MongoDB 监控 MongoDB PHP

MongoDB Java

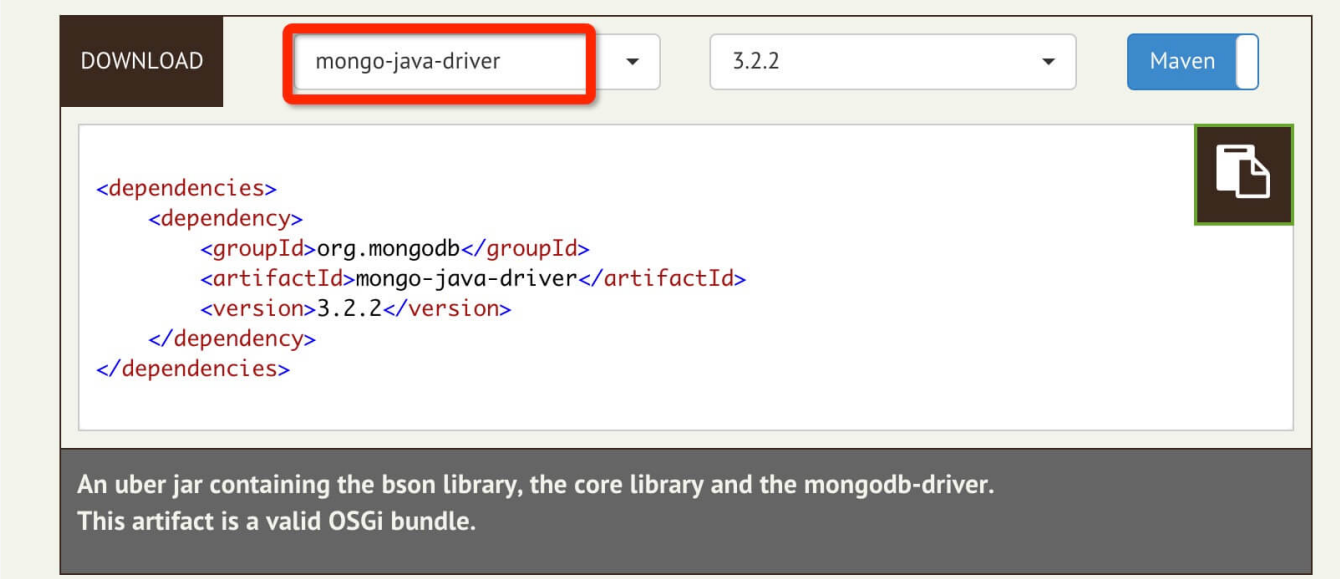
环境配置

在 Java 程序中如果要使用 MongoDB，你需要确保已经安装了 Java 环境及 MongoDB JDBC 驱动。

本章节实例时候 Mongo 3.x 以上版本。

你可以参考本站的Java教程来安装Java程序。现在让我们来检测你是否安装了 MongoDB JDBC 驱动。

- 首先你必须下载mongo jar包，下载地址：https://mongodb.github.io/mongo-java-driver/, 请确保下载最新版本。



- 你需要将 mongo-java-driver-3.2.2.jar （找到合适的版本）包含在你的 classpath 中。。
- 国内 mongodb-driver jar 下载地址：http://central.maven.org/maven2/org/mongodb/mongo-java-driver/

连接数据库

连接数据库，你需要指定数据库名称，如果指定的数据库不存在，mongo会自动创建数据库。

连接数据库的Java代码如下：

```
import com.mongodb.MongoClient;
import com.mongodb.client.MongoDatabase;

public class MongoDBJDBC{
   public static void main( String args[] ){
      try{
       // 连接到 mongodb 服务
         MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

         // 连接到数据库
         MongoDatabase mongoDatabase = mongoClient.getDatabase("mycol");
       System.out.println("Connect to database successfully");
```

```
       }catch(Exception e){
         System.err.println( e.getClass().getName() + ": " + e.getMessage() );
      }
    }
}
```

现在，让我们来编译运行程序并连接到数据库 mycol。

你可以根据你的实际环境改变 MongoDB JDBC 驱动的路径。

本实例将 MongoDB JDBC 启动包 mongo-java-driver-3.2.2.jar 放在本地目录下:

```
$ javac -cp .:mongo-java-driver-3.2.2.jar MongoDBJDBC.java
$ java -cp .:mongo-java-driver-3.2.2.jar MongoDBJDBC
Connect to database successfully
Authentication: true
```

本实例中 Mongo 数据库无需用户名密码验证。如果你的 Mongo 需要验证用户名及密码，可以使用以下代码：

```
import java.util.ArrayList;
import java.util.List;
import com.mongodb.MongoClient;
import com.mongodb.MongoCredential;
import com.mongodb.ServerAddress;
import com.mongodb.client.MongoDatabase;

public class MongoDBJDBC {
    public static void main(String[] args){
        try {
            //连接到MongoDB服务 如果是远程连接可以替换“localhost”为服务器所在IP地址
            //ServerAddress()两个参数分别为 服务器地址 和 端口
            ServerAddress serverAddress = new ServerAddress("localhost",27017);
            List<ServerAddress> addrs = new ArrayList<ServerAddress>();
            addrs.add(serverAddress);

            //MongoCredential.createScramSha1Credential()三个参数分别为 用户名 数据库名称 密码
            MongoCredential credential = MongoCredential.createScramSha1Credential("username", "database
Name", "password".toCharArray());
            List<MongoCredential> credentials = new ArrayList<MongoCredential>();
            credentials.add(credential);

            //通过连接认证获取MongoDB连接
            MongoClient mongoClient = new MongoClient(addrs,credentials);

            //连接到数据库
            MongoDatabase mongoDatabase = mongoClient.getDatabase("databaseName");
            System.out.println("Connect to database successfully");
```

```
         } catch (Exception e) {
             System.err.println( e.getClass().getName() + ": " + e.getMessage() );
         }
     }
}
```

创建集合

我们可以使用 com.mongodb.client.MongoDatabase 类中的createCollection()来创建集合

代码片段如下：

```
import com.mongodb.MongoClient;
import com.mongodb.client.MongoDatabase;

public class MongoDBJDBC{
   public static void main( String args[] ){
      try{
      // 连接到 mongodb 服务
      MongoClient mongoClient = new MongoClient( "localhost" , 27017 );



      // 连接到数据库
      MongoDatabase mongoDatabase = mongoClient.getDatabase("mycol");
      System.out.println("Connect to database successfully");
      mongoDatabase.createCollection("test");
      System.out.println("集合创建成功");

      }catch(Exception e){
        System.err.println( e.getClass().getName() + ": " + e.getMessage() );
     }
   }
}
```

编译运行以上程序，输出结果如下:

```
Connect to database successfully
集合创建成功
```

获取集合

我们可以使用com.mongodb.client.MongoDatabase类的 getCollection() 方法来获取一个集合

代码片段如下：

```
import org.bson.Document;
import com.mongodb.MongoClient;
```

```
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;

public class MongoDBJDBC{
   public static void main( String args[] ){
      try{
       // 连接到 mongodb 服务
         MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

         // 连接到数据库
         MongoDatabase mongoDatabase = mongoClient.getDatabase("mycol");
       System.out.println("Connect to database successfully");

       MongoCollection<Document> collection = mongoDatabase.getCollection("test");
       System.out.println("集合 test 选择成功");
      }catch(Exception e){
        System.err.println( e.getClass().getName() + ": " + e.getMessage() );
     }
   }
}
```

编译运行以上程序，输出结果如下:

```
Connect to database successfully
集合 test 选择成功
```

插入文档

我们可以使用com.mongodb.client.MongoCollection类的 insertMany() 方法来插入一个文档

代码片段如下：

```
import java.util.ArrayList;
import java.util.List;
import org.bson.Document;

import com.mongodb.MongoClient;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;

public class MongoDBJDBC{
   public static void main( String args[] ){
      try{
         // 连接到 mongodb 服务
         MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

         // 连接到数据库
```

```
         MongoDatabase mongoDatabase = mongoClient.getDatabase("mycol");
         System.out.println("Connect to database successfully");

         MongoCollection<Document> collection = mongoDatabase.getCollection("test");
         System.out.println("集合 test 选择成功");
         //插入文档
         /**
         * 1. 创建文档 org.bson.Document 参数为key-value的格式
         * 2. 创建文档集合List<Document>
         * 3. 将文档集合插入数据库集合中 mongoCollection.insertMany(List<Document>) 插入单个文档可以用 mongo
Collection.insertOne(Document)
         * */
         Document document = new Document("title", "MongoDB").
         append("description", "database").
         append("likes", 100).
         append("by", "Fly");
         List<Document> documents = new ArrayList<Document>();
         documents.add(document);
         collection.insertMany(documents);
         System.out.println("文档插入成功");
      }catch(Exception e){
         System.err.println( e.getClass().getName() + ": " + e.getMessage() );
      }
   }
}
```

编译运行以上程序，输出结果如下:

```
Connect to database successfully
集合 test 选择成功
文档插入成功
```

检索所有文档

我们可以使用 com.mongodb.client.MongoCollection 类中的 find() 方法来获取集合中的所有文档。

此方法返回一个游标，所以你需要遍历这个游标。

代码片段如下：

```
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;

public class MongoDBJDBC{
```

```
    public static void main( String args[] ){
       try{
          // 连接到 mongodb 服务
          MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

          // 连接到数据库
          MongoDatabase mongoDatabase = mongoClient.getDatabase("mycol");
          System.out.println("Connect to database successfully");

          MongoCollection<Document> collection = mongoDatabase.getCollection("test");
          System.out.println("集合 test 选择成功");

          //检索所有文档
          /**
          * 1. 获取迭代器FindIterable<Document>
          * 2. 获取游标MongoCursor<Document>
          * 3. 通过游标遍历检索出的文档集合
          * */
          FindIterable<Document> findIterable = collection.find();
          MongoCursor<Document> mongoCursor = findIterable.iterator();
          while(mongoCursor.hasNext()){
             System.out.println(mongoCursor.next());
          }

       }catch(Exception e){
          System.err.println( e.getClass().getName() + ": " + e.getMessage() );
       }
    }
}
```

编译运行以上程序，输出结果如下:

```
Connect to database successfully
集合 test 选择成功
Document{{_id=56e65fb1fd57a86304fe2692, title=MongoDB, description=database, likes=100, by=Fly}}
```

更新文档

你可以使用 com.mongodb.client.MongoCollection 类中的 updateMany() 方法来更新集合中的文档。

代码片段如下：

```
import org.bson.Document;
import com.mongodb.MongoClient;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
```

```
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;

public class MongoDBJDBC{
   public static void main( String args[] ){
      try{
         // 连接到 mongodb 服务
         MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

         // 连接到数据库
         MongoDatabase mongoDatabase = mongoClient.getDatabase("mycol");
         System.out.println("Connect to database successfully");

         MongoCollection<Document> collection = mongoDatabase.getCollection("test");
         System.out.println("集合 test 选择成功");

         //更新文档   将文档中likes=100的文档修改为likes=200
         collection.updateMany(Filters.eq("likes", 100), new Document("$set",new Document("likes",200
)));
         //检索查看结果
         FindIterable<Document> findIterable = collection.find();
         MongoCursor<Document> mongoCursor = findIterable.iterator();
         while(mongoCursor.hasNext()){
            System.out.println(mongoCursor.next());
         }

      }catch(Exception e){
         System.err.println( e.getClass().getName() + ": " + e.getMessage() );
      }
   }
}
```

编译运行以上程序，输出结果如下:

```
Connect to database successfully
集合 test 选择成功
Document{{_id=56e65fb1fd57a86304fe2692, title=MongoDB, description=database, likes=200, by=Fly}}
```

删除第一个文档

要删除集合中的第一个文档，首先你需要使用com.mongodb.DBCollection类中的 findOne()方法来获取第一个文档，然后使用remove 方法删除。

代码片段如下：

```
import org.bson.Document;
import com.mongodb.MongoClient;
```

```
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;

public class MongoDBJDBC{
   public static void main( String args[] ){
      try{
         // 连接到 mongodb 服务
         MongoClient mongoClient = new MongoClient( "localhost" , 27017 );

         // 连接到数据库
         MongoDatabase mongoDatabase = mongoClient.getDatabase("mycol");
         System.out.println("Connect to database successfully");

         MongoCollection<Document> collection = mongoDatabase.getCollection("test");
         System.out.println("集合 test 选择成功");

         //删除符合条件的第一个文档
         collection.deleteOne(Filters.eq("likes", 200));
         //删除所有符合条件的文档
         collection.deleteMany (Filters.eq("likes", 200));
         //检索查看结果
         FindIterable<Document> findIterable = collection.find();
         MongoCursor<Document> mongoCursor = findIterable.iterator();
         while(mongoCursor.hasNext()){
           System.out.println(mongoCursor.next());
         }

      }catch(Exception e){
        System.err.println( e.getClass().getName() + ": " + e.getMessage() );
     }
   }
}
```

编译运行以上程序，输出结果如下:

```
Connect to database successfully
集合 test 选择成功
```

更多操作可以参考：http://mongodb.github.io/mongo-java-driver/3.0/driver/getting-started/quick-tour/

参考文档：http://blog.csdn.net/ererfei/article/details/50857103

← MongoDB 监控 MongoDB PHP →

点我分享笔记