

实战：考试系统

Geek Pie 应用/网站开发 2017秋学期第4讲

回顾：第一个 ReactJS 应用

- 编译：Babel
- 打包：Webpack
- 框架：React、ReactDOM、.....
- 你的应用代码
 - ReactDOM.render(*Component*, *DOM_node*)
 - class SomeComponentName extends React.Component
 -
- 基本命令
 - npm install

考试系统：动手之前

- 如何存储和表示不同题型的问题和答案？
 - 选择题
 - 填空题
- 启封、试答……
- 如何实现限时作答
 - 时间同步问题
- 在线系统有什么传统考试做不到的地方？
 - 题库随机抽题

组件库

万不得已才用去操心 CSS,
预置有不少功能

各个组件库有各自特长

LoginForm

NumberInput

PasswordInput

RadioButton

SearchInput

Select

TextInput

Listing

List

ListPlaceholder

Table

TableHeader

Tiles

Visualization

AnnotatedMeter

Chart

Distribution

Legend

Map

TableHeader

A table header that can be used as a sort control.

Name ↓	Note
Alan	plays accordion
Chris	drops the mic
Tracy	travels the world

Properties

labels

[{node}, ...]

Header cell contents.

onSort

{function ({index}, {ascending})}

Function that will be called when the user clicks on a header cell. It is passed the index of the cell and which direction to sort in.

sortAscending

true|false

Indicates which direction the sort is currently going.

sortIndex

{number}

选择题：前后搭配

```
content = models.TextField()
choices = models.TextField()
correct_answer = models.TextField()
```

```
let choices = p.choices.split(/\r\n?|\ns/);

choices.map( c => {
  let choiceid = `problem-${p.id}-choice-${c[0]}`;
  return (
    <RadioButton type='radio' key={choiceid}
      id={choiceid}
      name={choiceid}
      data-p-id={p.id}
      label={c}
      value={c[0]}
      checked={comp.state.answers.get(p.id.toString())
        === c[0]}
      onChange={comp.handleRadioChange}/>
  )
})
```

```
handleRadioChange = (e) => {
  const pid = e.target.dataset.pId;
  const ans = e.target.value;
  this.setState( (prevState, props) =>
    {answers: prevState.answers.set(pid, ans)});
};
```


随机抽题：对应数据库模型

```
class ProblemCategory(models.Model):  
    name = models.CharField(max_length=30)  
    description = models.TextField()
```

```
class Problem(models.Model):  
    p_category = models.ForeignKey(ProblemCategory)  
    p_type = models.ForeignKey(ProblemType)
```

```
class ProblemSource(models.Model):  
    problem_category = models.ForeignKey(ProblemCategory) // 来源说明  
    number = models.PositiveIntegerField() // 题目数量
```

随机抽题：相关实现

```
class Quiz(models.Model):
    title = models.CharField(max_length=255)
    .....
    problem_source = models.ManyToManyField(ProblemSource)

    def getProblemSet(self):
        problem_set = []
        for p_source in self.problem_source.all():
            qualified_problems = Problem.objects.filter(p_category=p_source.problem_category)\
                .all().order_by('?')[:p_source.number]
            problem_set.extend(qualified_problems)
        return problem_set

    def getFullScore(self):
        ps = self.problem_source.all()
        pn = ps.aggregate(Sum('number'))['number__sum']
        return pn*self.points_per_problem
```


回顾：多对多

id	问题类型	需题数量
1	辐射	20
2	电气	15
3	有机化学	15
4	电气	35
...

Table problem_source

id	Title
1	2016年开学测试
2	培训成果验收
3	物质学院专题测试
4	基础测试
...

Table problem_source

多对多中间表

id	problem_source_id	quiz_id
1	1	2
2	1	3

随机抽题：对应数据库模型

```
class ProblemCategory(models.Model):  
    name = models.CharField(max_length=30)  
    description = models.TextField()
```

```
class Problem(models.Model):  
    p_category = models.ForeignKey(ProblemCategory)  
    p_type = models.ForeignKey(ProblemType)
```

```
class ProblemSource(models.Model):  
    problem_category = models.ForeignKey(ProblemCategory) // 来源说明  
    number = models.PositiveIntegerField() // 题目数量
```


限时收卷

```
token = secrets.token_urlsafe(10)
```

```
def QuizHandinView(request, quiz_id):
    import json
    answersheet = json.loads(request.body)

    try:
        assert(int(quiz_id) == answersheet['quiz_id'])
        quiz = Quiz.objects.get(pk=quiz_id)

        session = answersheet['session']
        session_rec = QuizStartRecord.objects.get(pk=session['re
        assert(session_rec.token == session['token'])

        correct_answers = []
        earned_points = 0

        answers = answersheet['answers'];

        for ans in answers:
            problem = Problem.objects.get(pk=ans['problem'])
            if problem.correct_answer == ans['choice']:
                earned_points += quiz.points_per_problem
                correct_answers.append(ans['problem'])

        res_dict = {
            'earned_points': earned_points,
            'correct': correct_answers,
            'passed': earned_points >= quiz.pass_threshold,
            'total_score': quiz.getFullScore()
        }
```

```
take_quiz.jsx
107 handleSubmission = (e) => {
108     let answer_collection = Array.from(
109         this.state.answers.entries()
110     ).map( kvp => {
111         let o = {};
112         o["problem"] = parseInt(kvp[0])
113         o["choice"] = kvp[1];
114         return o;
115     });
116
117     let anssh = {
118         quiz_id: parseInt(this.state.quiz_id),
119         session: this.state.session,
120         answers: answer_collection
121     }
122
123     console.log(anssh);
124
125     fetch(API_ROOT+'quiz/'+this.state.quiz_id+'/handin/', {
126         method: 'POST',
127         headers: {
128             "Content-Type": "application/json"
129         },
130         body: JSON.stringify(anssh)
131     }).then( r => r.text() ).then( j => {
132         console.log(j);
133         this.setState({answer_sheet: j});
134     });
```