 图5视图帮助序列图 在类图表中，大家可以发现，可能存在没有任何相关帮助类的视图，这种情况下，通常代表视图的JSP页面会有一些静态的或小数量的脚本代码。 这里我们对于序列图中的各个元素加以简单的介绍： (1)视图(view)。视图负责向用户展示动态数据信息，而帮助类则负责支持视图的工作，即打包和建立相应的数据模型。 (2)帮助类(helper)。一个帮助类负责帮助视图或控制器完成相关的处理工作，包括收集数据、存储中间模型等；帮助类也可以在保证数据完整性和准确性的情况下，为不同显示需求修改数据模型，也就是说，根据用户的请求，帮助类可以向视图提供未经处理的原始数据，或是已经格式化后的Web内容；一个视图同时可以和多个帮助类协同工作，而后者通常是由JavaBeans和标记(tag)实现的。 (3)值bean(ValueBean)。值bean实际上是用于存储中间数据模型的帮助类的另一种叫法，例如在序列图5中，business service就根据请求返回了一个值bean。 (4)业务服务(business service)。业务服务是指用户试图得到的，应用系统可以提供的相关服务；通常来说，业务服务可以通过一个业务代表(business delegate)来访问，而后者主要是提供对于业务服务的控制和保护。 在应用系统的视图模块中使用帮助类可以将不同的程序逻辑很好地分离开来，并在视图模块之外为开发人员提供设计程序逻辑的空间；基于JavaBean和标记(tag)所开发的帮助类通常都可以被多个视图模块重用，因此也提高了组件的重用

性和可维护性；把显示逻辑从数据处理逻辑分离出来，也有利于开发团队中角色及人物的划分；比如说，如果各种程序逻辑过于结合的话，软件开发人员可能需要在HTML，网页中修改代码而Web设计师则需要在处理数据访问的JSP中修改页面布置，这些情况都可能会导致系统设计和开发中由于不同技术人员的介入，而产生相关的问题。 5、会话面 会话面(session facade)模式在合作的企业对象间调节操作，并将应用函数合成一个单一简单的界面；它减少了类之间合作的复杂性，并使得类的调用者在该类变化的时候无需改动，这种模式通常以一个会话bean实现，以用来隐藏底层ejb的复杂交互。 这种设计模式出现的背景在于EJB通常既包括程序数据，又包括程序逻辑，而这些代码都会通过一定的界面作用于客户层，在多层次的J2EE平台应用程序中，就会造成一定的困难。 具体来说，在J2EE平台上的多层次系统中，通常会存在以下的问题： (1)层次之间联系过于紧密，客户层和后端的业务对象具有较强的依赖关系； (2)在客户和服务器之间有多次方法调用，因而导致了Web性能方面的问题； (3)缺乏一定的客户访问机制，使得一些后台对象被随便访问。 一个多层次的J2EE应用程序通常具有很多由EJB实现的服务器端对象，它们通常负责提供系统服务、数据信息等，也就是说作为业务对象，它们既包括相关的程序数据，也包括其程序逻辑；在J2EE应用系统中，负责程序逻辑的对象通常由会话bean实现，而表示持久性存储，并在多个用户间共享的对象则由实体bean来实现；当然，应用系统的用户需要访问企业对象来满足自己的需求，如果企业对象向用户提供接口，用户可以直接地与相关对象通信，但是这样一来，用户必须负责管理

所调用的企业对象之间的关系，并且能够处理其间的业务流程；然而，如果用户和业务对象之间存在过于直接的交互，两者的联系就会过于紧密，同时也使得用户过于依赖企业对象的具体实现，并负责管理与交互过程有关的业务对象查找和创建，以及不同的对象间相互调用的关系，甚至一些时候用户还需要管理多次调用之间的事务管理环节。 在用户需求不断增加时，这也是应用系统经常发生的情况，用户与不同的企业对象之间的交互也会变得越来越复杂，而企业对象可能需要一定内部的更新才能满足前者的需要，但是这样的话用户又需要根据企业对象实现的变化而做出相应的改变，这种情况将为应用系统带来相当大的麻烦；在访问EJB应用系统时，用户需要与远程对象进行交互。如果用户直接与所有相关的业务对象交互的话，将带来很大的Web负担；因为对于每一个ejb的激活，都将产生一次远程的调用，而如果存在大量的系统用户，用户与对象间的交互就将为Web通信带来很大的压力，使系统性能受到很大破坏；如果用户可以直接访问后端的企业对象，但是系统中又缺少一个统一的用户访问机制，那么这些访问很有可能变得杂乱无章，引起系统性能的下降，甚至导致一些安全问题。 为了解决以上的问题，开发人员可以采用会话面的设计模式，即使用会话bean来实现一个面(facade)来包含一个工作流中所有相关对象的交互；这个会话面负责管理业务对象，并向用户提供一个统一的服务访问层，会话面可以面向底层对象的交互过程，并提供一个仅仅包含必须提供的接口的服务层，由此它将复杂的对象交互和用户之间隔离开来； 会话面也负责管理企业数据和企业对象之间的交互，并表达其中需要的企业逻辑，因此会话面

也可以管理企业对象之间的作用关系；同时，根据工作流的需要，会话面也管理对象的创建、查找、修改和删除。 在一个复杂的应用系统中，会话面可以将其生命周期的管理下放到一个单独的帮助对象去，比如说，会话面可以将管理会话和实体bean生命周期的工作交给服务定位对象；同时，在应用系统中，检查业务对象之间的作用关系也是非常重要的，一些关系可能是暂时的，即只使用于一定的交互过程，而另外一些关系则是永久的，暂时的关系适合建模于会话面中的工作流，永久的关系则需要具体情况具体分析。 图6中的类图简要描述了会话面的设计模式，图7给出了会话面的序列表示，即参与组件及其交互关系。 图7会话面序列图 