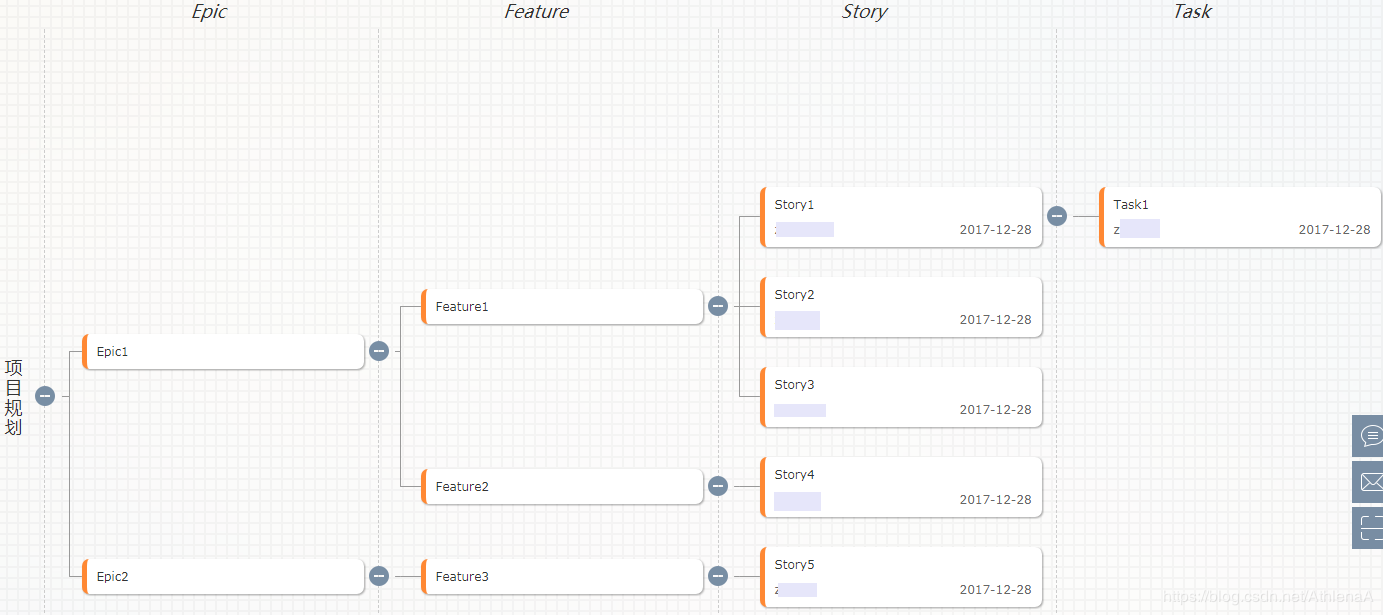
**项目规划样例**



**Epic**

中文通常翻译为史诗，指公司的关键战略举措，可以是重大的业务方向，也可以是重大的技术演讲。企业通过对Epic的发现、定义、投资、管理和落地达成，使得企业的战略投资主题得以落地，并获得相应的市场地位和回报。

Epic的粒度比较大，需要分解为Feature，并通过Feature继续分解细化为User Story来完成最终的开发和交付。

Epic通常持续数月（months），需要多个迭代才能完成最终的交付。Epic应该对所有研发人员可见，这样可以让研发人员了解他们交付的Story承载怎样的战略举措，让研发人员能更好的理解其工作的价值。

Epic通常和公司的经营、竞争力、市场环境紧密相关，举例如下：  
例1  
市场差异化：用户体验全面超越竞争对手  
例2  
更好的解决方案：新增支持工业互联网的解决方案  
例3  
增加收入：产品需要在下个财季增加100万付费用户  
例4  
重大技术方向：产品需要全部切换为容器

**Feature**

中文通常翻译为特性，代表可以给客户带来价值的产品功能或特性。

Feature向上承接Epic，向下分解为User Story。相比Epic，Feature更具体形象，客户可以直接感知，通常在产品发布时作为ReleaseNotes的一部分发布给客户。

Feature通常持续数个星期（weeks），需要多个迭代完成交付。

Feature应该对客户都有实际的价值，特性的描述通常需要说明对客户的价值，与产品的形态、交付模式有关，举例如下：  
推荐模板：用户<角色> …希望<结果>… 以便于<目的>

例1  
用户A希望提供导入、导出功能，以便于用户批量整理数据，更高效。  
例2  
用户B希望提供超期的邮件通知，以便于用户及时处理任务。  
例3  
用户C希望优化鼠标拖动的体验，以便于让用户操作更快。  
例4  
用户D希望增加昵称功能，让用户更个性化。

**Story**

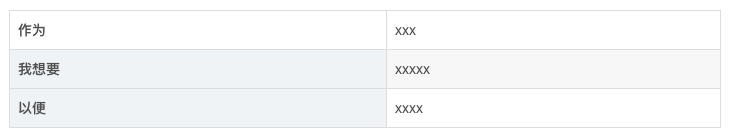
中文通常翻译为用户故事，User Story的简称。是从用户角度对产品需求的详细描述，更小粒度的功能。Story承接Feature，并放入有优先级的backlog中，持续规划、滚动调整优先级，始终让高优先级的Story更早的交付给客户。

story 包括三部分：用户故事卡片、详细描述、验收标准。

**（1）用户故事卡片**

三要素：用户、任务和活动、目标

框架：



实例：



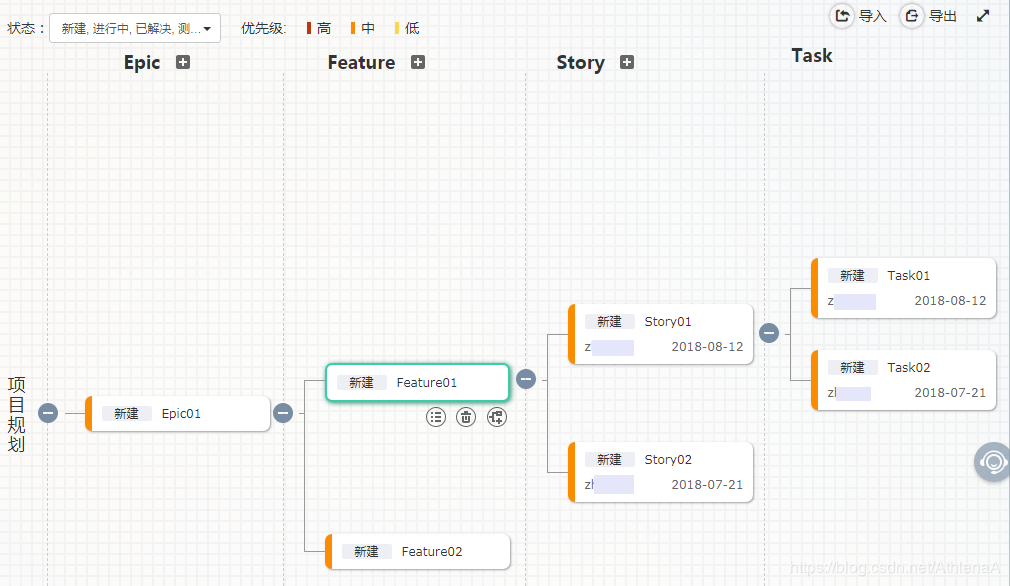
**（2）详细描述**

对如何实现“我想要”的详细描述。

**（3）验收标准（**如何确定Story已经完成？）

通过验证验收标准里的一系列内容，就能验证实现符合story的需求。

**操作步骤**  
1.进入项目查看页面。  
2.单击“工作 > 项目规划”菜单，如下图所示：



**项目规划中的参数项说明：**状态：项目规划页面筛选，可以通过勾选感兴趣的工作项状态按需显示工作项。  
优先级：三种颜色分别代表“高”、“中”、“低”优先级。  
导入：既可以下载模板填写关键字后导入，也可以从其它项目导出的xls,xlsx导入。  
导出：将被勾选“状态”的工作项导出压缩包，解压后为xlsx文件。  
+/-：点击可以展开或隐藏该类型下所有子工作项。  
查看详情：查看此工作项详情，可以修改详细内容。  
删除：删除此工作项及其所有子工作项。  
插入子主题：输入名称即可新增一条子工作项，其它信息可以新建后单击名称继续编辑。  
说明：  
工作项“处理人”名称，新建时默认是该工作项“创建人”；可以通过查看工作项详情中设置其它成员为“处理人”。  
工作项截止日期，默认是创建该工作项时间的第二天；可以通过查看工作项详情编辑工作项具体起止时间。  
添加Epic类型工作项（可添加多个），设置Epic工作项名称（如“Epic1”）后回车。

**简单理解：**

* Epic是公司重要战略举措；
* Feature是对你的用户有价值的功能；
* Story是分解的细粒度的开发交付的内容，是用户的细分场景；
* Task是完成需求的过程性的工作。

综合：对于软件企业，是实现从战略举措到战术执行落地的分层设计。

**feature和story的对比：**

* feature是外部可以感知的内容，story未必
* feature一般跨迭代跨团队，而story是一个迭代内部和团队内部
* feature针对多角色多维度多场景的功能描述，story关注一个具体的角色和场景
* feature一般不从具体的用户描述，更抽象，价值更大

**敏捷开发**

**瀑布模型 vs. 敏捷开发**

**瀑布模型**

即常规的开发流程：

1. 需求分析
2. 设计
3. 编码
4. 测试
5. 交付

只要是在“瀑布模型”下，返工就无法根本避免，因为来自于以下事实：

* 只要有沟通，就会有信息的变形，因此《需求分析》是不可靠的。
* 即使《需求分析》是可靠的，随着项目的进展，竞争对手的出现，需求也可能过时。
* 瀑布模型，这种自上而下的开发方式，无法响应这个快速变化的时代。

**敏捷开发**

《敏捷宣言》：  
个体和互动 高于 流程和工具；  
工作的软件 高于 详尽的文档；  
客户合作 高于 合同谈判；  
响应变化 高于 遵循计划。

也就是说，尽管右项有价值，我们更重视左项的价值。

**【敏捷开发的定义-百度百科】**敏捷开发以用户的需求进化为核心，采用迭代、循序渐进的方法进行软件开发。在敏捷开发中，软件项目在构建初期被切分成多个子项目，各个子项目的成果都经过测试，具备可视、可集成和可运行使用的特征。  
换言之，就是把一个大项目分为多个相互联系，但也可独立运行的小项目，并分别完成，在此过程中软件一直处于可使用状态。

**【敏捷冲刺】**因为世界是易变的，因此敏捷开发就不提倡像“瀑布模型”那样，一开始就制定整个产品的详细开发计划。而是提倡“小步快跑”的方式来实行整个产品功能。通过一个小计划接着一个小计划，通过反复迭代，最终实现产品的自我“进化”，自我完善。  
特点：短节奏，快调整  
好处：灵活多变，反应敏捷。任何时候，只要市场有变化，就马上调整下一步的开发计划。甚至是彻底放弃，及时止损。

**【敏捷开发中的术语-知乎形象（吐槽）版】**

* 敏捷开发：我们也不知道到底要开发啥，走一步看一步吧
* 用户故事：老板说明天要上这个功能，怎么实现我不管
* 快速迭代：上次做的功能点击率太低，把广告改成全屏
* 用户痛点：昨天用户投诉了，把广告调整下
* 拥抱变化：老板天天都有新想法，大家要适应（不要怪我）
* 持续交付：每个版本都有问题，总是持续交给测试，交付间隔10分钟
* 结对开发：bug太多了，直接去测试妹子的工位边测边改
* 代码评审：这个代码是你审的，将来出了问题你是要负责任的
* 弹性工作：不限定下班时间，修完bug才能走
* 四个润会：（Scrum Meeting）每天早上9:00开始，想上班迟到没门

**DevCloud 华为软开云**

https://www.huaweicloud.com/devcloud/  
一站式云端DevOps平台（是面向软件开发者提供的云服务，包括：

* 项目管理（敏捷迭代开发、社交化写作、多维度看板、项目统计报表）
* 配置管理
* 代码检查
* 编译、构建
* 测试管理
* 部署
* 发布管理