[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
G06F 17/30 (2006.01)

[12] 发明专利说明书

专利号 ZL 200610086573.6

[45] 授权公告日 2009 年 9 月 23 日

[11] 授权公告号 CN 100543742C

[22] 申请日 2006.6.30

[21] 申请号 200610086573.6

[73] 专利权人 华为技术有限公司

地址 518129 广东省深圳市龙岗区坂田华为总部办公楼

[72] 发明人 曹祖鹏

[56] 参考文献

US2005/0050039A1 2005.3.3

CN1508703A 2004.6.30

US5664217A 1997.9.2

审查员 于 白

[74] 专利代理机构 北京集佳知识产权代理有限公司

代理人 逯长明

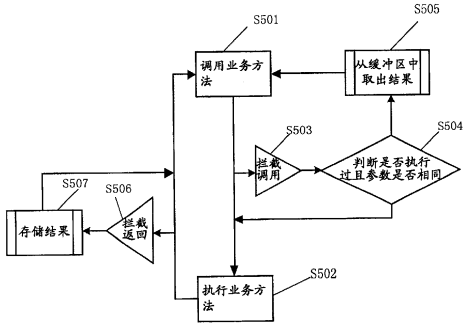
权利要求书 1 页 说明书 9 页 附图 6 页

[54] 发明名称

企业应用构架中利用缓冲技术调用数据的方法及装置

[57] 摘要

本发明公开了一种企业应用构架中利用缓冲技术调用数据的方法，包括：预先设置的拦截器拦截调用者对业务方法的调用；所述拦截器判断所述业务方法是否执行过，如果是，所述拦截器从缓冲区中取出需要调用的数据返回给所述调用者，否则，返回调用者调用所述业务方法的流程中，其中，所述业务方法将执行后的结果直接返回给所述调用者，在所述业务方法将执行后的结果直接返回给所述调用者时，所述拦截器将所述业务方法执行后的结果存储到所述缓冲区中。 本发明不需过多的修改原代码和改变原有的业务流程，不但减少了开发人员和维护人员的工作量，还可以在开发期不考虑缓冲的情况下，根据系统实际运行后的结果来灵活调整缓冲的执行，特别适合系统调优。



1.一种企业应用构架中利用缓冲技术调用数据的方法，其特征在于包括：

预先设置的拦截器拦截调用者对业务方法的调用；

所述拦截器判断所述业务方法是否执行过，如果是，所述拦截器从缓冲区中取出需要调用的数据返回给所述调用者，否则，返回调用者调用所述业务方法的流程中，其中，所述业务方法将执行后的结果直接返回给所述调用者，在所述业务方法将执行后的结果直接返回给所述调用者时，所述拦截器将所述业务方法执行后的结果存储到所述缓冲区中。

2.如权利要求 1 所述的企业应用构架中利用缓冲技术调用数据的方法，其特征在于所述拦截器拦截调用者对业务方法的调用的过程包括：拦截器拦截调用者发送给业务方法的参数。

3.如权利要求 1 所述的企业应用构架中利用缓冲技术调用数据的方法，其特征在于所述拦截器拦截调用者对业务方法的调用的过程包括：拦截器拦截调用者发送给业务方法的参数，同时，还进一步判断业务方法的名称是否已在配置文件中配置过，如果是，再判断所述调用者发送给业务方法的参数是否被所述业务方法处理过，否则，拦截器将所述调用者发送给业务方法的参数传送给所述业务方法进行处理。

4.如权利要求 2 或 3 所述的企业应用构架中利用缓冲技术调用数据的方法，其特征在于判断所述业务方法是否执行过的过程包括：判断所述调用者发送给业务方法的参数是否被所述业务方法处理过。

5.如权利要求 1 所述的企业应用构架中利用缓冲技术调用数据的方法，其特征在于所述拦截器拦截调用者对业务方法的调用的过程包括：所述拦截器判断所述调用者调用的业务方法的名称是否已配置在所述拦截器的配置文件中，如果是，拦截所述调用者发出的业务方法的参数。

6.如权利要求 1 所述的企业应用构架中利用缓冲技术调用数据的方法，其特征在于：所述业务方法执行后的结果为所述业务方法对所述业务方法的参数处理后的结果。

企业应用构架中利用缓冲技术调用数据的方法及装置

技术领域

本发明涉及企业应用构架技术，尤其涉及企业应用构架中调用数据技术。

背景技术

目前，存在多种构架形式的企业应用构架，例如J2EE中的企业应用构架及.net中的企业应用构架，每种构架都会存在调用数据技术。以J2EE中的企业应用构架为例，这种构架基本都采用了MVC（Model-View-Controller，模型-视图-控制）的N层构架模式，其典型模式如图1所示。其中，Servlet是用Java编写的Server（服务器）端程序，它与协议和平台无关，Servlet运行于Java－enabled Web Server中，Servlet可以动态地扩展Server的能力，并采用请求－响应模式提供Web服务； JSP是Java Server Page的缩写，是Sun公司出品的Web开发语言；DAO是Data Access Object（数据接入对象）的缩写，Business Object(业务对象)代表数据客户端，正是该对象需要访问数据源以获取和存储数据，DAO是该模式的主要对象，DAO抽取该Business Object的低层数据访问实现，以保证对数据源的透明访问，Business Object也可以把数据加载和存储操作委托给DAO。

现在结合图2，对基于图1的调用数据的方法进行说明。在步骤S201中，用户操作Web Browser(页面浏览器)，查看Web Layer(页面层)中的JSP Page（JSP页面）；进入步骤S202，Web Layer调用Bussiness Layer（业务层）中的JavaBean来进行业务逻辑操作；最后进入步骤S203，JavaBean读写数据库并将结果返回给Web Layer。其中，JavaBean是一种基于Java的软件组件，用户可以使用JavaBean将功能、处理、值、数据库访问和其他任何可以用Java代码创造的对象进行打包，并且其他的开发者可以通过内部的JSP Page、Servlet、其他JavaBean、applet程序或者应用来使用这些对象，用户可以认为JavaBean提供了一种随时随地的复制和粘贴的功能，而不用关心任何改变，DAO对象就是JavaBean的一种。

在一个大型的系统中，频繁的操作数据库是不可避免的，如果多次的从数据库中取相同的数据，必然会占用大量的I/O(输入/输出)接口和网络带宽，导致不必要的资源浪费，而如果在Bussiness Layer中设置一层缓冲，并将那

些常用的数据存储到缓冲中，当 JavaBean 需要读这些常用的数据时，就可避免频繁的操作数据库，减少不必要的资源开销。

为此，本领域普通技术人员又提出了一种利用缓冲技术调用数据的方法进行说明。如图 3 所示，在步骤 S301 中，用户操作 Web Browser，查看 Web Layer（页面层）中的 JSP Page；进入步骤 S302，Web Layer 调用 Bussiness Layer 中的业务方法进行业务逻辑操作；进入步骤 S303，判断上述业务方法是否执行过且参数是否相同，如果是，进入步骤 S304，否则，进入步骤 S305；在步骤 S304 中，从设置在 Bussiness Layer 中的缓冲区中提取已存储且需要调用的数据返回给 Web Layer；在步骤 S305 中，执行业务方法，将得到的结果存储到 Bussiness Layer 中的缓冲区中并返回给 Web Layer。

在上述方法中，当确定业务方法已执行过且参数相同，则可直接从 Bussiness Layer 中的缓冲区中提取需要调用的数据返回给 Web Layer，而没有必要再次执行业务方法进而对数据库进行操作，从而避免了频繁的操作数据库，但是，这种方法还存在如下缺点：

1.每个需要缓冲的方法中都需要引入这样复杂的缓冲代码，不但会导致大量的代码重复，而且处理业务逻辑的代码与缓冲的代码混合在一起，会给代码维护工作带来很多困难；

2.设计人员必须在开发的时候就改变原有的数据调用流程，引入和业务逻辑无关的 cache（高速缓冲存储器）对象；

3.如果需要通过这样的缓冲方法来优化已经运行的数据调用的流程，则必须获得相应的源代码，加入缓冲的实现，并修改所有对此方法的引用才行，这些条件在实际应用中往往是不具备的。

4.如果 Cache 的性能不能满足要求，没有代码也无法作出修改，如果需要在方法中去掉缓冲，则必须修改源代码以去掉缓冲的部分，并进行重新编译发布，这个过程违反了 OCP（Open-Close Principle，开闭原则）。

从上述缺点来看，图 3 所示的利用缓冲技术调用数据方法的最大缺点是每个需要缓冲的业务方法中都引入缓冲代码，导致业务逻辑代码与缓冲代码混合在一起，不利于代码的维护和修改，实质上，这种方法实现了一种侵入式的缓冲，已违反了 OCP。另外，这个缺点不是 J2EE 企业构架所特有的，其他构架中也都存在这种问题。

发明内容

本发明要解决的技术问题在于提供一种利用缓冲技术调用数据的方法，以实现在符合 OCP 的前提下避免频繁操作数据库的目的。

本发明提供了一种企业应用构架中利用缓冲技术调用数据的方法，包括：A.预先设置的拦截器拦截调用者对业务方法的调用；B.所述拦截器判断所述业务方法是否执行过，如果是，转步骤 C，否则，转步骤 D；C.所述拦截器从缓冲区中取出需要调用的数据返回给所述调用者；D. 返回调用者调用所述业务方法的流程中，其中，所述业务方法将执行后的结果直接返回给所述调用者，在所述业务方法将执行后的结果直接返回给所述调用者时，所述拦截器将所述业务方法执行后的结果存储到所述缓冲区中。

所述拦截器拦截调用者对业务方法的调用的过程包括：拦截器拦截调用者发送给业务方法的参数。

所述拦截器拦截调用者对业务方法的调用的过程包括：拦截器拦截调用者发送给业务方法的参数，同时，还进一步判断业务方法的名称是否已在配置文件中配置过，如果是，再判断所述调用者发送给业务方法的参数是否被所述业务方法处理过，否则，拦截器将所述调用者发送给业务方法的参数传送给所述业务方法进行处理。

判断所述业务方法是否执行过的过程包括：判断所述调用者发送给业务方法的参数是否被所述业务方法处理过。

所述拦截器拦截调用者对业务方法的调用的过程包括：所述拦截器判断所述调用者调用的业务方法的名称是否已配置在所述拦截器的配置文件中，如果是，拦截所述调用者发出的业务方法的参数。

在本发明中，调用业务方法的调用者首先触发拦截器，或者拦截器首先拦截调用者对业务方法的调用，之后拦截器判断业务方法是否执行过，如果是，从缓冲区中查询需要的结果，否则，调用业务方法，将业务方法执行后的结果返回给调用者，本领域普通技术人员都应该知道，拦截技术一般都是

挂接在正常流程之上的，即不需过多的修改原代码和改变原有的业务流程，不但减少了开发人员和维护人员的工作量，还可以在开发期不考虑缓冲的情况下，根据系统实际运行后的结果来灵活调整缓冲的执行，特别适合系统调优。

本发明通过灵活的配置文件，允许将缓冲加入到任何一个业务方法中，而不需修改业务方法的代码。

本发明的缓冲算法是由拦截技术挂接在正常流程之上，该技术可以随时修改、更换最优的缓冲实现，实现最优化的缓冲，而不影响任何一个客户端的代码。

本发明通过拦截器的正则表达式的匹配模式，可以在运行时改变拦截的方法，而不做任何编码。

在本发明中，拦截器不但可以拦截调用者对业务方法的调用，还可以对执行业务方法后返回的结果进行拦截，并将结果存储到缓冲区中，进一步说，只要经过拦截器第一次拦截的业务方法，其结果都会存储到缓冲区中，如果拦截器再次对同样的业务方法进行拦截，拦截器就会在缓冲区中找到需要调用的结果返回给调用者，而不必再对数据库进行操作，一个应用软件中会有多个业务方法，采用本发明上述的技术手段可以进一步避免对数据库频繁的操作。

附图说明

图 1 为 J2EE 采用的 MVC 的 N 层构架模式示意图；

图 2 为基于图 1 的调用数据方法的流程图；

图 3 为现有的一种利用缓冲技术调用数据方法的流程图；

图 4 为本发明方法的实施例的流程图；

图 5 为本发明拦截器的工作示意图；

图 6 为本发明装置的结构示意图。

具体实施方式

下面我们将结合附图，对本发明的最佳实施方案进行详细描述。首先要指出的是，本发明中用到的术语、字词及权利要求的含义不能仅仅限于其字面和普通的含义去理解，还包括进而与本发明的技术相符的含义和概念，这是因为我们作为发明者，要适当地给出术语的定义，以便对我们的发明进行最恰当的描述。因此，本说明和附图中给出的配置，只是本发明的首选实施方案，而不是要列举本发明的所有技术特性。我们要认识到，还有各种各样的可以取代我们方案的同等方案或修改方案。

本发明在企业应用构架的业务层中设置了拦截器，调用者在调用业务方法的过程中会触发拦截器，或者拦截器在调用者对业务方法的调用过程中对调用进行拦截，，之后，拦截器首先会在缓冲区中查询业务方法执行后的结果，如果缓冲区中存在所述结果，则拦截器将所述结果返回给调用者，否则，拦截器调用业务方法，之后，将执行的结果返回给调用者。

现在结合图 4，对本发明方法的实施例进行说明。

在步骤 S401 中，调用业务方法的调用者触发拦截器。

本实施例所述的调用者可以为企业应用构架中的 Web Layer ，业务方法可以为 JavaBean 中的一个函数。

拦截器可以由动态代理类实现。Jdk1.3 已开始支持动态代理类，同样，在.net 系统中，利用反射机制也可以实现相似的动态代理类。动态代理类是一种特别的类，它能在运行期间决定实现哪个接口，动态代理是一种强大的语言结构，它使我们可以为一个或多个接口创建实现对象，而不需要预先有一个实现类。本发明编写了一个通用的 Proxy 类，来代理需要缓冲的业务方法，Proxy 类主要包含以下内容:

Protected Proxy(InvocationHandler h): 构造函数，用于给内部的调用句柄 h 赋值;

Static Class getProxyClass (ClassLoader loader, Class[] interfaces): 获得一个代理类，其中 loader 是类装载器，interfaces 是真实类所拥有的全部接口的数组;

Static Object newProxyInstance(ClassLoader loader, Class[] interfaces, InvocationHandler h): 返回代理类的一个实例，返回后的代理类可以当作被代

理类使用(可使用被代理类的在 Subject 接口中声明过的方法)。

在实际应用中，调用者其实都是调用 Proxy 类，这样 Proxy 类就具备了拦截器的能力。本发明的缓冲操作在 Proxy 类中进行，不影响实际的操作流程。我们可以在配置文件中设置需要进行拦截的业务方法的名称，调用者可以通过将业务方法的参数及业务方法的名称发送给拦截器的方式触发拦截器，业务方法可以对业务方法的参数进行处理，处理后的结果就是调用者需要的结果。所述业务方法的名称可以使用正则表达式编写，例如，

<value>.*find.*</value>

<value>.*get.*</value>

上述表达式指明了拦截器需要拦截的是名称包含 find、get 的业务方法，当调用者将包含 find、get 的业务方法的名称及参数发送给拦截器时，拦截器即被触发，需要说明的是，如果调用者将配置文件中未配置的业务方法的名称发送给拦截器时，拦截器可以提示调用者此业务方法的名称未在配置文件中配置，或者不作出任何响应。还需要说明的是，如果拦截器发现调用者发送的是配置文件中未配置的业务方法的名称，则可将所述名称在配置文件中配置，以便以后能够接受调用者调用所述业务方法时的触发后者拦截调用者对所述业务方法的调用。

进入步骤S402,拦截器判断业务方法是否执行过,如果是,进入步骤S403,否则，进入步骤 S404。在此步骤中，拦截器可以判断业务方法是否对调用者传送过来的参数进行过处理，如果是，确定业务方法已执行过，否则，即使业务方法对其他参数进行过处理，也确定业务方法未执行过。

在步骤 S403 中，所述拦截器从缓冲区中取出需要调用的数据返回给所述调用者。缓冲区可以是独立于拦截器的实体，也可以设置在拦截器中。在本实施例中，可以将业务方法所在对象的名称、业务方法的名称及业务方法的参数组成的关键词、以及业务方法执行后的结果分别作为哈希表的 Key（索引）及 Value（值）存储到所述缓冲区中，例如，如果业务方法所在对象的名称为 calculator，业务方法的名称为 add，调用者传送过来的参数为 1 和 2，则 Key 可以为 calculator-add（1，2）或 calculator-add-1，2 等形式，当然，Key 也可以为三者通过运算或处理得到的任何形式的字符串。在本步骤中，拦截器可以在缓冲区中查询业务方法所在对象的名称、业务方法的名称及业务方

法的参数组成的 Key 所对应的数据，如果在缓冲区中发现匹配的 Key，则将 Key 对应的 Value 返回给调用者。在实际应用中，由于哈希表中的数据被破坏、拦截器没有使用正确的 Key 查询数据或者拦截器错误的确定业务方法执行过等原因，都可能导致拦截器在缓冲区中没有找到对应的数据，这时，可以通过步骤 S404 解决这个问题。

在步骤 S404 中，拦截器调用所述业务方法，并将所述业务方法执行后的结果返回给所述调用者。拦截器可以通过将业务方法的参数发送给业务方法的方式调用业务方法，业务方法接收到参数后，对参数进行处理，拦截器最后将处理后的结果返回给调用者，例如，如果业务方法为 add（），参数为 1 和 2，则 add（）对 1 和 2 的处理结果为 3，拦截器就将 3 返回给调用者。此外，拦截器还可将业务方法所在对象的名称、业务方法的名称及业务方法的参数组成的关键词、以及执行后的结果分别作为哈希表的索引 Key 及值 Value 存储到缓冲区中，以便将来调用者需要得到处理相同参数的结果时，拦截器可直接从缓冲区中提取结果。

在上述实施例中，当调用者触发拦截器时，拦截器才开始工作，实际上，拦截器可以在没有被触发的情况下，主动的对调用者对业务方法的调用进行拦截，例如，当调用者调用业务方法时，拦截器可以主动的拦截调用者发送给业务方法的参数，之后判断参数是否被所述业务方法处理过，当然，拦截器在拦截参数的同时，还可以进一步判断业务方法的名称是否已在配置文件中配置过，如果是，再继续判断参数是否被所述业务方法处理过，否则，拦截器将参数传送给实际的业务方法进行处理。所述拦截器拦截调用者对业务方法的调用的过程包括：所述拦截器判断所述调用者调用的业务方法的名称是否已配置在所述拦截器的配置文件中，如果是，拦截所述调用者发出的业务方法的参数。

为使本领域普通技术人员更加容易的实现本发明，现在结合图 5，进一步说明本发明采取的技术手段。

如图 5 所示，步骤 S501 到步骤 S502 的流程是没有使用缓冲技术调用数据的流程，当然，步骤 S502 涉及到对数据库的操作，例如从数据库中读取数据等。本发明在原有流程的基础上增加了步骤 S503- S507。拦截器在本发明的工作原理为：执行步骤 S501 后，拦截器要对调用进行拦截（即步骤 S503），

判断业务方法是否执行过（即步骤 S504），如果是，从缓冲区中取出相应的结果（即步骤 S505），之后，返回步骤 S501；否则，返回原有的流程。如果通过原有流程执行业务方法（即步骤 S502）后，拦截器对业务方法执行后返回的结果进行拦截（即步骤 S506），将所述结果存储到缓冲区中（即步骤 S507），最后返回原有流程，即返回步骤 S501。从图 5 中可以看出，拦截器只是挂接到原有的流程中，无论是拦截调用还是拦截返回均没有破坏原有的流程，对于业务方法的调用者来说，其始终认为调用的是业务方法本身，从而实现了在不改变原有的流程下增加了缓冲技术。

另外，缓冲区中存储的结果有时可能需要删除或修改，此时就需要清除缓冲区中相应的数据。本发明采取的手段是，在拦截器中侦听含有 update 或 delete 名称的方法调用，如果有相应的方法调用则使缓冲失效。

上述实施例是实现本发明方法的优选实施方式。此外，本发明还提供了一种企业应用构架中利用缓冲技术调用数据的装置。如图 6 所示，企业应用构架中利用缓冲技术调用数据的装置 601 包括：拦截器 6011，用于接受调用者 602 的触发和/或拦截调用者 602 对业务方法的调用；缓冲区 6012，用于存储业务方法执行后的结果；如果拦截器 6011 确定所述业务方法已执行过，则从缓冲区 6012 中取出需要调用的数据返回给调用者 602，否则，调用所述业务方法，并将所述业务方法执行后的结果返回给所述调用者 602。

拦截器 6011 还包括：接受触发及拦截单元，用于接受调用者 602 的触发和/或拦截调用者 602 对业务方法的调用；判断单元，用于判断所述业务方法是否执行过；调用单元，用于调用所述业务方法或缓冲区 6012 中的数据；数据/结果返回单元，用于将从缓冲区 6012 中取出的数据或所述业务方法执行后的结果返回给调用者 602。

所述判断单元还用于判断调用者 602 调用的业务方法的名称是否已配置在拦截器 6011 的配置文件中，如果所述业务方法的名称已配置在所述配置文件中，则所述接受触发及拦截单元拦截调用者 602 发出的业务方法的参数。

如果拦截器 6011 在缓冲区 6012 中没有找到需要调用的数据，则拦截器 6011 调用所述业务方法，并将所述业务方法执行后的结果返回给调用者 602。

调用者 602 可以将业务方法的参数及拦截器 6011 的配置文件中已配置的业务方法名称发送给拦截器 6011，以实现对拦截器 6011 的触发。在配置文件

中，可以使用正则表达式编写业务方法的名称。

拦截器 6011 可以在缓冲区 6012 中查询业务方法的参数与业务方法的名称组成的关键词对应的数据，之后，拦截器 6012 将所述数据返回给调用者 602。

拦截器 6011 还可以将业务方法执行后的结果存储到缓冲区 6012 中，例如，拦截器 6011 将业务方法的参数与业务方法的名称组成的关键词、业务方法执行后的结果分别作为哈希表的 Key 及 Value 存储到缓冲区 6012 中。

以上所述仅是本发明的优选实施方式，应当指出，对于本技术领域的普通技术人员来说，在不脱离本发明原理的前提下，还可以作出若干改进和润饰，这些改进和润饰也应视为本发明的保护范围。

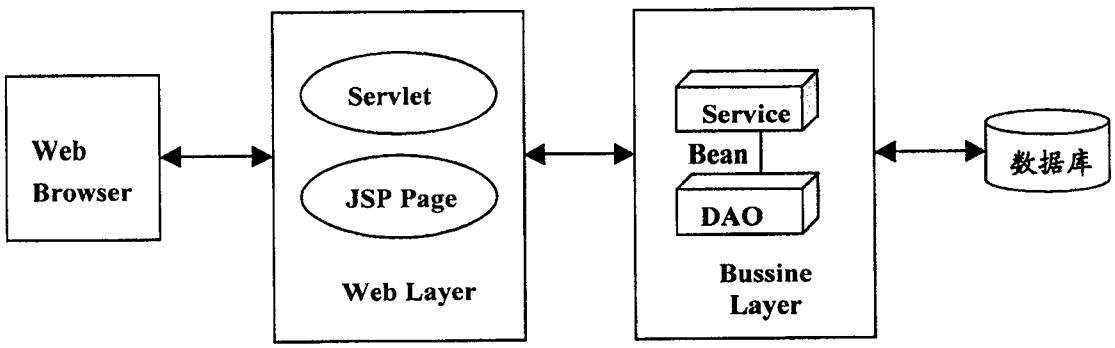

Web
Browser
Servlet
JSP Page
Web Layer
Service
Bean
DAO
Bussine
Layer
数据库

图1

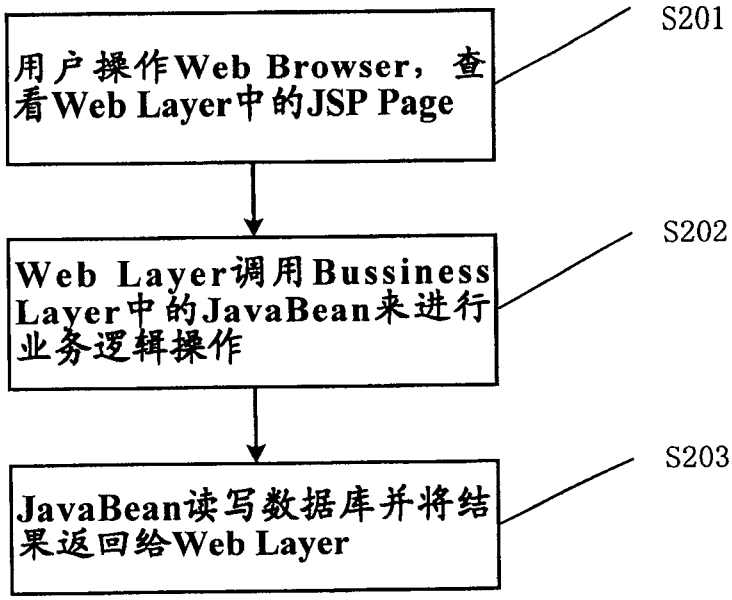

S201
用户操作Web Browser，查看Web Layer中的JSP Page
S202
Web Layer调用Bussiness Layer中的JavaBean来进行业务逻辑操作
S203
JavaBean读写数据库并将结果返回给Web Layer

图2

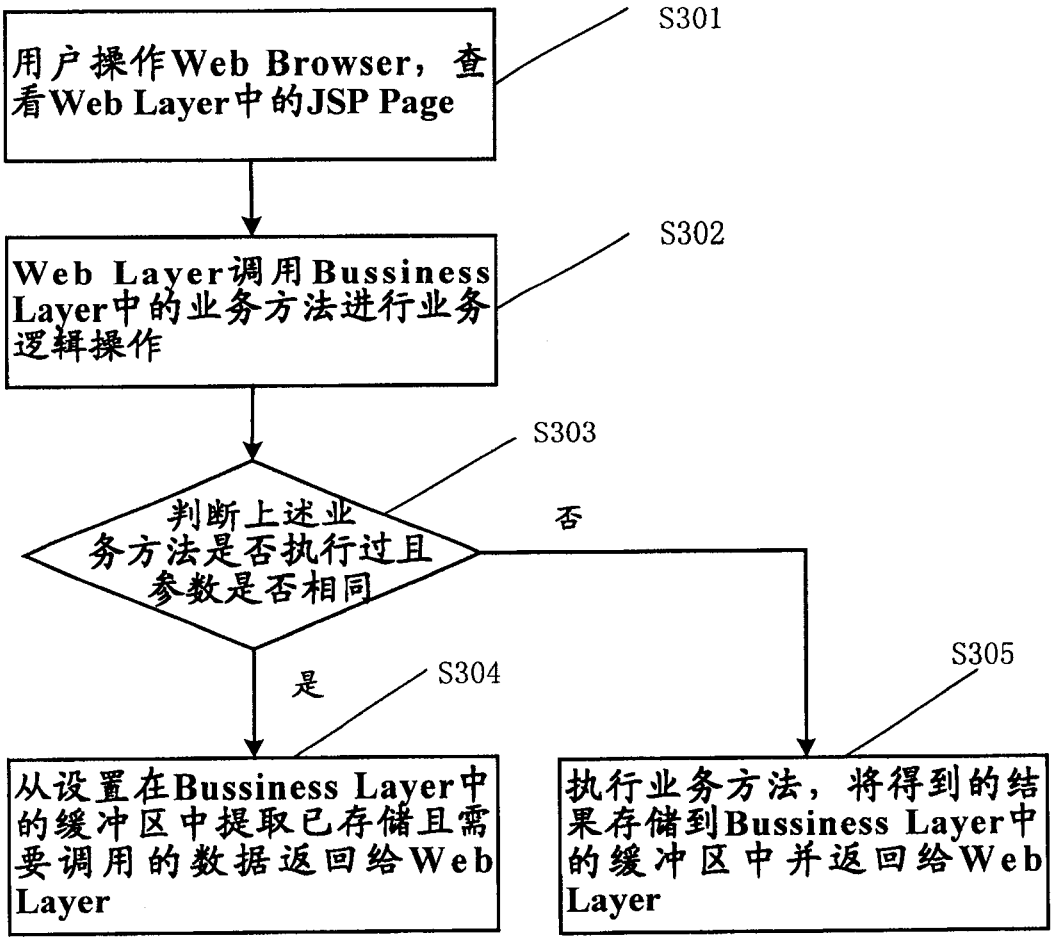

用户操作Web Browser，查看Web Layer中的JSP Page
S301
Web Layer调用Bussiness Layer中的业务方法进行业务逻辑操作
S302
判断上述业务方法是否执行过且参数是否相同
S303
否
是
S304
S305
从设置在Bussiness Layer中的缓冲区中提取已存储且需要调用的数据返回给Web Layer
执行业务方法，将得到的结果存储到Bussiness Layer中的缓冲区中并返回给Web Layer

图3

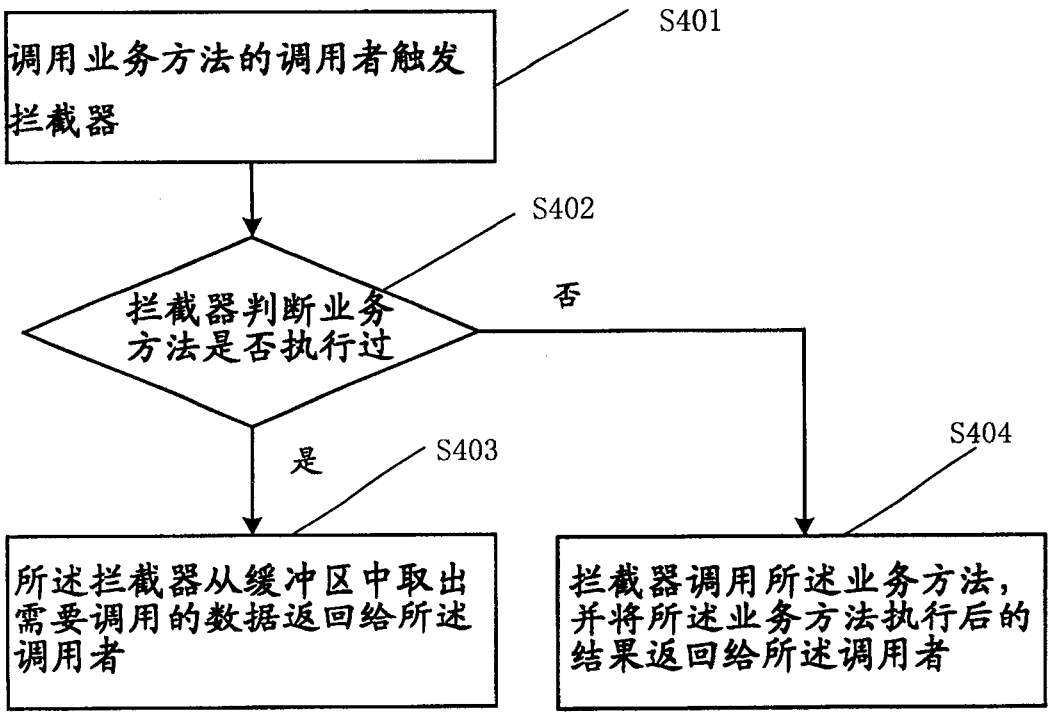

调用业务方法的调用者触发拦截器
S401
S402
拦截器判断业务方法是否执行过
否
是
S403
S404
所述拦截器从缓冲区中取出需要调用的数据返回给所述调用者
拦截器调用所述业务方法，并将所述业务方法执行后的结果返回给所述调用者

图4

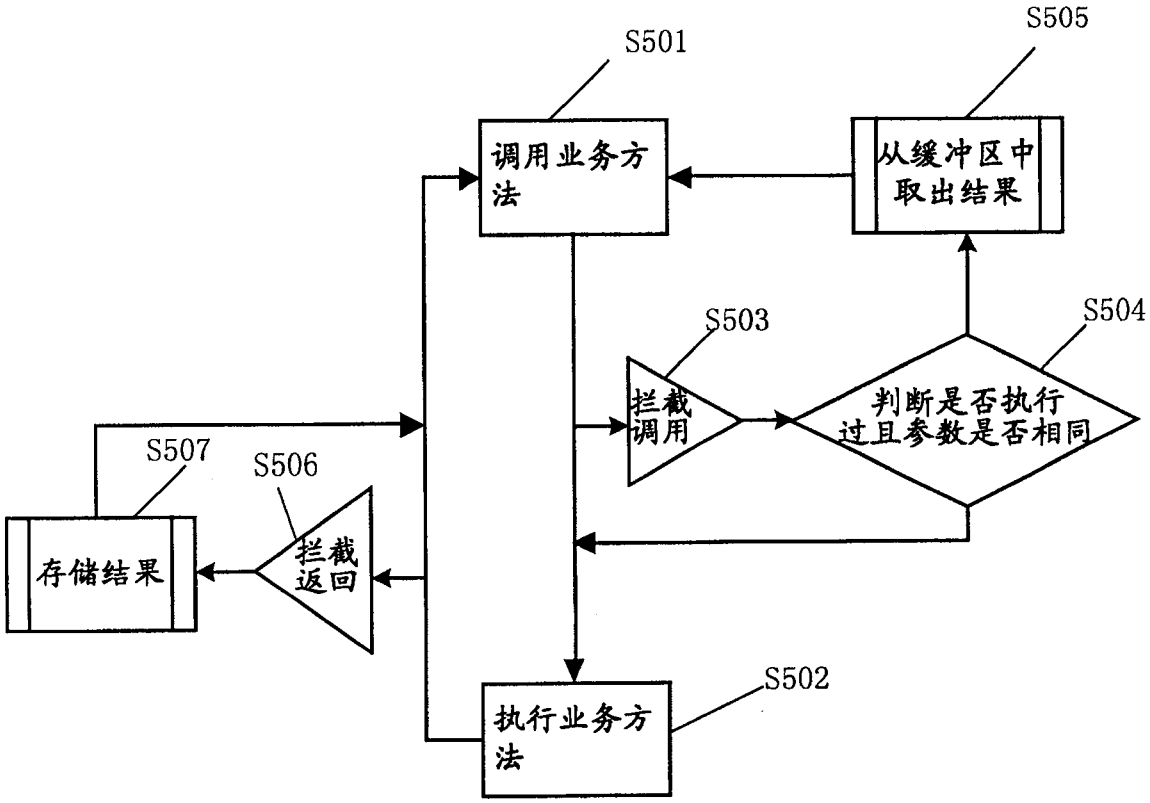

S501
调用业务方法
S505
从缓冲区中取出结果
S503
拦截调用
S504
判断是否执行过且参数是否相同
S507
存储结果
S506
拦截返回
S502
执行业务方法

图5

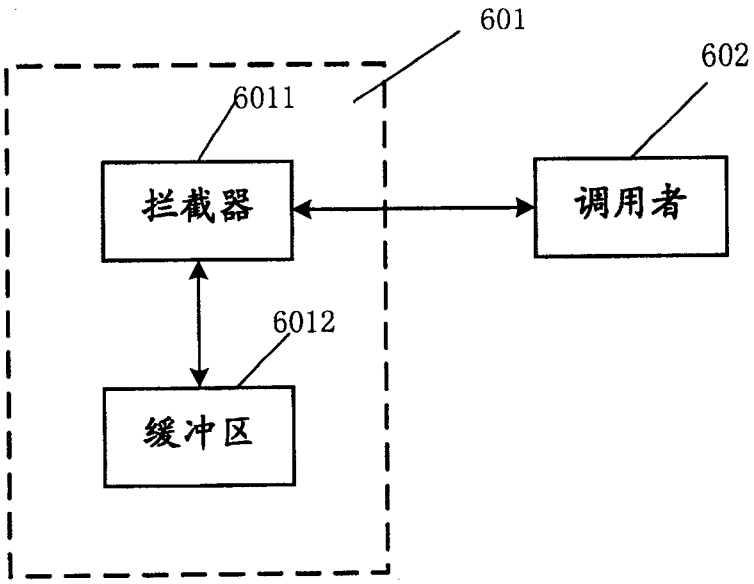


图6