步：拷贝之前的controller和entity包到此项目中



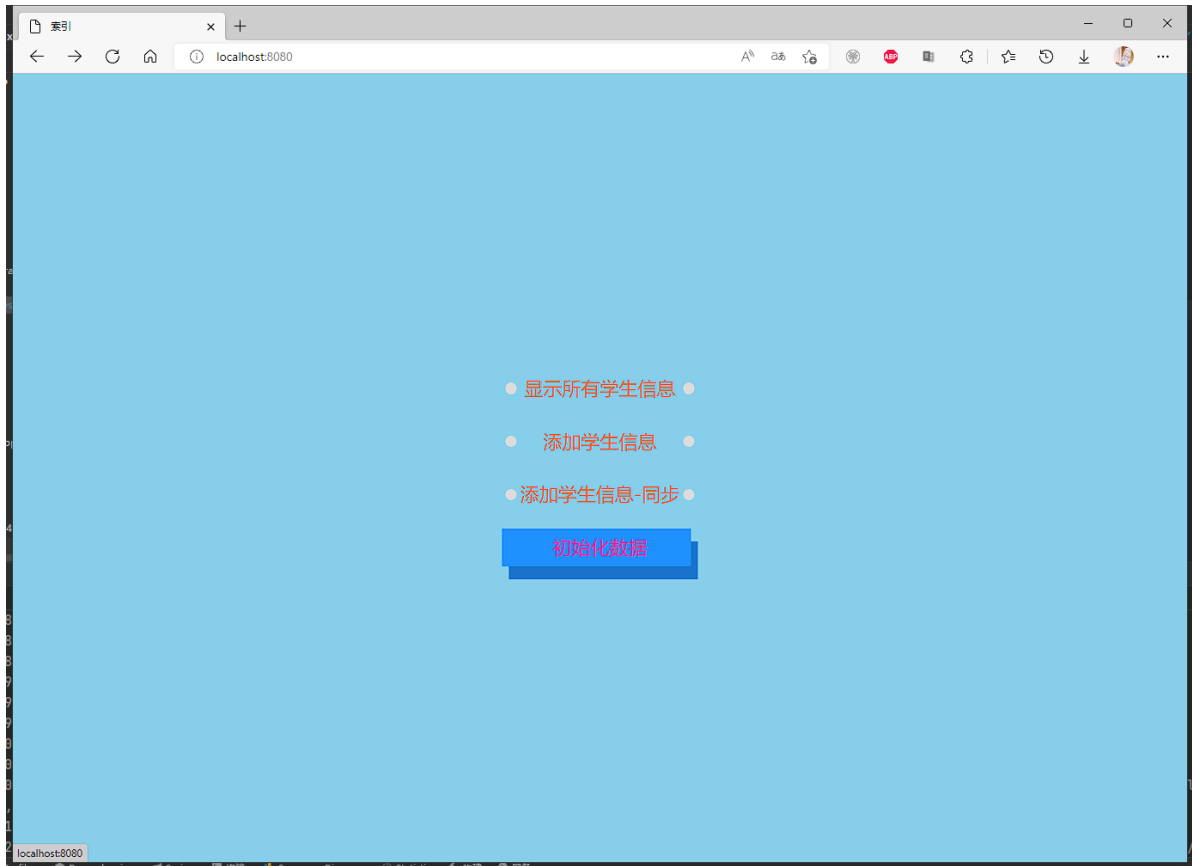
#### 第五步：修改配置文件application.yml

```
1  # 开启debug模式，输出调试信息，常用于检查系统运行状况
2  #debug: true
3
4  # 设置日志级别，root表示根节点，即整体应用日志级别
5  logging:
6    # 日志输出到文件的文件名
7    file:
8      name: server.log
9    # 字符集
10   charset:
11     file: UTF-8
12   # 分文件
13   logback:
14     rollingpolicy:
15       #最大文件大小
16       max-file-size: 16KB
17       # 文件格式
18       file-name-pattern: logs/server_log/%d{yyyy/MM月/dd日/}%i.log
19   # 设置日志组
20   group:
21     # 自定义组名，设置当前组中所包含的包
22     mao_pro: mao
23   level:
24     root: info
25     # 为对应组设置日志级别
26     mao_pro: debug
27     # 日志输出格式
28   # pattern:
29     # console: "%d %clr(%p) --- [%16t] %clr(%-40.40c){cyan} : %m %n"
```

## 第六步：启动程序

```
1
2
3  .  _ _ _ _ _
4  /\ / _ _ _ _ _ ( ) _ _ _ _ _ \ \ \ \
5  ( ( ) \ _ _ | ' _ | ' _ | ' _ \ \ _ | \ \ \ \
6  \ \ _ _ | | _ | | | | | | | ( _ | ) ) ) )
7  ' | _ _ | . _ | | | | | _ \ , | / / / /
8  =====|_|=====|_|_/ _ / _ / _ /
9  :: Spring Boot ::                (v2.7.1)
10 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)
11 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
12 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"
13 2022-10-30 21:39:24.830 INFO 18896 --- [          main]
    o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s):
    8080 (http)
14 2022-10-30 21:39:24.842 INFO 18896 --- [          main]
    o.apache.catalina.core.StandardService : Starting service [Tomcat]
15 2022-10-30 21:39:24.842 INFO 18896 --- [          main]
    org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache
    Tomcat/9.0.64]
16 2022-10-30 21:39:24.924 INFO 18896 --- [          main] o.a.c.c.C.
    [Tomcat].[localhost].[/] : Initializing Spring embedded
    WebApplicationContext
17 2022-10-30 21:39:24.925 INFO 18896 --- [          main]
    w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext:
    initialization completed in 709 ms
18 2022-10-30 21:39:24.956 DEBUG 18896 --- [          main]
    mao.tools_xss.filter.XssFilter : XSS filter [XSSFilter] init start
    ...
19 2022-10-30 21:39:24.981 DEBUG 18896 --- [          main]
    mao.tools_xss.filter.XssFilter : ignorePathList=null
20 2022-10-30 21:39:24.982 DEBUG 18896 --- [          main]
    mao.tools_xss.filter.XssFilter : ignoreParamValueList=
    ["sam!p:LogoutRequest"]
21 2022-10-30 21:39:24.982 DEBUG 18896 --- [          main]
    mao.tools_xss.filter.XssFilter : XSS filter [XSSFilter] init end
22 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]
23 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 ''
24 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



```
1 2022-10-30 21:39:45.985 INFO 18896 --- [nio-8080-exec-2]
   o.s.web.servlet.DispatcherServlet      : Completed initialization in 1 ms
2 2022-10-30 21:39:45.989 DEBUG 18896 --- [nio-8080-exec-2]
   mao.tools_xss.filter.XssFilter         : XSS filter [XssFilter] starting
3 2022-10-30 21:39:45.989 DEBUG 18896 --- [nio-8080-exec-1]
   mao.tools_xss.filter.XssFilter         : XSS filter [XssFilter] starting
4 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
5 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
6 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]
7 2022-10-30 21:39:46.002 INFO 18896 --- [nio-8080-exec-1]
   m.u.controller.StudentController       : 初始化完成
8 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
```

```

9 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
10 2022-10-30 21:39:46.016 DEBUG 18896 --- [nio-8080-exec-1]
   mao.tools_xss.filter.XssFilter         : XSS filter [XssFilter] stop
11 2022-10-30 21:39:46.028 DEBUG 18896 --- [nio-8080-exec-2]
   mao.tools_xss.utils.XssUtils          : xssfilter value after
   xssCleaantext/html,application/xhtml+xml,application/xml;q=0.9,image/webp,ima
   ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
12 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
13 2022-10-30 21:39:46.035 DEBUG 18896 --- [nio-8080-exec-2]
   mao.tools_xss.utils.XssUtils          : xssfilter value after
   xssCleaantext/html,application/xhtml+xml,application/xml;q=0.9,image/webp,ima
   ge/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
14 2022-10-30 21:39:46.038 DEBUG 18896 --- [nio-8080-exec-2]
   mao.tools_xss.filter.XssFilter         : XSS filter [XssFilter] stop

```

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/save.html'. The page has a light blue background. In the center, there is a registration form with a light green border. The form contains four input fields with labels in Chinese: '学生学号' (Student ID), '学生姓名' (Student Name), '学生性别' (Student Gender), and '学生年龄' (Student Age). The '学生姓名' field contains the payload '133<script>alert("xss攻击")' in red text. Below the input fields is a '提交' (Submit) button. The browser's developer tools are open at the bottom, showing the 'Elements' panel.

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
提交	



```

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

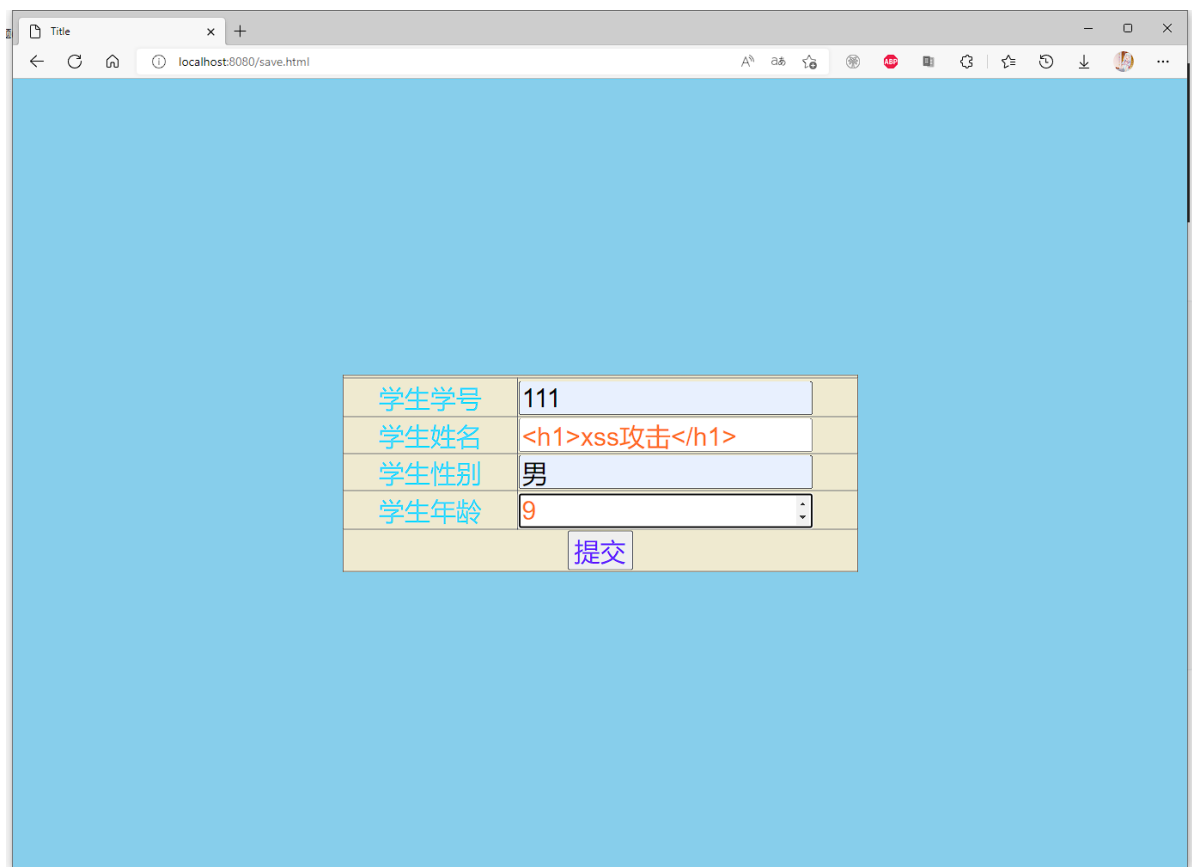
2022-10-30 21:41:38.242 DEBUG 18896 --- [nio-8080-exec-6] mao.tools_xss.filter.XssFilter      : XSS fiter [XSSFilter] stop

```

```

1 2022-10-30 21:41:38.214 DEBUG 18896 --- [nio-8080-exec-6]
   mao.tools_xss.filter.XssFilter      : XSS fiter [XSSFilter] starting
2 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
3 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>
4 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。
5 2022-10-30 21:41:38.236 DEBUG 18896 --- [nio-8080-exec-6]
   mao.tools_xss.utils.XssUtils        : xssfilter value after xssClean133
6 2022-10-30 21:41:38.240 INFO 18896 --- [nio-8080-exec-6]
   m.u.controller.StudentController    : 添加成功:
7 id: 111
8 name: 133
9 sex: 男
10 age: 8
11
12 2022-10-30 21:41:38.242 DEBUG 18896 --- [nio-8080-exec-6]
   mao.tools_xss.filter.XssFilter      : XSS fiter [XSSFilter] stop

```



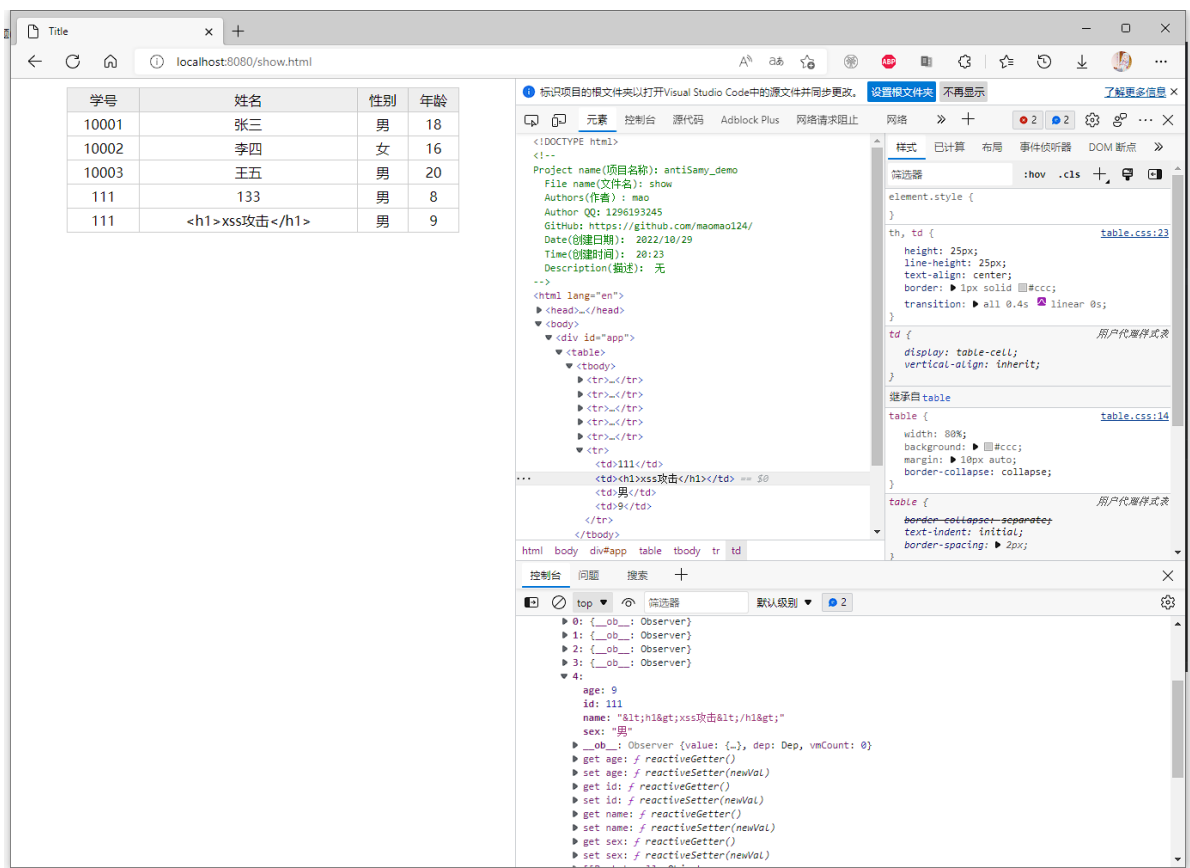
```
2022-10-30 21:42:45.468 INFO 18896 --- [nio-8080-exec-9] m.u.controller.StudentController : 添加成功:
id: 111
name: &lt;h1&gt;xss攻击&lt;/h1&gt;|
sex: 男
age: 9
```

```

1 2022-10-30 21:42:45.463 DEBUG 18896 --- [nio-8080-exec-9]
   mao.tools_xss.filter.XssFilter      : XSS fiter [XssFilter] starting
2 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
3 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>
4 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.
5 2022-10-30 21:42:45.467 DEBUG 18896 --- [nio-8080-exec-9]
   mao.tools_xss.utils.XssUtils        : xssfilter value after
   xssClean<h1>xss攻击</h1>
6 2022-10-30 21:42:45.468 INFO 18896 --- [nio-8080-exec-9]
   m.u.controller.StudentController    : 添加成功:
7 id: 111
8 name: <h1>xss攻击</h1>
9 sex: 男
10 age: 9
11
12 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



end

by mao

2022 10 30

