缓存技术及在Rai owPortal的应用 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/253/2021_2022__E7_BC_93_E5_AD_98_E6_8A_80_E6_c97_253821.htm 1. ASP.NET缓存技术概述 将数据库中的数据缓存到内存(也可以存储在其他场所)，则无需在请求每个页面时都访问数据库。由于从内存中返回数据的速度始终比新提供的数据速度快，因而可以大大提高应用程序的性能。 ASP.NET为你使用缓存技术提供最大的灵活性，你可以缓存整个HTML页面，或是部分HTML页面，或是各种对象。你可以设置过期策略，或是设置依赖性，即在其他资源如文件或数据库表改变时，自动移出缓存。 ASP.NET中有两种基本的缓存： 输出缓存 页面输出缓存是最为简单的缓存机制，该机制将整个ASP.NET页面内容保存在服务器内存中。当用户请求该页面时，系统从内存中输出相关数据，直到缓存数据过期。在这个过程中，缓存内容直接发送给用户，而不必再次经过页面处理生命周期。通常情况下，页面输出缓存对于那些包含不需要经常修改内容的，但需要大量处理才能编译完成的页面特别有用。需要注意的是，页面输出缓存是将页面全部内容都保存在内存中，并用于完成客户端请求。 可以通过在Web.config进行配置，设置缓存策略，在一组ASP.NET页面中使用。还可以通过HttpCachePolicy类编程性设置页面缓存。 数据缓存 应用程序数据缓存提供了一种编程方式，可通过键/值对将任意数据存储在内存中。使用应用程序缓存与使用应用程序状态类似。但是，与应用程序状态不同的是，应用程序数据缓存中的数据是易失的，即数据并不是在整个应用程序生命周期中都

存储在内存中。应用程序数据缓存的优点是由ASP.NET管理缓存，它会在项过期、无效，或内存不足时移除缓存中的项，还可以配置应用程序缓存，以便在移除项时通知应用程序。 同时还有两种特殊的缓存，基于以上的缓存模型： 部分缓存 部分缓存本质上是输出缓存。顾名思义，页面部分缓存是将页面部分内容保存在内存中以便响应用户请求，而页面其他部分内容则为动态内容。页面部分缓存的实现包括两种方式：控件缓存和替换后缓存。前者也可称为片段缓存，这种方式允许将需要缓存的信息包含在一个用户控件内，然后，将该用户控件标记为可缓存的，以此来缓存页面输出的部分内容。这一方式缓存了页面中的特定内容，而没有缓存整个页面，因此，每次都需重新创建整个页。例如，如果要创建一个显示大量动态内容(如股票信息)的页，其中有些部分为静态内容(如每周总结)，这时可以将静态部分放在用户控件中，并允许缓存这些内容。缓存后替换与控件缓存正好相反。这种方式缓存整个页，但页中的各段都是动态的。例如，如果要创建一个在规定时间段内为静态的页，则可以将整个页设置为进行缓存。如果向页添加一个显示用户名的Label控件，则对于每次页刷新和每个用户而言，Label的内容都将保持不变，始终显示缓存该页之前请求该页的用户的姓名。使用缓存后替换机制，可以将页配置为进行缓存，将页的个别部分标记为不可缓存。在此情况下，可以向不可缓存部分添加Label控件，这样将为每个用户和每次页请求动态创建这些控件。 数据源缓存 数据源缓存是指在数据源控件如SqlDataSource, ObjectDataSource, and XMLDataSource中缓存数据，实际上是数据缓存，只不过缓存由控件内部实现。 缓

存依赖 缓存依赖允许缓存项依赖于另外一个资源，这样资源变化时，缓存项自动移出。 ASP.NET包括3种依赖类型 依赖于其他缓存项 依赖于文件或文件夹 依赖于数据库查询。 另外你还可以使用聚合依赖，或是自定义缓存依赖。 缓存功能也有其自身的不足。例如，显示的内容可能不是最新、最准确的，为此，必须设置合适的缓存策略。缓存增加了系统的复杂性并使其难于测试和调试，你设置的断点、监控变量等由于缓存在调试时都可能无效。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com