32 我们一起学时序图

更新时间：2020-06-03 16:06:45

> 每个人的生命都是一只小船，理想是小船的风帆。——张海迪

1. 前言

前一节我们学习了类图，我们知道通过类图可以表示类之间的依赖、关联和泛型关系。

那么如何表示类之间的调用路链呢？

这就需要用到我们本节所要讲到的时序图。

《手册》设计规约章节，对时序图有下面的规定：

> 【强制】如果系统中某个功能的调用链路上的涉及对象超过 3 个，使用时序图来表达并且明确各调用环节的输入与输出。

那么我们要思考几个问题：

- 什么是时序图？
- 时序图的使用场景有哪些？
- 如何画时序图？

接下来本节将重点研究这几个问题。

2. 时序图是什么？为什么需要它？

当你真正清楚一个概念，分析该概念和其他类似概念的区别之后，你就很容易知道为什么要用这个知识或者技术，在适合的场景才更容易想起来用它。

接下来我们将学习时序图是什么？为什么需要用时序图？

2.1 时序图的概念

时序图又称顺序图，属于 UML 行为图， 时序图主要用来表示对象之间的交互顺序。

> 时序图反映了一系列对象的交互与协作关系，清晰立体地反映系统的调用纵深链路。

时序图的核心元素包括：对象（Actor）、生命线（Lifeline）、控制焦点（Focus of control）、消息（Message）等。

2.2 时序图的分类

根据《大象：Thinking in UML》中关于时序图的相关描述，我们可知，时序图使用场景分为三类：业务模型时序图、概念模型时序图和设计模型时序图。

业务模型时序图用于为领域模型中的业务实体交互建模，目标是实现业务用例。

概念模型时序图依据业务模型场景采用分析类来重新绘制一遍，目标同样是实现业务用例。因为分析类本身代表了系统原型，所以这时的时序图已经有了实现的影子。

设计模型时序图使用设计类作为对象绘制，目标是实现概念模型中的某个事件流，一般以一个完整交互为单位，消息细致到方法级别。因为设计模型时序图工作量实在太大，不需要为每一个交互都绘制时序图，但一定要有足够的概念模型时序图来支撑需求与实现之间的过渡。

2.2 为什么要用时序图？

我们可以将时序图和其它 UML 图进行对比，来理解这个问题。

在需求分析和软件设计阶段，用例图、类图、时序图、活动图等使用比较多。

如果我们想表示多个对象之间的（时间）顺序，表示多个类之间的调用链路，那么用来表示参与者或系统之间关系的用例图显然不适合；而描述类的属性和方法类图以及类之间的依赖、关联和泛型关系的类图也显得力不从心；而活动图更侧重活动的流转，视角完全不同。因此就需要有一种图形可以从粗和细的粒度上表达表示用例之间的顺序关系，这就是时序图出现的重要原因之一。

这里也说明了脱离场景无法谈好坏。

脱离某个具体场景，你敢说用例图或者时序图就是 UML 图里最好的吗？如果真的是为啥还需要那么多种 UML 图呢？

真正的牛人不仅仅是某个具体知识学得很透彻，更是能够根据特定场景选择最适合的方案的人。

3. 画时序图的准备

3.1 了解基本组件

如下图所示，时序图的核心元素包括：参与者（Actor）、生命线（Lifeline）、控制焦点（Focus of control）、消息（Message）等。

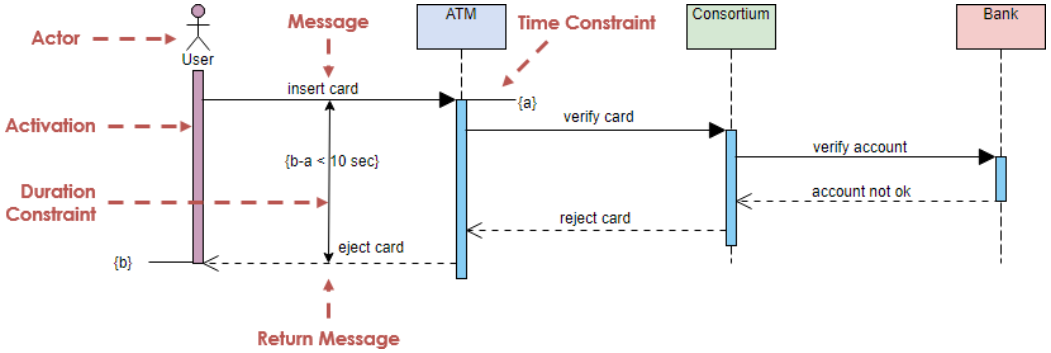
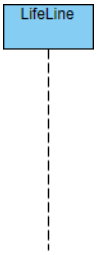


图 1： 时序图介绍

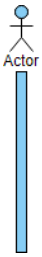
3.1.1 生命线

生命线 是一条垂直的虚线，用来表示交互的独立个体，表示序列图中的对象在一段时间内的存在。



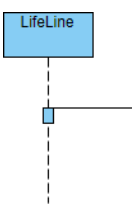
3.1.2 参与者

参与者，可以指人，外部其他系统，还可以指子系统。



3.1.3 控制焦点

控制焦点　表示对象执行一项操作的时期，是时序图中表示时间段的符号，用覆盖在生命线长矩形表示，矩形的顶部和箭头对齐，分别表示开始和结束时间，因此矩形的长度也表示持续的时间。



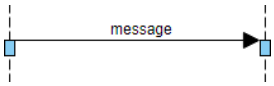
3.1.4 消息

消息（Message）是对象之间的一种通信机制。

通常为了提高可读性，时序图的第一个消息总是从顶端开始，一般位于图的左上角。

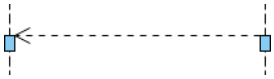
调用消息（Call Message）

调用消息对目标生命线的一次调用。通常使用实心箭头表示同步调用，使用左右朝向的开放箭头表示异步调用。



返回消息（Return Message）

返回消息表示目标对象传递给调用者的消息。使用朝向调用者的虚线开放箭头表示。



自调用消息（Self Message）

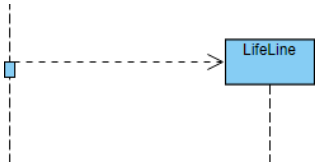
表示对当前生命线的调用消息，相当于一个对象的 A 函数调用该对象的 B 函数。

递归消息（Recursive Message）

递归消息表示对当前生命线的调用消息，相当于一个对象的 A 函数内部再次调用 A 函数。

创建消息（Create Message）

创建消息表示目标生命的实例化消息，即初始化一个对象。使用朝向初始化对象的带虚线开放箭头表示。

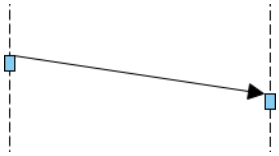


销毁消息（Destory Message）

销毁消息表示破坏目标对象生命周期的请求。使用叉号结束生命线。

持续消息（Duration Message）

持续消息显示消息调用的两点之间的距离。



3.1.5 注释

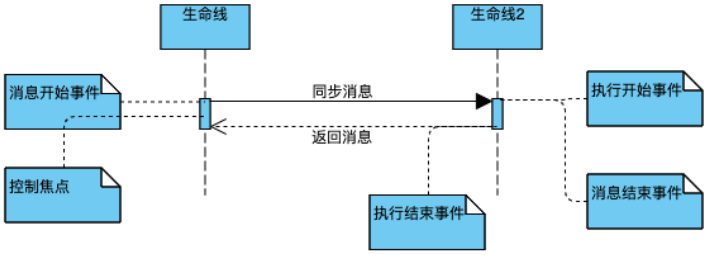
可以将注释附着在各种元素上，注释不包括时序的语义，但可能包含对建模非常有用的信息。

3.2 消息和控制焦点

事件是指发生事情的交互的任意一点。

控制焦点也称为执行的发生，它是生命线上的一个窄的长的矩形。

如下图所示，同步消息箭头起始位置为消息开始事件，箭头所指向的位置为消息结束事件。控制焦点的顶端表示执行开始事件，底端表示执行结束事件。



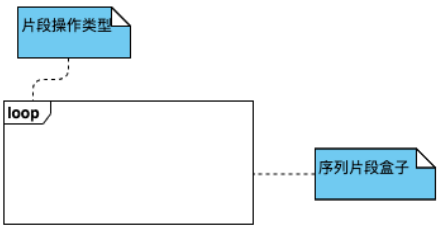
3.3 序列片段（Sequence Fragments）

UML 2.0 引入了序列片段，通过它可以更轻松地精确创建和维护时序图。

序列片段也成为组合片段，使用一个框来表示，它包含时序图的一部分交互。

序列片段左上角的符号表示片段的操作类型。

片段的主要类型包括： ref, assert, loop, break, alt, opt, neg。



片段操作类型	片段类型介绍
alt	只执行条件为真的片段
opt	可选，仅在条件为真时才执行
par	并行：每个片段并行执行
loop	循环：指此片段可以多次执行

片段操作类型	片段类型介绍
region	关键区（临界区）：一次只能有一个线程执行
neg	否定：片段显示无效的交互
ref	参考：指在另一个图上定义的交互
sd	时序图：用于包围整个时序图

4. 画时序图

4.1 画图一般步骤

- 确定交互的上下文（即场景，如创建订单、购买商品、退货等）
- 识别参与过程的交互对象
- 为每个对象创建生命线
- 从初始消息开始，依次画出后续消息
- 为了清晰起见，调你家所需的返回消息
- 结合上述序列片段的几种操作类型，考虑消息的复杂逻辑
- 对时序图进行修改、美化

4.2 示例

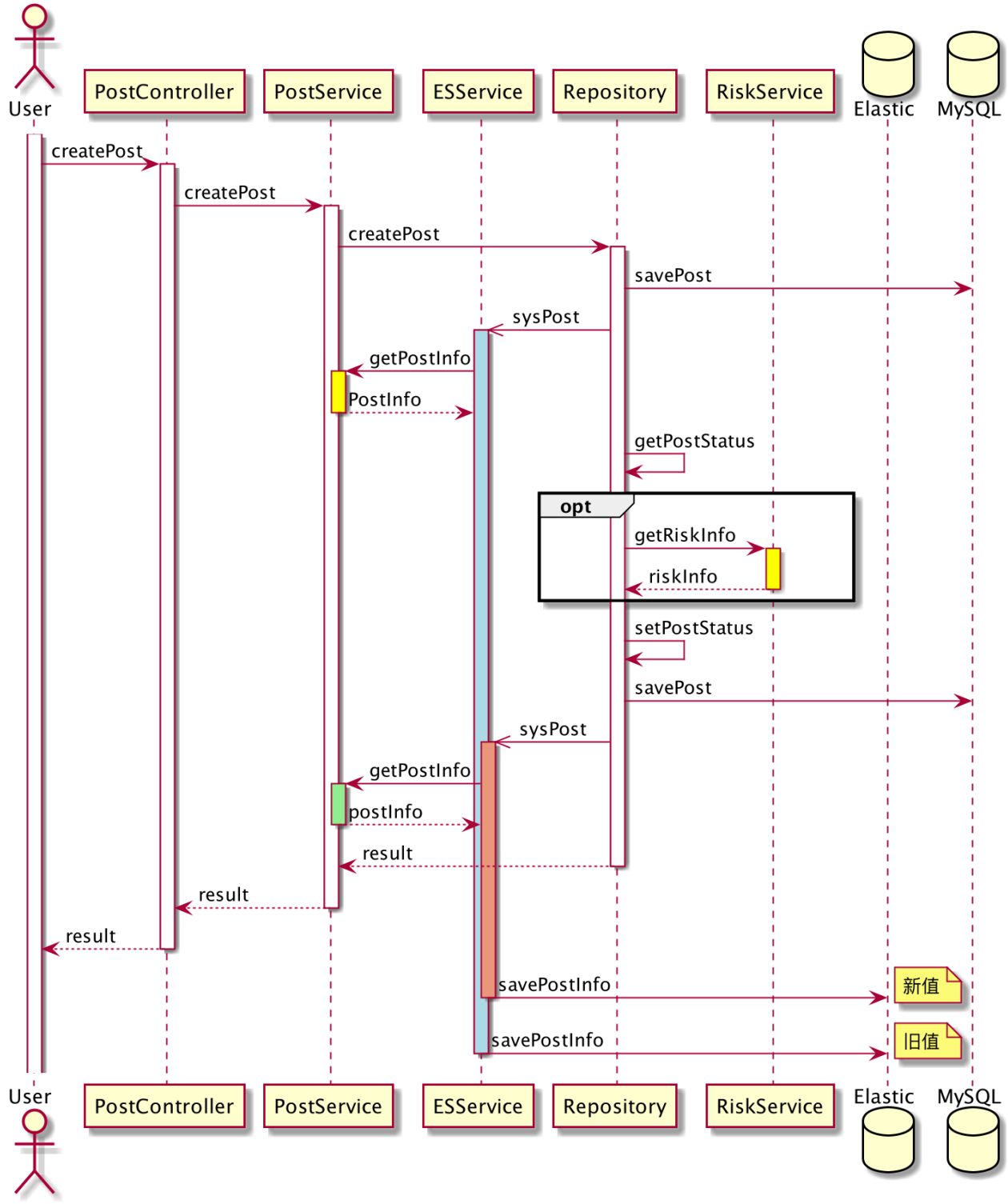
为了更好地说明时序图的优势，我先用文字描述一遍，再用时序图描述一遍同一件事，大家自行对感受一下。

开发中遇到一个异步同步数据导致旧值覆盖新值 BUG，分别使用文字和时序图对该问题描述。

文字描述：

> 发表朋友圈时可以选择文本或图片，如果只有文本不需要风控信息，如果发图片则需要检查风控信息。
>
> 点评状态最初为初始化状态，查询风控信息后更新为通过或者风控状态。
>
> 由于方案设计的不合理，在初始状态时存储一份，如果是文本则直接通过，如果是图片则查询风控信息后再存储一次。
>
> 而且为了查询性能，支持复杂的数据结构，支持超长度文本类型，支持丰富的查询需求，发送朋友圈状态数据存储到本地数据库后，还要异步通知存储服务，存储服务反查该朋友圈动态的信息存储一份 。
>
> 由于两次保存间隔极短，而通知存储服务反查是异步的，两次消息的顺序性没法保证，而且即使消息顺序抵达，未必先抵达的消息可以先处理，优先存储到 Elstic 中，所以可能存在 Elstic 旧值覆盖新值的情况，出现 BUG。

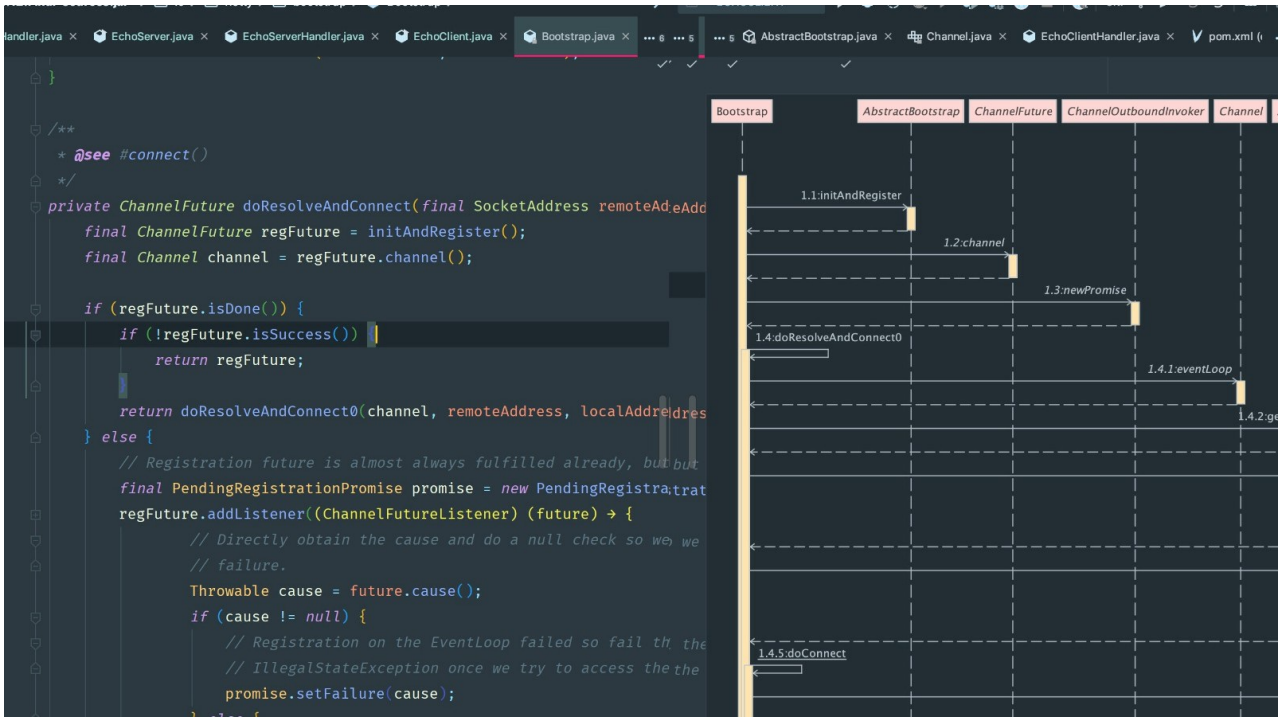
为了更清楚地描述该 BUG ，我们画出对应的时序图：



4.3 其它

4.3.1 根据插件学时序图

大家还可以使用 IDEA SequenceDiagram 插件来学习时序图，它可以根据代码调用链路自动生成时序图。

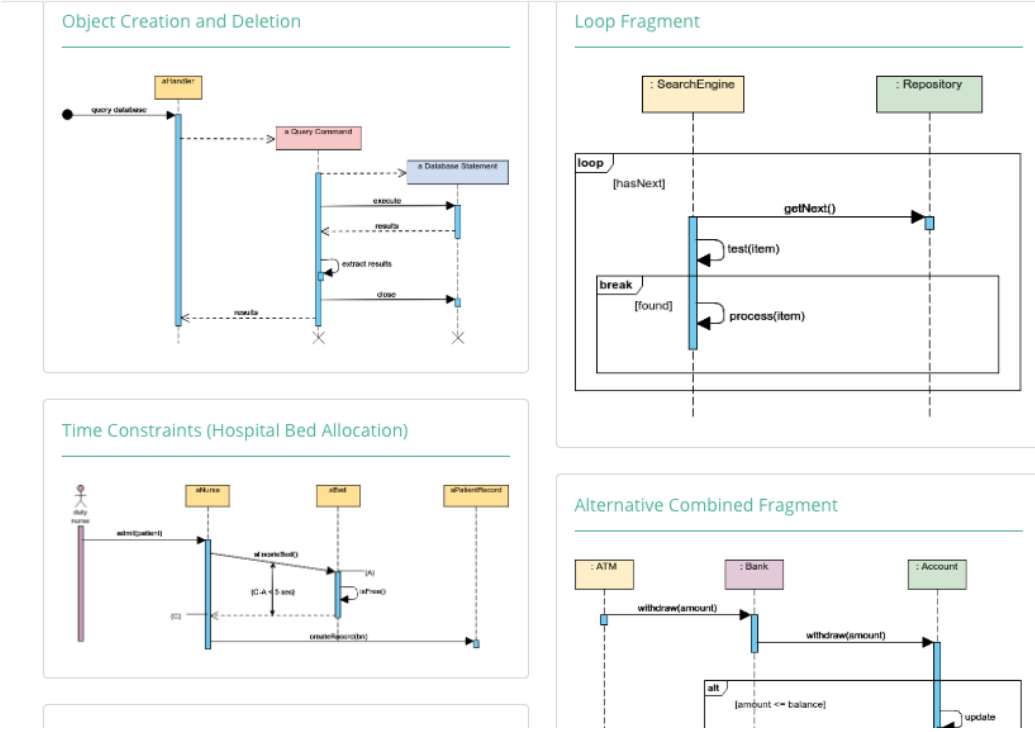


该插件还支持点击生命线顶部的类名，点击请求的函数名称跳转到对应函数中。

这对我们学习时序图、熟悉自己项目代码和学习开源代码都有极大的帮助。

4.3.2 根据案例学时序图

visual paradigm 给出了大量的时序图范例，大家可以参考学习。



5. 总结

本节主要学习了时序图的定义，时序图的使用场景，时序图的核心组件和画图步骤，并给出了一个范例。

介绍了两种高效学习时序图的方式，即使用时序图插件和看时序图案例学时序图，希望大家多学习多实践。

下一节我们将学习状态机图。

6. 课后练习

回忆一下自己去 ATM 机取款的步骤，画出对应的时序图。

参考资料

阿里巴巴与 Java 社区开发者.《 Java 开发手册 1.5.0》华山版. 2019

维基百科 - 时序图

顺序图的语法和功能

谭云杰.《大象：Thiking in UML》. 中国水利水电出版社. 2012

张传波.《火球：UML 大战需求分析》. 中国水利水电出版社. 2012

}

31 我们一起学类图

33 我们一起学状态机图