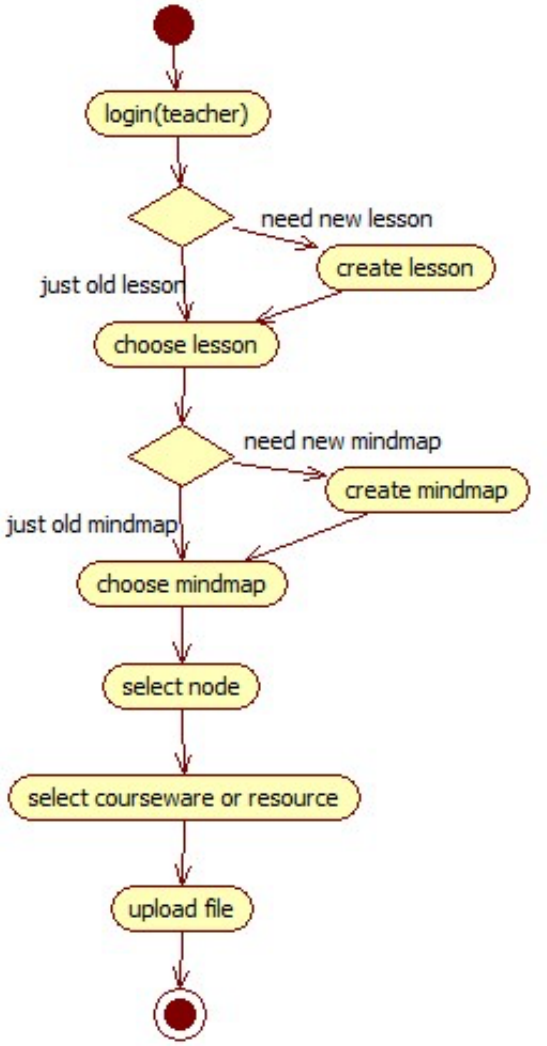
高级web 关键功能流程图

上传文件



> 文件：前端采用ng2-file-uploader第三方库实现上传文件功能。在需要使用到的组件中引用import { FileUploader } from 'ng2-file-upload'; 然后用后台对应的处理文件上传功能的url初始化一个FileUploader对象。

```
public url: string = 'http://13.67.110.158:8080/mindmap/upload/1';
public uploader: FileUploader = new FileUploader({url: this.url});
```

将文件的FileItem对象添加到uploader.queue队列里，就可以调用queue里的FileItem对象的各种函数来进行各种操作，如调用upload()函数来上传文件，调用cancel()来取消上传文件，调用remove()来将文件从上传队列里移除，即将文件的FileItem对象从uploader.queue里移除。
文件的描述存储在一个单独的列表里，没有利用uploader上传，而是和文件分别上传，利用标识文件的属性和文件配对。
链接：利用双向绑定，将html文件里input和ts文件里的属性绑定在一起。

```
<div class="modal-body">
    <li><input type="text" placeholder="链接名称" [(ngModel)]="linkname" name="linkname" class="form-control"></li>
    <li><input type="text" placeholder="链接描述" [(ngModel)]="linkcontent" name="linkcontent" class="form-control"></li>
</div>
```

点击确定按钮，将linkname和linkcontent的值上传到后台。

```
<button type="button" class="btn btn-primary" data-dismiss="modal" (click)="uploadLink()">确定</button>
```

文件的下载：

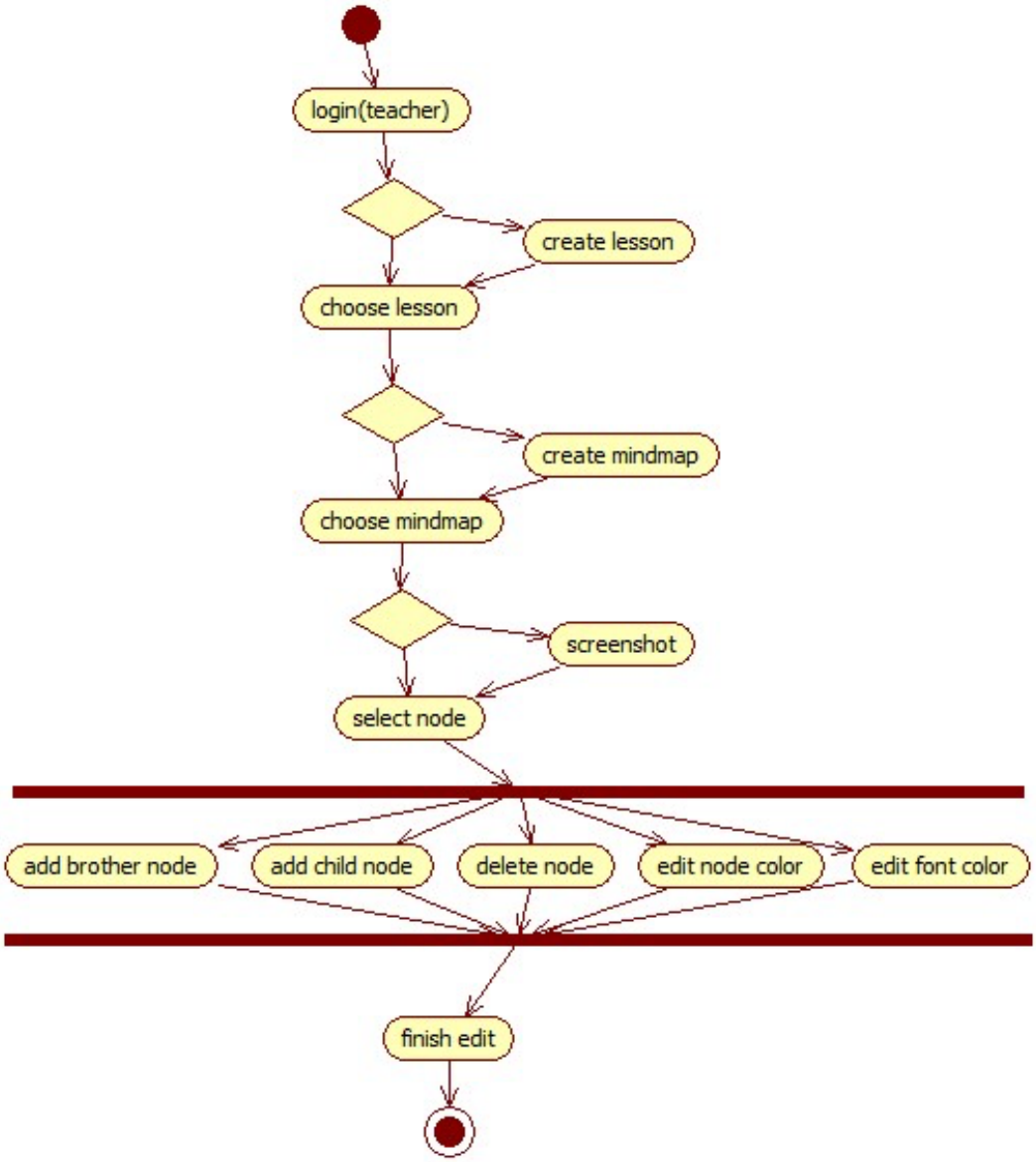
```
// Blob请求
requestBlob(url: any, data?: any): Observable<any> {
  return this.http.request( 'post', url, {body: data, observe: 'response', responseType: 'blob'});
}

// Blob文件转换下载
downFile(result: any, fileName: string) {
  var data = result.body;

  var blob = new Blob([data]);
  var objectUrl = URL.createObjectURL(blob);
  var a = document.createElement('a');
  a.setAttribute('style', 'display:none');
  a.setAttribute('href', objectUrl);
  a.setAttribute('download', fileName);
  a.click();
  a.addEventListener('click', function() {
    URL.revokeObjectURL(objectUrl);
    document.getElementById('download').remove();
  });
}
```

过程是将后台传回来的文件数据先转化为Blob对象，然后创建这个Blob对象的url，再通过创建一个隐藏的herf属性的值是Blob对象的url的a标签并点击它来下载文件。增加对a标签的click的监听，确保在文件下载完毕后才移除a标签和url。
文件或链接的展示： 后台返回存储文件或链接的信息的数组，遍历数组在html文件里展示。

思维导图的操作



思维导图编辑：
增加：增加分为增加兄弟节点和增加子节点，实现过程类似。首先利用jsmind自带的方法get_selected_node获取mindmap当前选中的节点，然后创建一个新的节点，再利用jsmind自带的方法add_node向目标节点增加子节点。增加兄弟节点和增加节点的不同在于，根节点无法被添加兄弟节点。

```
addChildNode() {
  const selected_node = this.mindMap.get_selected_node();
  if(!selected_node){
    alert('请先选择一个节点！');
    return;
  }
  const nodeid = jsMind.util.uuid.newid();
  const topic = '新节点';
  const node = this.mindMap.add_node(selected_node, nodeid, topic);
  this.items[this.currentMap] = this.mindMap.get_data("node_tree");
  this.saveMindMap();
}
```

```
addBrotherNode(e) {
  console.log(e)
  const selected_node = this.mindMap.get_selected_node();
  if(!selected_node){
    alert('请先选择一个节点！');
    return;
  }

  if(!selected_node.parent){
    alert('根节点无法被添加兄弟节点！');
    return;
  }
  const nodeid = jsMind.util.uuid.newid();
  const topic = '新节点';
  const node = this.mindMap.add_node(selected_node.parent, nodeid, topic)
  this.items[this.currentMap] = this.mindMap.get_data("node_tree");
  this.saveMindMap();
}
```

删除：删除当前选中的节点，同样利用jsmind自带的get_selected_node获取当前节点，然后再利用jsmind自带的remove_node移除节点。

```
removeNode() {
  const selected_id = this.mindMap.get_selected_node();
  if(!selected_id){
    alert('请先选择一个节点！');
    return;
  }
  if(!selected_id.parent) {
    window.alert('根节点无法被删除！');
    return;
  }
  this.mindMap.remove_node(selected_id);
  this.items[this.currentMap] = this.mindMap.get_data("node_tree");
  this.saveMindMap();
}
```

修改颜色：修改颜色分为修改节点颜色和字体颜色，这里我很快在jsmind文档里找到了修改颜色的方法set_node_color，通过数据绑定获取界面中修改后节点颜色和字体颜色，进行修改。

```
changeFontColor() {
  const selected_node = this.mindMap.get_selected_node();
  console.log(selected_node);
  console.log(this.fontColor)
  if(!selected_node){
    alert('请先选择一个节点！');
    return;
  }
  this.mindMap.set_node_color(selected_node.id, null, this.fontColor);
  this.items[this.currentMap] = this.mindMap.get_data("node_tree");
  this.saveMindMap();
}
```

截图：截图用的是jsmind自带的截图功能，代码只有一行。

```
mapShoot() {
  // this.mindMap.show(this.currentMap);
  this.mindMap.screenshot.shootDownload();
}
```