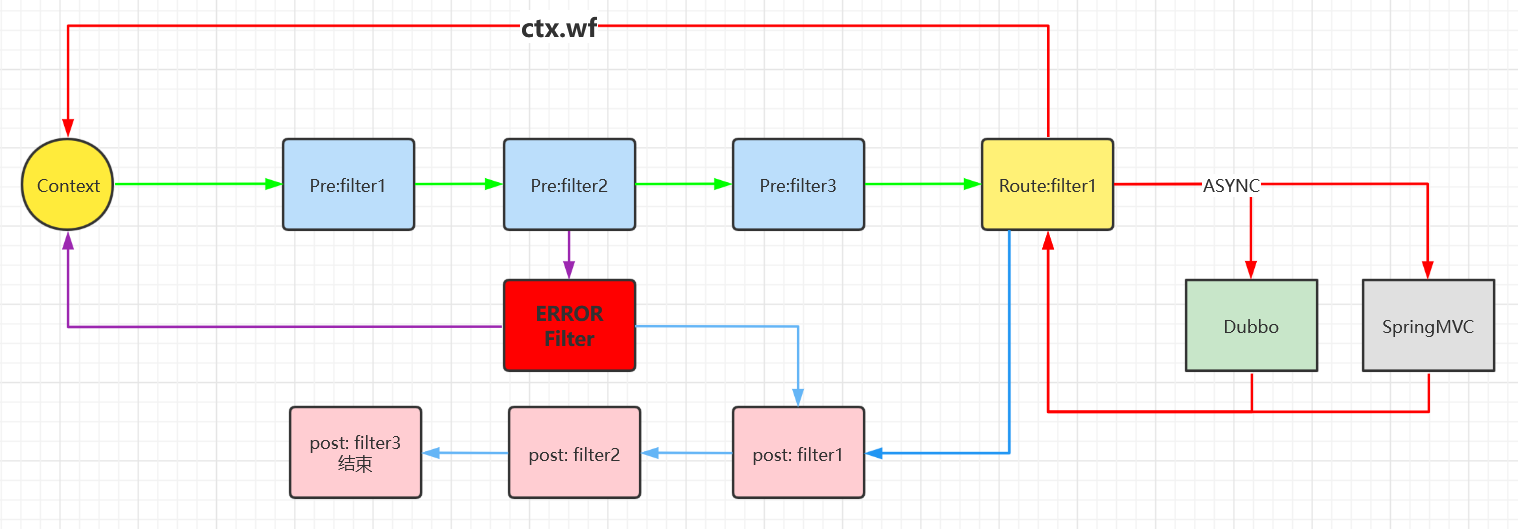
### 网关过滤定义与设计

* 过滤器设计：
  + 过滤器SPI加载的；
  + 过滤器是有类型的：
    - Pre：前置过滤器
    - Route：中置过滤器（路由）, 一定只有一个
    - Post：后置过滤器
    - Error：异常过滤器
  + 过滤器是串行化执行的，类似一个链表；
  + 
    - 1 过滤器一定是先执行前置过滤器，然后中置过滤器，开启一个异步的线程去对下游服务发起请求；
    - 2 下游服务响应之后，直接调用ctx.writeAndFlush方法写回数据；
    - 3 如果中间出现任何异常，都进行执行Error过滤器，直接调用ctx.writeAndFlush方法写回数据；
    - 4 最后执行后置过滤器；
  + 过滤器核心领域模型：
  + ProcessorFilter接口：
    - entry方法：真正的子类过滤器执行业务逻辑的方法
      * TimeoutPreFilter：
      * try finally：最终要触发下一个节点的流转：super.fireNext(ctx, args);
    - check方法：判断过滤器是否执行，通过Context对象取到规则对象再取到指定的过滤器配置
    - transformEntry方法：数据传输的方法，一些准备工作
      * AbstractEntryProcessorFilter：filterConfigClass的一些缓存加载和解析的工作；
    - fireNext方法：就是让节点持续向下触发；
  + ProcessorFilter层次结构：
    - AbstractLinkedProcessorFilter：链表的基础元素抽象类
      * ProcessorChainFilter：链表抽象类，添加了两个抽象接口定义：addFirst addLast
        + DefaultProcessorChainFilter：

链表实现类，虚拟的头结点；

执行链表的entry方法进行调用；

* + - * AbstractEntryProcessorFilter：真正的实际过滤器要继承的父类
        + TimeoutPreFilter
        + LoadbalancePreFilter...
  + ProcessorFilterFactory
    - buildfilterchain：创建过滤器链条方法
    - doFilterChain：正常的过滤器链条执行：pre + route + post
    - doErrorFilterChain：异常的过滤器链条执行：error + post
  + ProcessorFilterFactory层次结构：
    - AbstractProcessorFilterFactory抽象类：主要实现了buildfilterchain方法用于创建过滤器链条
      * DefaultProcessorFilterFactory子类实现：SPI方式加载所有的过滤器，调用接口定义的执行过滤器方法