MySQL 数据备份与还原 - 逆心

一、数据备份

1、使用mysqldump命令备份

mysqldump命令将数据库中的数据备份成一个文本文件。表的结构和表中的数据将存储在生成的文本文件中。

mysqldump命令的工作原理很简单。它先查出需要备份的表的结构，再在文本文件中生成一个CREATE语句。然后，将表中的所有记录转换成一条INSERT语句。然后通过这些语句，就能够创建表并插入数据。

1、备份一个数据库

mysqldump基本语法：

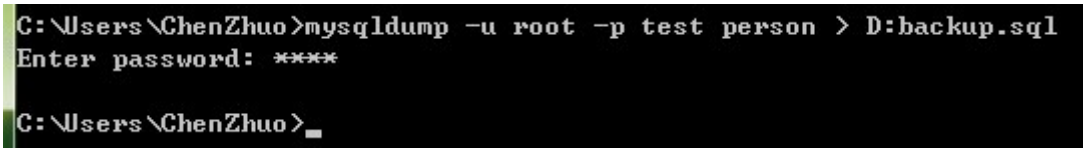
mysqldump -u username -p dbname table1 table2 ...-> BackupName.sql

其中：

- dbname参数表示数据库的名称；
- table1和table2参数表示需要备份的表的名称，为空则整个数据库备份；
- BackupName.sql参数表设计备份文件的名称，文件名前面可以加上一个绝对路径。通常将数据库被分成一个后缀名为sql的文件；

使用root用户备份test数据库下的person表

```
mysqldump -u root -p test person > D:\backup.sql
```



其生成的脚本如下：

```
DROP TABLE IF EXISTS `person`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `person` (
  `Id` int(11) NOT NULL auto_increment,
  `CountryId` int(11) default NULL,
  `Name` varchar(20) NOT NULL,
  `Sex` int(11) default '0',
  PRIMARY KEY  (`Id`),
  UNIQUE KEY `Name` (`Name`),
  KEY `FK_CID_PID` (`CountryId`),
  CONSTRAINT `FK_CID_PID` FOREIGN KEY (`CountryId`) REFERENCES `country` (`Id`)
) ENGINE=InnoDB AUTO_INCREMENT=11 DEFAULT CHARSET=gb2312;
SET character_set_client = @saved_cs_client;

--
-- Dumping data for table `person`
--

LOCK TABLES `person` WRITE;
/*!40000 ALTER TABLE `person` DISABLE KEYS */;
INSERT INTO `person` VALUES (1,1,'刘备',1),(2,1,'关羽',1),(3,1,'张飞',1),(4,2,'曹操
/*!40000 ALTER TABLE `person` ENABLE KEYS */;
UNLOCK TABLES;
```

文件的开头会记录MySQL的版本、备份的主机名和数据库名。

文件中以“--”开头的都是SQL语言的注释，以"/*!40101"等形式开头的是与MySQL有关的注释。40101是MySQL数据库的版本号，如果MySQL的版本比1.11高，则/*!40101和*/之间的内容就被当做SQL命令来执行，如果比4.1.1低就会被当做注释。

2、备份多个数据库

语法：

```
mysqldump -u username -p --databases dbname2 dbname2 > Backup.sql
```

加上了--databases选项，然后后面跟多个数据库

```
mysqldump -u root -p --databases test mysql > D:\backup.sql
```

3、备份所有数据库

mysqldump命令备份所有数据库的语法如下：

```
mysqldump -u username -p -all-databases > BackupName.sql
```

示例：

```
mysqldump -u -root -p -all-databases > D:\all.sql
```

2、直接复制整个数据库目录

MySQL有一种非常简单的备份方法，就是将MySQL中的数据库文件直接复制出来。这是最简单，速度最快的方法。

不过在此之前，要先将服务器停止，这样才可以保证在复制期间数据库的数据不会发生变化。如果在复制数据库的过程中还有数据写入，就会造成数据不一致。这种情况在开发环境可以，但是在生产环境中很难允许备份服务器。

注意：这种方法不适用于InnoDB存储引擎的表，而对于MyISAM存储引擎的表很方便。同时，还原时MySQL的版本最好相同。

3、使用mysqlhotcopy工具快速备份

一看名字就知道是热备份。因此,mysqlhotcopy支持不停止MySQL服务器备份。而且，mysqlhotcopy的备份方式比mysqldump快。mysqlhotcopy是一个perl脚本，主要在Linux系统下使用。其使用LOCK TABLES、FLUSH TABLES和cp来进行快速备份。

原理：先将需要备份的数据库加上一个读锁，然后用FLUSH TABLES将内存中的数据写回到硬盘上的数据库，最后，把需要备份的数据库文件复制到目标目录。

命令格式如下：

```
[root@localhost ~]# mysqlhotcopy [option] dbname1 dbname2
backupDir/
```

- dbname：数据库名称；
- backupDir：备份到哪个文件夹下；

常用选项：

- --help：查看mysqlhotcopy帮助；
- --allowold：如果备份目录下存在相同的备份文件，将旧的备份文件加上_old；
- --keepold：如果备份目录下存在相同的备份文件，不删除旧的备份文件，而是将旧的文件更名；
- --flushlog：本次辈分之后，将对数据库的更新记录到日志中；

- --noindices：只备份数据文件，不备份索引文件；
- --user=用户名：用来指定用户名，可以用-u代替；
- --password=密码：用来指定密码，可以用-p代替。使用-p时，密码与-p之间没有空格；
- --port=端口号：用来指定访问端口，可以用-P代替；
- --socket=socket文件：用来指定socket文件，可以用-S代替；

mysqlhotcopy并非mysql自带，需要安装Perl的数据库接口包；下载地址为:http://dev.mysql.com/downloads/dbi.html

目前，该工具也仅仅能够备份MyISAM类型的表。

二、数据还原

1、还原使用mysqldump命令备份的数据库的语法如下：

mysql -u root -p [dbname] < backup.sq
示例：

```
mysql -u root -p < C:\backup.sql
```

2、还原直接复制目录的备份

通过这种方式还原时，必须保证两个MySQL数据库的版本号是相同的。MyISAM类型的表有效，对于InnoDB类型的表不可用，InnoDB表的表空间不能直接复制。

一、简介
在Asp.net MVC实现的Comet推送的原理很简单。

服务器端：接收到服务器发送的AJAX请求，服务器端并不返回，而是将其Hold住，待到有东西要通知客户端时，才将这个请求返回。

客户端：请求异步Action，当接收到一个返回时，立即又再发送一个。

缺点：会长期占用一个Asp.net处理线程。但相比于轮询，其节省了带宽。

示例：

新建一个Controller如下：

```
    //Comet服务器推送控制器(需设置NoAsyncTimeout,防止长时间请求挂起超时错误)
    [NoAsyncTimeout, SessionState(SessionStateBehavior.ReadOnly)]
    public class CometController : AsyncController   //需要继承自异步的AsyncController
    {
        /// <summary>
        /// 异步方法，处理客户端发起的请求
        /// </summary>
        public void IndexAsync()
        {
            AsyncManager.OutstandingOperations.Increment();
            AsyncManager.Parameters["info"] = "怎么了";
            AsyncManager.OutstandingOperations.Decrement();
        }

        /// <summary>
        /// 当异步线程完成时向客户端发送响应
        /// </summary>
        /// <param name="token">数据封装对象</param>
        /// <returns></returns>
        public ActionResult IndexCompleted(string info)
        {
            return Json(info, JsonRequestBehavior.AllowGet);
        }
    }
```

随便找一个页面，通过AJAX请求这一个异步Action：

```
<html>
<head>
    <title>AJAX测试</title>
    <script src="/Content/jquery-1.10.2.min.js"></script>
    <script type="text/javascript">
        $(function () {
            getCometServerPush();
        })

        function getCometServerPush() {
            $.ajax({
                cache: false,
                url: '/Comet/Index',
                success: function (data) {
                    $("#info").html(data);
                    getCometServerPush();
                }
            });
        }

    </script>
</head>
<body>
    <div id="info"></div>
</body>
</html>
```

上面的示例，如果你在Action上下一个断点，会不停的看到断点在循环。说明异步客户端不停地在推送。当然这个示例仅仅是说明推送的原理。

二、应用

应用：监控服务器上的一个txt文件，当有变化时，推送内容到客户端。

```
    //Comet服务器推送控制器(需设置NoAsyncTimeout,防止长时间请求挂起超时错误)
    [NoAsyncTimeout, SessionState(SessionStateBehavior.ReadOnly)]
    public class CometController : AsyncController   //需要继承自异步的AsyncController
    {
        /// <summary>
        /// 异步方法，处理客户端发起的请求
        /// </summary>
        public void IndexAsync()
        {
            AsyncManager.OutstandingOperations.Increment();

            FileSystemWatcher FSW = new FileSystemWatcher();
            FSW.Filter = "123.txt";              //仅仅监控123.txt文件
            FSW.Path = Server.MapPath(@"/");   //设置监控路径
            FSW.EnableRaisingEvents = true;  //启动监控
            //FileSystemWatcher暂时有个多次触发的问题，但与本推送示例无关，故不解决
            FSW.Changed += (object source, FileSystemEventArgs e) =>
            {
                AsyncManager.Parameters["info"] = System.IO.File.ReadAllText(Server.MapPath(@"/123.txt"),System.Text.Encoding.Default); ;
                AsyncManager.OutstandingOperations.Decrement();
            };
        }

        /// <summary>
        /// 当异步线程完成时向客户端发送响应
        /// </summary>
```

```
        /// <param name="token">数据封装对象</param>
        /// <returns></returns>
        public ActionResult IndexCompleted(string info)
        {
            return Json(info, JsonRequestBehavior.AllowGet);
        }
    }
```

更多流逼的功能，留着读者自由发挥。