InterSystems IRIS デモ：XEP による Java オブジェクト永続性

Version 2023.1
2024-01-02

InterSystems Corporation　1 Memorial Drive　Cambridge MA 02142　www.intersystems.com

InterSystems IRIS デモ : XEP による Java オブジェクト永続性
InterSystems IRIS Data Platform　Version 2023.1　2024-01-02

InterSystems Worldwide Response Center (WRC)
Tel:　+1-617-621-0700
Tel:　+44 (0) 844 854 2917
Email:　support@InterSystems.com

目次

InterSystems IRIS デモ：XEP による Java オブジェクト永続性

ここでは、InterSystems IRIS® データ・プラットフォームでのきわめて迅速な Java オブジェクトの格納と取得のサポートを提供する XEP API (**com.intersystems.xep**) を紹介します。Java オブジェクト永続性を実現する XEP アプローチの概要を示し、API の主要機能の実例を示す単純なシナリオを紹介します。

これらのアクティビティは、既定の設定と機能のみを使用する設計になっているので、ユーザはこの概要の範囲を超える詳細を扱うことなく、XEP の基本部分を十分に理解することができます。XEP の完全なドキュメントは、"InterSystems XEP による Java オブジェクトの永続化" を参照してください。

1 迅速な Java オブジェクトの保存と検索

Java はオブジェクト指向言語なので、Java アプリケーションでデータをオブジェクトとしてモデル化するのは自然なことです。ただし、そのデータをデータベース内に保存する必要があるアプリケーションの場合、これによって問題が発生する可能性があります。JDBC を使用してオブジェクトの保存と検索を行う場合、オブジェクト・データを一連のリレーショナル・テーブルに変換し、クエリ結果セットをオブジェクトに戻すときに、問題に直面します。この問題に対する有益な解決策は、Hibernate などのオブジェクト・リレーショナル・マッピング (ORM) フレームワークを使用して、プロセスを自動化することです。InterSystems IRIS は標準の Hibernate インタフェースを提供しているため、大規模で複雑なオブジェクト階層には、これを使用することをお勧めします。

一方で、リアルタイムのデータ取得などのタスクを実行するアプリケーションにとって、主な問題はデータの複雑さではなく、速度です。Hibernate は (適切に最適化されていれば) 決して低速ではありませんが、単純なデータややや複雑なデータをきわめて迅速に格納および取得する必要があるタスクの場合は、XEP の方が優れた選択肢です。ほとんどの場合、XEP は Hibernate と JDBC のいずれよりもかなり迅速です。

2 XEP の仕組み

XEP は軽量の Java API で、Java オブジェクト・データを永続イベントとして投影します。永続イベントは、Java オブジェクト内のデータ・フィールドのコピーを含む InterSystems IRIS クラス (通常は **%Persistent** のサブクラス) のインスタンスです。他のこうしたインスタンスと同様に、オブジェクト・アクセス、SQL クエリ、直接グローバル・アクセスによって取得できます。

永続イベントを作成して格納するには、事前に XEP が対応する Java クラスを分析して、データベースにスキーマをインポートする必要があります。スキーマは、Java オブジェクトの保存に使用する永続イベント・クラスの構造を定義します。スキーマをインポートすることで、永続イベント・クラスのデータベース・エクステントが自動的に作成されます (まだ存在していない場合)。単純なスキーマのインポートで XEP が必要とするものは、この分析からの情報ですべてまかなわれる場合もあります。もっと複雑な構造では、XEP がインデックスを生成して、フィールドのインポートに関する既定のルールを上書きできるように、追加情報を提供できます。

クラスにスキーマを作成した後は、さまざまな XEP メソッドを使用して、イベントの格納、更新、削除、SQL クエリの実行、およびクエリ結果セットでの反復を行うことができます。

3 体験 : XEP の実践

それではご自身で実際に XEP を試してみましょう。この XepSimple デモは、きわめて小さなプログラムですが、XEP の主要機能のほとんどについて例を提供しているので、XEP API を使用する方法の概要を学ぶことができます。

3.1 開始の前に

この手順を使用するには、バージョン 1.8 の JDK および選択した Java IDE がインストールされた、操作するためのシステム、および接続先として、稼働している InterSystems IRIS インスタンスが必要です。InterSystems IRIS の選択肢としては、いくつかのタイプのラインセンス付与されたインスタンスおよび無料の評価版インスタンスがあります。操作しているシステムでインスタンスをホストする必要はありません (ただし、相互のネットワーク・アクセスが必要です)。操作を実行するインスタンスをまだ用意できていない場合にインスタンスのタイプ別の導入方法の詳細を確認するには、″InterSystems IRIS の基礎 : IDE の接続″ の ″InterSystems IRIS の導入″ を参照してください。同じドキュメントの ″InterSystems IRIS 接続情報″ および ″Java IDE″ の情報を使用して、IDE を InterSystems IRIS インスタンスに接続します。

3.2 サンプル・コードの追加

以下のデモは、XEP と InterSystems IRIS の操作方法を示しています。XepSimple デモは、きわめて小さなプログラムですが、XEP の主要機能のほとんどについて例を提供しているので、XEP API を使用する方法の概要を学ぶことができます。

ここにリストされているコードを単に確認することも、InterSystems GitHub リポジトリからコードをダウンロードして自身で実行することもできます。ダウンロードには、作業を開始するにあたって必要な情報がすべて含まれる ReadMe が含まれています。

デモ・プログラムは 4 つのセクションに分かれており、それぞれが 4 つの主要 XEP クラス EventPersister、Event、EventQuery、および EventQueryIterator のいずれかを紹介しています。

クラス XepSimple

Java

```
package xepsimple;
import com.intersystems.xep.*;
import xep.samples.SingleStringSample;

public class XepSimple {
  public static void main(String[] args) throws Exception {
// Generate 12 SingleStringSample objects for use as test data
    SingleStringSample[] sampleArray = SingleStringSample.generateSampleData(12);

// EventPersister
    EventPersister xepPersister = PersisterFactory.createPersister();
    xepPersister.connect("127.0.0.1",51773,"User","_SYSTEM","SYS"); // connect to localhost
    xepPersister.deleteExtent("xep.samples.SingleStringSample");   // remove old test data
    xepPersister.importSchema("xep.samples.SingleStringSample");   // import flat schema

// Event
    Event xepEvent = xepPersister.getEvent("xep.samples.SingleStringSample");
    for (int i=0; i < sampleArray.length; i++) {
      SingleStringSample sample = sampleArray[i];  // array initialized on line 8
      sample.name = "Sample object #" + i;
      xepEvent.store(sample);
      System.out.println("Persisted " + sample.name);
    }

// EventQuery
    String sqlQuery = "SELECT * FROM xep_samples.SingleStringSample WHERE %ID BETWEEN ? AND ?";

    EventQuery<SingleStringSample> xepQuery = xepEvent.createQuery(sqlQuery);
    xepQuery.setParameter(1,3);    // assign value 3 to first SQL parameter
    xepQuery.setParameter(2,12);   // assign value 12 to second SQL parameter
    xepQuery.execute();            // get resultset for IDs between 3 and 12
```

```
// EventQueryIterator
    EventQueryIterator<SingleStringSample> xepIter = xepQuery.getIterator();
    while (xepIter.hasNext()) {
      SingleStringSample newSample = xepIter.next();
      newSample.name = newSample.name + " has been updated";
      xepIter.set(newSample);
      System.out.println(newSample.name);
    }

    xepQuery.close();
    xepEvent.close();
    xepPersister.close();
  } // end main()
} // end class XepSimple
```

XepSimple は、以下のタスクを実行します。

- まず、サンプルのデータ・クラス xep.samples.SingleStringSample のメソッドを呼び出すことで、サンプルのオブジェクトがいくつか生成されます。
- EventPersister は、XEP の主なエントリ・ポイントで、データベースへの接続を提供します。

  このクラスは、xepPersister という名前のインスタンスを作成します。このインスタンスは、データベースへの接続を確立して、既存のサンプル・データをすべて削除します。次に、このクラスは、importSchema() を呼び出して、サンプル・クラスを分析し、スキーマをデータベースに送信します。その結果、SingleStringSample 永続オブジェクトを保持するエクステントが作成されます。

  EventPersister には、InterSystems IRIS インスタンスを接続するために必要な情報の変数 (`host`、`port`、`irisnamespace`、`username`、および `password`) が含まれます。"InterSystems IRIS の基礎 : IDE の接続" の "InterSystems IRIS 接続情報" で説明しているとおりに、インスタンスの適切な情報でこれらの変数を更新します。これは、IDE をインスタンスに接続したときに使用したものと同じ情報です。示されているとおりに **USER** ネームスペースを指定することも、インストールしたインスタンスで作成した別のネームスペースを使用することもできます。インスタンスをローカルにインストールして、接続でサーバ・アドレスとして `localhost` を使用している場合は、プログラムでローカルの共有メモリ接続が使用されます。これはむしろ、標準の TCP/IP 接続よりも高速です。
- Event は Java オブジェクトと対応するデータベース・オブジェクトとの間のインタフェースをカプセル化します。

  スキーマが生成されると、xepPersister は、サンプル・クラスに xepEvent という名前の Event オブジェクトを作成できます。ループで、SingleStringSample の各インスタンスに変更が加えられた後、これらのインスタンスがデータベースに永続化されます。xepEvent store() メソッドは、xepPersister で定義されている接続およびスキーマを利用します。
- EventQuery は、SQL クエリの準備と実行に使用されます。

  xepEvent オブジェクトの createQuery() メソッドにクエリ文字列を渡すことによって、xepQuery という名前の EventQuery<SingleStringSample> オブジェクトが作成されます。この文字列によって、2 つのパラメータ (? 文字) を受け取る SQL クエリが定義されます。setParameter() の呼び出しによってパラメータ値が定義され、execute() の呼び出しによってクエリの結果セットがフェッチされます。
- EventQueryIterator は、結果セットからの行の読み取り、および対応する永続オブジェクトの更新または削除に使用されます。

  xepQuery にはクエリの結果セットが含まれているので、getIterator() を呼び出すことによって、このオブジェクト用に xepIter という名前の反復子を作成できます。ループでこの反復子の next() メソッドが、データの各行を取得し、それを SingleStringSample オブジェクトに割り当てる処理に使用されます。次にこのオブジェクトに変更が加えられ、反復子の set() メソッドによって、データベース内の対応する永続オブジェクトが更新されます。
- 処理が完了したら、XEP オブジェクトに対して close() メソッドを呼び出すことによって、クリーン・アップが行われます。

3.3 SingleStringSample クラス

興味をお持ちの方のために、インターシステムズのサンプル・クラスのリストを示します。

xep.samples.SingleStringSample

Java

```
public class SingleStringSample {
  public  String name;
  public SingleStringSample() {}
  SingleStringSample(String str) {
      name = str;
  }

  public static SingleStringSample[] generateSampleData(int objectCount) {
      SingleStringSample[] data = new SingleStringSample[objectCount];
      for (int i=0;i<objectCount;i++) {
          data[i] = new SingleStringSample("single string test");
      }
      return data;
  }
}
```

このクラスが選択された理由のひとつは、XEP がこのクラスからスキーマを生成できるようにアノテーションを追加しておく必要はないためです。スキーマの最適化のために、ほとんどのクラスに 1 つ以上のアノテーションが必要です (これは、このドキュメントの範囲を超えたトピックです)。

3.4 次の手順

XepSimple デモは、詳細にとらわれずに XEP の基本機能を体験していただくことを目的としており、明らかに実際のプロダクション・コードのモデルではありません。例外の確認さえ行いませんでした。簡略化されている中でも最も重要な部分がサンプル・データです。スキーマの生成や最適化に役立つアノテーションやその他のメカニズムを必要としない、いくつかの小さな Java オブジェクトをクラスから永続化しました。実際のアプリケーションでは通常、いくつかのアノテーションが必要です (一般的には Hibernate よりも少数ですが)。

XEP をプロダクション・システムに導入する場合、スキーマの最適化、インデックスの制御、バッチ・ロード、およびその他の重要なタスク向けに XEP が提供するさまざまなツールを理解する必要があります。XEP の主要ドキュメント "InterSystems XEP による Java オブジェクトの永続化" には、こういった機能の包括的な説明が記載されています。このドキュメントの最後に示すソースでは、その他の面での InterSystems IRIS Java サポートを説明しています。

4 XEP およびオブジェクト永続性の詳細

Java オブジェクト永続性とその他の InterSystems Java 相互運用性テクノロジの詳細は、以下を参照してください。

- "JDBC とインターシステムズデータベース" には、JDBC を介した InterSystems IRIS への接続、基本情報、特別な機能に関する情報が掲載されており、実際に自分で試してみる機会が提供されています。これは InterSystems IRIS Java サポートをよく知るための最も簡単な出発点になります。
- "InterSystems JDBC ドライバでの Java の使用法" の “InterSystems Java 接続オプション” は、JDBC ドライバによって有効になるすべての InterSystems IRIS Java テクノロジの概要を示しています。
- InterSystems IRIS は、SQL テーブル、オブジェクト、および多次元ストレージを使用した簡単なデータベース・アクセスを実現する Java API を提供します。各アクセス・タイプの詳細は、以下のドキュメントを参照してください。
  - SQL テーブルへのアクセスについては、"InterSystems JDBC ドライバでの Java の使用法"。InterSystems JDBC ドライバにより、InterSystems IRIS は外部アプリケーションへの JDBC 接続を確立し、SQL 経由で外部データ・ソースにアクセスできるようになります。

- ネイティブの多次元ストレージへのアクセスについては、"Native SDK for Java の使用法"。InterSystems IRIS Native SDK を使用すると、InterSystems IRIS オブジェクト・インタフェースおよび SQL テーブル・インタフェースの基盤となるネイティブのツリーベースの多次元ストレージ・データ構造に直接アクセスできます。
- オブジェクトへのアクセスについては、"InterSystems XEP による Java オブジェクトの永続化"。XEP は、単純に中程度の複雑さのオブジェクト階層を操作するが、非常に高速のオブジェクト・データ永続性と検索が必要とされるトランザクション処理アプリケーションに対して最適化されています。

· "Java サード・パーティ API 用実装リファレンス" の "Hibernate のサポート" では、Hibernate の InterSystems IRIS 言語について説明しています。この言語は、Java プロジェクト内の大規模で複雑なオブジェクト階層で推奨される永続性テクノロジ JPA (Java Persistence Architecture) のサポートを実装します。