EnglishScoreService.run()的流程

- 传入的参数：long startWeekTime, int lessonId,
- 返回的参数：空

1. UserMissionLogic.getUserMissions(startweekTime) :获取本周内所有的user, missions列表
2. EnglishScoreService.runBatch()：分batch对每个user和missions进行计算英语分

EnglishScoreService.runBatch()的流程

- 传入的参数：Map<Integer, Set> user2MissionIds, long startWeekTime
- 返回的参数：空

1. userEnglishScoreCollectManager：多线程获取用户能力模型信息和英语力的信息；
    1. 用参数user2MissionIds[获取的userId对应的missions的map]，userInfoLogic[空的]，englishScoreCalculatedLogic[空的]，SKILL_MODEL_READER_THREADS_NUM[读skill_model数据库的线程数]来初始化一个userEnglishScoreCollectManager实例
    2. 然后调用userEnglishScoreCollcetManager.getUserEnglishScores()接口，来获取每个用户的EnglishScore
    3. 返回的结果是Map<Integer, List>，里面的key是userId，value是List englishScores，里面的EnglishScoreDbo中的skillType为1~5记录的是用户在每个abilityType[也叫skillType]上的得分，skillType=6，计算的是用户的能力值分数【前序level的基础分+当前level的得分】
2. englishScoreWriterManager：来批量存储用户的英语力
    1. 用参数Map<Integer, List> user2EnglishScores[用户ID，和它对应的EnglishScoreDbo数据的结构形式], EnglishScoreDbStorage englishScoreDbStorage[存储英语分的数据库],Int Insert_BATCH_SIZE[插入的batch的数值大小], Int ENGLISH_SCORE_WRITER_THREADS_NUM[写入数据库的线程数]来初始化一个englishScoreWriterManager
    2. 初始化时，还调用了getBatches(Map<Integer, List> user2EnglishScores, int batchSize)函数
        1. getBatches()函数实现的主要功能如下：
        2. 将List按照batch大小存在List<List> result中【注意EnglishScoreDbo中有userId】.
        3. 最后返回List<List> result，里面的每个元素List都是batch大小
    3. 用 englishScoreWriterManager.saveEnglishScores()来向数据库中插入数据

UserEnglishScoreCollectManager主要实现的功能：

- 调用这个接口时，首先会按照传入的参数进行初始化
- getUserEnglishScores()：获取用户的英语分
    1. this.initTasks()这个函数，在初始化userEnglishScoreCollectManager这个实例时就会调用，用来初始化this.readerTasks。

2. 建立固定的几个readThread，每个readThread都会有一个run()函数来计算用户的英语分，是用一个全局变量taskIndex来控制每个线程取得user的信息
    1. 里面有个private类,userInfoTask：用来存储每个userid的信息[userId，set<>missionIds, startWeekTime, startDate,]，最后就都存在this.readerTasks这个list里面了
    2. UserInfoTask.doTask()函数实现的功能是：调用UserInfoLogic.getUserInfo(userId, missionIds, startWeekTime, startDate)，然后返回userInfo这个实例【存储了用户的信息】。
        1. UserInfoLogic.getUserInfo实现的主要功能是：
            1. 传入的参数有：int userId. set missionIds, long startWeekTime, int startDate
            2. 首先根据missionId获取系统课的mission，然后获取本周最后一节课的mission，继而获取用户的level、edition、learnedKeypointIds
            3. 最后初始化userInfo这个实例
                1. userInfo这个类还有的属性主要有userId、startWeekTime、mission、episode、level、edition、learnedKeypointIds、skillModels；
            4. 然后执行函数initUserSkillModels(userInfo)
                1. initUserSkillModels()这个函数实现的主要功能是：
                2. 首先根据用户的userId和keypointId结合生成一个新的MD5值，这些所有的MD5值存在result这个list中
                3. 再用skillModelDbStorage.getSkillModelListByIds(result)去数据库中根据这些result中的MD5值检索
                    1. skillModelDbStorage.getSkillModelListByIds(result)这个函数实现的主要功能是：
                        1. 按照id【就是result中的MD5值】去数据库ability_skill_model里面查找对应的记录，然后存储下来，存储的数据结构为skillModelDbo这个类
                            1. skillModelDbo这个类主要有这几个属性：long id[由userId和keypointId结合算的MD5值]， int userId, int basicpointId【其实就是keypointId】, String skillModel【是一串数字，比如1,1,3,3;4,2,6,33】,**这个skillModel最后转化成AbilityModel，AbilityModel的属性有四个：int abilityType, int startCount, int questionCount, List startHistory,这里可以用调用的是SkillTypeUtils.parseSkillModelString方法可以将它切分成AbilityModel的形式**
                        2. 最后返回skillModelList，这个List里存储的是SkillModelDbo的实例
                4. 最后执行userInfo.setSkillModels()，来给userInfo的skillModels参数赋值
            5. 最后返回userInfo这个实例
    3. englishScoreCalculatedLogic.calculateEnglishScorePerUser(userInfo)：用来计算每个用户的englishScore。

1. englishScoreCalculatedLogic.calculateEnglishScorePerUser(uesrInfo)这个函数实现的主要功能是：
2. 传入的参数：userInfo这个实例
3. 首先根据userInfo来获取它的属性值skillModels，也就是List
4. 然后通过函数getBasicPoint2Ability2Score(skillModels)来得到在不同考点，不同能力上的得分
    1. getBasicPoint2Ability2Score(skillModels)这个函数实现的功能是【这里的basicPoint其实就是keypoint，我的理解就是每个keypoint对应着不同的ability，所以我们最后需要计算在每个keypoint下的每个ability的score】：
    2. 首先对于每一个skillModelDbo获取他们的keypointId，
    3. 然后调用接口将每个skillModelDbo中的skillModel由一个string转化为List，**这里可以用调用的是SkillTypeUtils.parseSkillModelString方法**
        1. AbilityModel的属性有四个：int abilityType, int startCount, int questionCount, List startHistory,
    4. 然后对于每一个abilityModel，获取他们的int abilityType、List startHistory，然后调用接口calculateScoreWithStartHistory(starHistory)计算分值
        1. calculateScoreWithStartHistory(starHistory)这个函数实现的功能是：
        2. 根据weight设置的权重，和starHistory中的对位相乘在相加最后得到的一个分值，然后将这个分值返回
    5. 再用Map<Integer, Double> score，来存储每个abilityType下的score，
    6. 最后再用Map<Integer, Map<Integer, Double>> basicPoint2Ability2Score，来存储每个keypoint下的每个abilityType的score。
    7. 最终返回Map<Integer, Map<Integer, Double>> basicPoint2Ability2Score
5. 然后再用calculateEnglishScorePerUser(userInfo, basicPoint2Ability2Score, userInfo.getStartWeekTime())这个函数进行计算
    1. 注意calculateEnglishScorePerUser(userInfo, basicPoint2Ability2Score, userInfo.getStartWeekTime())和上面同名函数传入的参数不同，这个函数传入的参数更多了，它实现的功能主要是：
    2. 首先获取用户当前level和edition下的基础分int baseAbilityScore【由读库获取所有前序Level的满分，然后再相加得到当前level的基础分】
    3. 然后在调用calculateScores(basicPoint2Ability2Score, baseAbilityScore)，计算用户的6个得分，其实现的主要功能如下：
        1. 传入的参数是basicPoint2Ability2Score[每个keypoint下的每个abilityType的score], baseAbilityScore[当前level的基础分]
        2. 首先对于每一个keypoint，获取它们的Map<Integer, Double> ability2Score【对应的每个ability下的score】
        3. 然后将这些每个ability下的score相加再除以ability的数目，即求得在ability下的平均值，然后将这个平均值存在List basicPointScores里【存储的是每个keypoint下的得分】。

4. 最后将所有的basicPointScores里的分值都相加，然后再加上当前level下的基础分baseAbilityScore，得到的就是用户的能力值double totalScore
5. 接下来会调用SkillTypeUtils这个类，下面看下这个类的主要功能：
    1. SkillTypeUtils里的ABILITY_MAP--Map<Integer, String>，里面存储的是ability的编号和它对应的字符串
    2. SkillTypeUtils里的ENGLISH_SCORE_KEYS--Set，里面存储的是ABILITY_MAP的key【从1~5】，另外还存了个6【6代表英语力得分】
6. 然后计算用户所有keypoint下的每个ability的defender均值
7. 最后返回Map<Integer, Double> result，里面key值为1~5存储用户在每个ability下的得分的均值，key为6存储用户的能力值得分【基础分+当前level上的得分】

4. 然后调用transEnglishScore(userId, startWeekTime, finalScores【就是第4步返回的result】)，将数据转换成数据库的格式，也就是存成List的形式
    1. transEnglishScore实现的主要功能如下：
    2. 将finalScores【Map<Integer.Double>,里面key值为1~5存储用户在每个ability下的得分的均值，key为6存储用户的能力值得分【基础分+当前level上的得分】】中存储的数据转换成数据结构EnglishScoreDbo.
    3. 最后返回一个List englishScores
5. 然后如果不是在计算历史数据，就进行批量更新classId **这块没太看懂**
6. 最后返回List englishScores，里面的EnglishScoreDbo中的skillType为1~5记录的是用户在每个abilityType[也叫skillType]上的得分，skillType=6，计算的是用户的能力值分数【前序level的基础分+当前level的得分】

6. 如果是第一周的话，就传入的basicPoint2Ability2Score为一个空的HashMap<>,也得到上面的那个List englishScores，里面的EnglishScoreDbo中的skillType为1~5记录的是用户在每个abilityType[也叫skillType]上的得分，skillType=6，计算的是用户的能力值分数【前序level的基础分+当前level的得分】
7. 最后返回的就是List englishScores，里面的EnglishScoreDbo中的skillType为1~5记录的是用户在每个abilityType[也叫skillType]上的得分，skillType=6，计算的是用户的能力值分数【前序level的基础分+当前level的得分】

4. 最后将上面得到的List englishScores 存入ConcurrentHashMap<Integer, List> user2EnglishScores里，里面的key为userId，value为上面得到的List englishScores

3. 返回的结果是this.user2EnglishScores--Map<Integer, List>，里面的key是userId，value是List englishScores，里面的EnglishScoreDbo中的skillType为1~5记录的是用户在每个abilityType[也叫skillType]上的得分，skillType=6，计算的是用户的能力值分数【前序level的基础分+当前level的得分】

EnglishScoreWriterManager.saveEnglishScores()实现的主要功能如下：

1. 在初始化englishScoreWriterManager这个实例时，传入了参数ap<Integer, List> user2EnglishScores[用户ID，和它对应的EnglishScoreDbo数据的结构形式], EnglishScoreDbStorage englishScoreDbStorage[存储英语分的数据库],Int Insert_BATCH_SIZE[插入的batch的数值大小], Int ENGLISH_SCORE_WRITER_THREADS_NUM[写入数据库的线程数]
2. 对于每一个numThread【第四个参数，写的线程数目】，都构建一个WriterThread()
    1. WriterThread()线程都会执行一个run()函数，run函数主要实现的功能如下：
        1. run()函数传入的参数为空
        2. 首先调用getBatch()函数，获取一个batch大小的List batch；
        3. 然后调用函数englishScoreDbStorage.insertOrUpdateBatch(batch)将数据插入数据库
            1. englishScoreDbStorage.insertOrUpdateBatch(batch)实现的主要功能如下：
            2. 如果在表zebra_english_score里面有当前id的记录，那就执行updateBatch，否则就执行insertBatch操作。
        4. 最后返回为空，也就是run函数的主要功能是将获取的List batch插入数据库