举个例子，比如我写了一个可以实现条纹相间的列表组件，发布后，使用者可以自定义每一行的内容或样式（普通的slot就可以完成这个工作）。而作用域插槽的关键之处就在于，父组件能接收来自子组件的slot传递过来的参数，具体看案例和注释。

**<!DOCTYPE html>**

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Vue作用域插槽</title>

<script src="https://cdn.bootcss.com/vue/2.3.4/vue.js"></script>

</head>

<body>

<div id="app2">

<!-- 组件使用者只需传递users数据即可 -->

<my-stripe-list :items="users" odd-bgcolor="#D3DCE6" even-bgcolor="#E5E9F2">

<!-- props对象接收来自子组件slot的$index参数 -->

<template slot="cont" scope="props">

<span>{{users[props.$index].id}}</span>

<span>{{users[props.$index].name}}</span>

<span>{{users[props.$index].age}}</span>

<!-- 这里可以自定[编辑][删除]按钮的链接和样式 -->

<a :href="'#edit/id/'+users[props.$index].id">编辑</a>

<a :href="'#del/id/'+users[props.$index].id">删除</a>

</template>

</my-stripe-list>

</div>

<script>

Vue.component('my-stripe-list', {

/\*slot的$index可以传递到父组件中\*/

template: `

<div>

<div v-**for**="(item, index) in items" style="line-height:2.2;" :style="index % 2 === 0 ? 'background:'+oddBgcolor : 'background:'+evenBgcolor">

<slot name="cont" :$index="index"></slot>

</div>

</div>

`,

props: {

items: Array,

oddBgcolor: String,

evenBgcolor: String

}

});

**new** Vue({

el: '#app2',

data: {

users: [

{id: 1, name: '张三', age: 20},

{id: 2, name: '李四', age: 22},

{id: 3, name: '王五', age: 27},

{id: 4, name: '张龙', age: 27},

{id: 5, name: '赵虎', age: 27}

]

}

});

</script>

</body>

</html>

