

# 标题模块

## 标题插入[POST]: /titleInsert/

[title\_name]      母猪的生产护理

[title\_type]      titleone\_title/titletwo\_title/titletheww\_title

[parent\_id]      3

前面为参数，后面为参数实例。

- 写给前端:  
这里三个属性要求前端提供，分别为标题名称、标题类型、和父标题的ID。

需要特别注意的是，一级标题的父标题为空，所以为空。

用户点击插入标题，则在其下插入一个文本框和确认键，点击确认之后请求接口，请求成功之后移除文本框和确认键，以正常的文本显示替代，最后重绑。

- 写给后端:

<input type="checkbox"/>	polls_title	★	📄 浏览	🔗 结构	🔍 搜索	➕ 插入	🧼 清空	🗑️ 删除	1	InnoDB	utf8_general_ci	32 KB	-
<input type="checkbox"/>	polls_titleone	★	📄 浏览	🔗 结构	🔍 搜索	➕ 插入	🧼 清空	🗑️ 删除	2	InnoDB	utf8_general_ci	32 KB	-
<input type="checkbox"/>	polls_titlethree	★	📄 浏览	🔗 结构	🔍 搜索	➕ 插入	🧼 清空	🗑️ 删除	0	InnoDB	utf8_general_ci	32 KB	-
<input type="checkbox"/>	polls_titletwo	★	📄 浏览	🔗 结构	🔍 搜索	➕ 插入	🧼 清空	🗑️ 删除	0	InnoDB	utf8_general_ci	32 KB	-

这四张表，层层绑定。

polls\_title 为论文的名称，一级标题绑定至此，polls\_titletwo 绑定至一级标题(即外键连接到)。

后端拿到父标题ID，获取父标题对象（一级标题的父标题为title，ID在session 中有）

然后利用 **标题类型** 和 **标题内容** 和 **父标题对象** 创建相应标题对象并保存

成功返回JSON 数据{code:200} 失败则{code:500}

## 标题编辑[POST]: /titleEdit/


[title\_name]      母猪的生产护理

[title\_type]      titleone\_title/titletwo\_title/titletheww\_title

[title\_id]      4

第一个为修改后的标题内容，第二个为该标题类型，前端将以上三个属性返回给后端即可

- 写给前端:  
**一些知识:** 要想拿到父标题的ID，自然是每个标题都有id,可以将每个h2标签的 id属性设置该标题的id,我们这样。

	#	名字	类型	排序规则	属性	空	默认	额外
<input type="checkbox"/>	1	<b>title_text</b>	varchar(100)	utf8_general_ci		否	无	
<input type="checkbox"/>	2	<b>order</b> 	int(11)			否	无	AUTO_INCREMENT
<input type="checkbox"/>	3	<b>up_grade_id</b>	int(11)			否	无	

这是title 的结构，我们在前端中这么调用它

```

<div id="main">
  {% for i in titleone %}
    <div class="titleone">
      <h2 class="titleone_title" id="{{ i.order }}">{{ i.title_text }}

```

这个是Django的模板引擎，比较好理解，应该看代码就能明白。

- 后端：这个不用写

## 标题删除[GET]: /titleDelete/

[title\_type]      titleone\_title/titletwo\_title/titleheww\_title

[title\_id]        4

这个不用多说