Hibernate JPA

Hibernate 是最流行的 ORM 框架之一，也是最早实现 JPA 的规范框架之一。它被 JBoss 收购后，目前作为 JBoss 的一个开源框架，它遵循 LGPL v2.1 开源许可协议，官方主页是 http://www.hibernate.org/。

14.1.1 Hibernate 与 JPA

Hibernate 3.2 以及以后的版本开始支持 JPA，如图 14-1 所示为 Hibernate 框架包含的所有子项目。其中，涉及 JPA 的子项目有三个，它们分别是：

— Hibernate Core：Hibernate 框架的核心实现。

— Hibernate Annotations：支持 JDK 5.0 的注释。

— Hibernate EntityManager：支持 JPA 的实现。

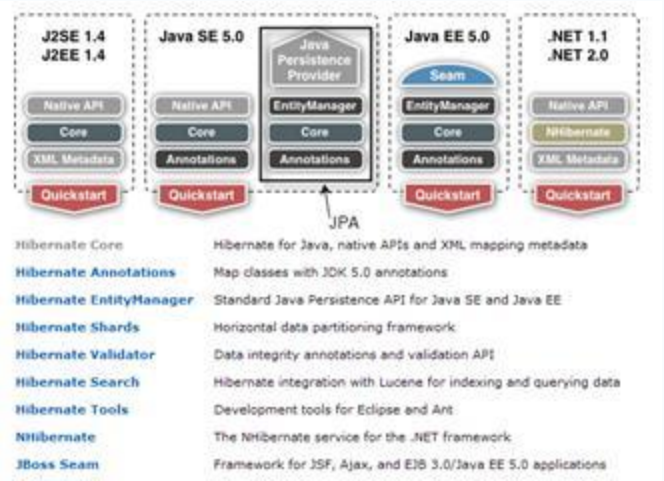

图 14-1 Hibernate 框架

14.1.2 Hibernate 下载

如果使用 JBoss 应用服务器，由于 JBoss 中已经集成了 Hibernate，所以不需要额外下载 Hibernate 类包。若在其他的环境中使用 Hibernate 实现，则需要下载其使用的类包。

目前最新版本为 Hibernate Core 3.2.5.GA、Hibernate Annotations 3.3.0 GA 和 Hibernate EntityManager 3.3.1 GA。使用浏览器打开 http://hibernate.org/30.html，进入到 Hibernate 的下载页面，如图 14-2 所示。

分别下载后找到 hibernate3.jar、hibernate-entitymanager.jar 和 hibernate-annotations.jar 三个 JAR 包文件，这三个 JAR 包文件是 Hibernate 实现 JPA 所必需的类包。运行时还要导入运行 Hibernate 所依赖的第三方类库，请参阅 Hibernate 官方手册。

若下载了旧版本的 Hibernate Core，要注意下载兼容的 Hibernate EntityManager 版本和 Hibernate Annotations 版本。如图 14-3 所示为 Hibernate 各个模块不同版本的兼容性。

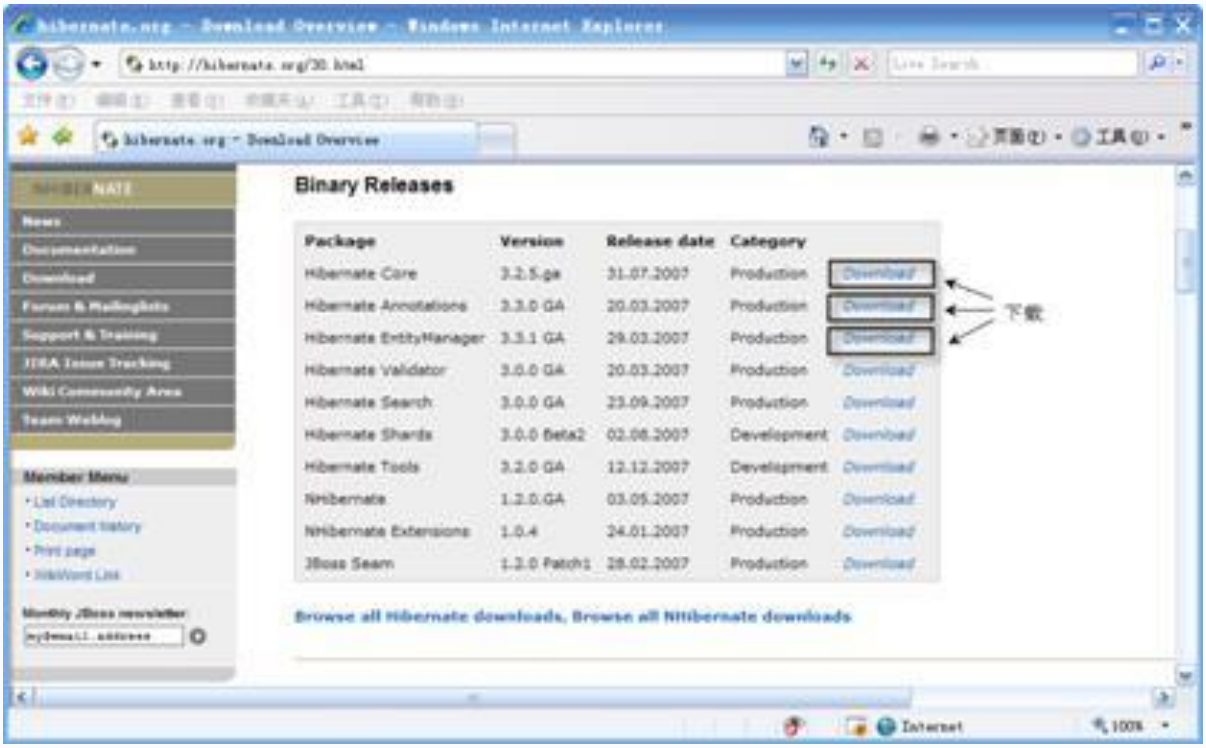

图 14-2 Hibernate 下载页面

Package	Version	Core	Annotations	EntityManager	Validator	Search	Shards	Tools
Hibernate Core	3.2.2 GA	-	3.2.x, 3.3.x	3.2.x, 3.3.x	3.0.x	3.0.x	3.0.x	3.2.0
Hibernate Annotations	3.3.0 GA	3.2.x	-	3.3.x	3.0.x	3.0.x	Not compatible	3.2.0
Hibernate EntityManager	3.3.1 GA	3.2.x	3.3.0.GA	-	3.0.x	3.0.x	Not compatible	3.2.0
Hibernate Validator	3.0.0 GA	3.2.x	3.3.x	3.3.x	-	3.0.x	3.0.x	3.2.0
Hibernate Search	3.0.0 GA	>= 3.2.2	3.3.x	3.3.x	3.0.x	-	3.0.x	(3.2.0)
Hibernate Shards	3.0.0 Beta2	3.2.x	3.3.x	Not compatible	3.0.x	3.0.x	-	-
Hibernate Tools	3.2.0 CR1	3.2.x	3.2.x and 3.3.x	3.2.x and 3.3.x	3.0.x	(3.2.0)	-	-

图 14-3　Hibernate 不同版本的兼容性比较

14.1.3　配置方式

Hibernate JPA 自定义配置可以通过多种方式进行配置，如下面列举的几种方法。

方法一：

在 persistence.xml 文件中配置，如下所示。

```
<persistence>

   <persistence-unit name="jpaUnit"  transaction-type="RESOURCE_LOCAL">

<provider>org.hibernate.ejb.HibernatePersistence</provider>
   <class>org.shirdrn.entity.MyUser</class>
       <properties>
        <property name="hibernate.dialect"
value="org.hibernate.dialect.SQLServerDialect"/>
        <property name="hibernate.connection.driver_class"
value="com.microsoft.jdbc.sqlserver.SQLServerDriver"/>
        <property name="hibernate.connection.username" value="sa"/>
        <property name="hibernate.connection.password" value="sa"/>
        <property name="hibernate.connection.url"
value="jdbc:microsoft:sqlserver://localhost:1433;databasename=student"/>
        <property name="hibernate.max_fetch_depth" value="1"/>
        <property name="hibernate.hbm2ddl.auto" value="update"/>
            <!-- 调整JDBC抓取数量的大小： Statement.setFetchSize() -->
            <property name="hibernate.jdbc.fetch_size" value="18"/>
             <!-- 调整JDBC批量更新数量 -->
            <property name="hibernate.jdbc.batch_size" value="10"/>
             <!-- 显示最终执行的SQL  -->
            <property name="hibernate.show_sql" value="true"/>
```

```
            <!-- 格式化显示的SQL -->
            <property name="hibernate.format_sql" value="true"/>
    </properties> </persistence-unit>
</persistence>
```

其中，“hibernate.show_sql ”为可配置的属性，Hibernate JPA 还提供很多不同属性的配置。

方法二：

通过代码，在创建 EntityManagerFactory 时指定，如下所示。

```
Map configOverrides = new HashMap();
configOverrides.put("hibernate.format_sql ", true);
EntityManagerFactory programmaticEmf =
Persistence.createEntityManagerFactory("jpaUnit", configOverrides);
```

当同时使用方法一和方法二设置时，方法二的方式为有效的配置。

方法三：

使用 Hibernate 专有的配置文件来配置，但首先要在 persistence.xml 文件中配置“hibernate.ejb.cfgfile”指定配置文件的位置，如下所示。

```
<persistence>
```

```
    <persistence-unit name="jpaUnit"  transaction-type="RESOURCE_LOCAL">
        <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <properties>
        <property name="hibernate.connection.driver_class"
            value="com.mysql.jdbc.Driver" />
        <property name="hibernate.connection.url"
            value="jdbc:mysql://localhost:3306/jpademo" />
        <property name="hibernate.connection.username" value="root" />
        <!—可选，配置 Hibernate 配置文件-->
               < property name="hibernate.ejb.cfgfile"
            value="/com/fengmanfei/jpa/hibernate.cfg.xml"/ >
    </properties>
</persistence-unit>
</persistence>
```

其中，“/com/fengmanfei/jpa/hibernate.cfg.xml”为 Hibernate 配置文件的保存位置。使用这种方式，适用于将现有 Hibernate 应用移植到 JPA 应用中来。但要注意，方法三的优先级最低，如果与方法一和方法二冲突，则方法一或方法二中的配置有效。

14.1.4 基本配置

方法一和方法二是 JPA 的标准配置，方法三是 Hibernate 特有的配置。并不是所有的属性都可以通过这三种方式配置,其中一些属性必须通过方法一和方法二来配置,这些属性的详细说明如下所示。

— 属性名：hibernate.ejb.classcache.<classname>

描述：指定缓存实体对象，<classname>为缓存类的全名，值为缓存类型，以逗号分隔。

示例如下：

```
<property name="hibernate.ejb.classcache. com.fengmanfei.jpa.entity.Customer"

value="read-write"/>
```

— 属性名：hibernate.ejb.collectioncache.<collectionrole>

描述：指定集合实体类缓存，设置同上。<collectionrole>为集合类的全名，值为缓存类型，以逗号分隔。

示例如下：

```
<property name="hibernate.ejb.collectioncache.com.fengmanfei.jpa.entity.Customer. orders"

value="read-write , RegionName "/>
```

★ 提示 ★

读者若想了解更多的缓存设置，请参阅 JBoss Cache 的相关文档。

— 属性名：hibernate.ejb.cfgfile

描述：指定使用 Hibernate 配置文件中的配置。

示例如下：

```
< property name="hibernate.ejb.cfgfile"  value="/com/fengmanfei/jpa/hibernate.cfg.xml"/ >
```

— 属性名：hibernate.archieve.autodetection

描述：创建 Entity Manager 时搜索文件的类型，多个值之间用逗号分隔。

可选值：

— class：.class 类文件。

— hbm：Hibernate 配置文件。

默认两个都搜索。

示例如下：

```
<property name="hibernate.archive.autodetection" value="class,hbm"/>
```

— 属性名：hibernate.ejb.interceptor

描述：自定义拦截器类名，拦截器必须实现了 org.hibernate.Interceptor 接口，并且有无参的构造方法。

示例如下：

```
<property name=" hibernate.ejb.interceptor "

value="com.fengmanfei.jpa.interceptor.MyInterceptor"/>
```

— 属性名：hibernate.ejb.naming_strategy

描述：设置注释命名策略。

可选值：

— EJB3NamingStrategy（默认）：EJB3 规范的命名实现。

— DefaultComponentSafeNamingStrategy：在默认的 EJB3NamingStrategy 上进行了扩展，允许在同一实体中使用两个同类型的嵌入对象而无须额外的声明。

示例如下：

```
<property name=" hibernate.ejb.naming_strategy "
value=" DefaultComponentSafeNamingStrategy "/>
```

*— 属性名：hibernate.ejb.event.<eventtype>

描述：配置事件监听器，其中<eventtype>为监听的事件类型，事件类型如表 14-1 中列举所示。而值则为具体监听器类的全名，如果有多个则使用逗号分隔。自定义拦截器类，拦截器必须实现了 org.hibernate.Interceptor 接口，并且有无参的构造方法，在 JPA 的环境中，尽量继承表 14-1 中的时间监听器类。

表 14-1　可选的监听事件类型

事件类型	监听器类
flush	org.hibernate.ejb.event.EJB3FlushEventListener
auto-flush	org.hibernate.ejb.event.EJB3AutoFlushEventListener
delete	org.hibernate.ejb.event.EJB3DeleteEventListener
flush-entity	org.hibernate.ejb.event.EJB3FlushEntityEventListener
merge	org.hibernate.ejb.event.EJB3MergeEventListener
create	org.hibernate.ejb.event.EJB3PersistEventListener
create-onflush	org.hibernate.ejb.event.EJB3PersistOnFlushEventListener
save	org.hibernate.ejb.event.EJB3SaveEventListener
save-update	org.hibernate.ejb.event.EJB3SaveOrUpdateEventListener

续表

事件类型	监听器类
pre-insert	org.hibernate.secure.JACCPreInsertEventListener,org.hibernate.valitator.event.ValidateEventListener
pre-update	org.hibernate.secure.JACCPreUpdateEventListener,org.hibernate.valitator.event.ValidateEventListener
pre-delete	org.hibernate.secure.JACCPreDeleteEventListener
pre-load	org.hibernate.secure.JACCPreLoadEventListener
post-delete	org.hibernate.ejb.event.EJB3PostDeleteEventListener
post-insert	org.hibernate.ejb.event.EJB3PostInsertEventListener
post-load	org.hibernate.ejb.event.EJB3PostLoadEventListener
post-update	org.hibernate.ejb.event.EJB3PostUpdateEventListener

示例如下：

```
<property name="hibernate.ejb.event.create" value="com.fengmanfei.listener. CreateListener" />
```

其中，CreateListener 继承 org.hibernate.ejb.event.EJB3PersistEventListener 类，代码如下所示。

```
package com.fengmanfei.listener;

import org.hibernate.HibernateException;

import org.hibernate.ejb.event.EJB3PersistEventListener;

import org.hibernate.event.PersistEvent;

public class CreateListener extends EJB3PersistEventListener {

    // 覆盖父类中的方法

    @Override

    public void onPersist(PersistEvent event) throws HibernateException {

        super.onPersist(event);

        //代码处理
```

```
    }

}
```

— 属性名：hibernate.ejb.use_class_enhancer

描述：是否启用应用服务器扩展类。

可选值：

- true：启用扩展类。
- false（默认）：禁用扩展类。

示例如下：

```
<property name=" hibernate.ejb.use_class_enhancer "  value=" true" />
```

— 属性名：hibernate.ejb.discard_pc_on_close

描述：是否在执行 clear()时脱离持久化上下文。

可选值：

- true：执行 clear()时脱离持久化上下文。
- false（默认）：执行 clear()时不脱离持久化上下文。

示例如下：

```
<property name=" hibernate.ejb.discard_pc_on_close "  value=" true" />
```

14.1.5 配置日志

Hibernate 使用 Apache commons-logging 来为各种事件记录日志。commons-logging 将直接将日志输出到 Apache Log4j（如果在类路径中包括 log4j.jar）或 JDK1.4 logging （如果运行在 JDK1.4 或以上的环境下）。

如果使用 Log4j，需要将 log4j.properties 文件保存在类路径中。Hibernate 根据对日志进行了详细的分类，以便能够控制日志的的输出信息，这些日志类别如表 14-2 所示。

表 14-2 Hibernate JPA 实现日志类别

属性名	描 述
org.hibernate.SQL	记录 SQL DML 语句
org.hibernate.type	记录 JDBC 参数
org.hibernate.tool.hbm2ddl	记录 SQL DDL 语句
org.hibernate.pretty	记录提交实体时，相关联的 20 个实体的状态
org.hibernate.cache	记录所有二级缓存
org.hibernate.transaction	记录事务相关的操作
org.hibernate.jdbc	记录获取 JDBC 资源的操作
org.hibernate.hql.ast.AST	记录 HQL 和 SQL AST 的查询语句
org.hibernate.secure	记录 JAAS 认证请求
org.hibernate	记录所有信息，建议在调试开发阶段设置

例如，下面为 log4j.properties 配置日志的示例代码。

```
### log4j 基本配置 ###

log4j.appender.file=org.apache.log4j.FileAppender

log4j.appender.file.File=hibernate.log

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{ABSOLUTE} %5p %c{1}:%L - %m%n
```

设置日志级别###

```
log4j.rootLogger=info, stdout
```

###输出 hibernate 调试过程中的错误日志

```
log4j.logger.org.hibernate=debug
```

###输出 HQL 查询调试日志

```
log4j.logger.org.hibernate.hql.ast.AST=debug
```

输出 SQL 语句调试日志

```
log4j.logger.org.hibernate.SQL=debug
```

输出 JDBC 参数查询的日志

```
log4j.logger.org.hibernate.type=info
```

输出缓存日志

```
log4j.logger.org.hibernate.cache=debug
```

输出事务日志###

```
log4j.logger.org.hibernate.transaction=debug
```

###输出获取 JDBC 资源日志###

```
log4j.logger.org.hibernate.jdbc=debug
```

14.1.6 配置缓存

Hibernate 除了自动对 Session 级别的事务进行一级缓存外，二级缓存的优化是 Hibernate 实现的一个亮点之一，有关二级缓存的属性如下所示。

— 属性名：hibernate.cache.provider_class

描述：二级缓存实现的类全名，所使用的缓存都需要实现 org.hibernate.cache. CacheProvider 接口，Hibernate 已经实现了一些缓存，开发人员可以直接配置使用，同时要启用二级缓存，配置 hibernate.cache.use_second_level_cache 为 true。

可选值：

org.hibernate.cache.HashtableCacheProvide、

org.hibernate.cache.EhCacheProvider、

org.hibernate.cache.OSCacheProvider、

org.hibernate.cache.SwarmCacheProvider

和 org.hibernate. cache.Tree CacheProvider 等。

示例如下：

```
<property name=" hibernate.cache.provider_class "

value=" org.hibernate.cache.HashtableCacheProvide "/>
```

★ 提示 ★

有关各种缓存实现的详细区别，读者可以参阅 Hiberante Core 的相关文档。

— 属性名：hibernate.cache.use_minimal_puts

描述：是否优化二级缓存来最小化读写操作，集群时的缓存优化。

可选值：

- true（默认）：启用最小化读写操作。
- false：禁用最小化读写操作。

示例如下：

```
<property name=" hibernate.cache.use_minimal_puts "  value=" false" />
```

— 属性名：hibernate.cache.use_query_cache

描述：是否缓存查询结果。

可选值：

- true：缓存查询结果。
- false：不缓存查询结果。

示例如下：

```
<property name=" hibernate.cache.use_query_cache "  value=" true" />
```

— 属性名：hibernate.cache.use_second_level_cache

描述：是否启用二级缓存。

可选值：

— true：启用二级缓存。

— false：不使用二级缓存。

示例如下：

```
<property name=" hibernate.cache.use_second_level_cache "  value=" true" />
```

— 属性名：hibernate.cache.query_cache_factory

描述：设置自定义的查询缓存类全名，缓存类必须实现 org.hibernate.cache.QueryCache 接口。

可选值：

— org.hibernate.cache.StandardQueryCache（默认）。

— 自定义缓存实现类。

示例如下：

```
<property name=" hibernate.cache.query_cache_factory "
value=" com.fengmanfei.cache.MyCache" />
```

— 属性名：hibernate.cache.region_prefix

描述：二级缓存的前缀名称。

示例如下：

```
<property name=" hibernate.cache.region_prefix "  value=" jpa" />
```

— 属性名：hibernate.cache.use_structured_entries

描述：是否使用结构化的方式缓存对象。

可选值：

- true：结构化方式缓存对象。
- false：不使用结构化的方式缓存对象。

示例如下：

```
<property name=" hibernate.cache.use_structured_entries "  value=" true" />
```

14.1.7 配置 JDBC 和数据库

Hibernate 自定义 JDBC 和数据库配置属性如下所示。

— 属性名：hibernate.jdbc.fetch_size

描述：JDBC 抓取记录的大小，相当于设置 Statement.setFetchSize()，默认值为 25。

示例如下：

```
<property name="hibernate.jdbc.fetch_size"  value="50" />
```

— 属性名：hibernate.jdbc.batch_size

描述：JDBC2 批量更新的大小，建议设置为 5～30 之间的值，默认为 5。

示例如下：

```
<property name="hibernate.jdbc.batch_size"  value=" 25" />
```

— 属性名：hibernate.jdbc.batch_versioned_data

描述：JDBC 执行批量操作时，是否同时更新版本数据。

可选值：

— true（默认）：执行批量操作 executeBatch()返回成功的记录数，并且更新版本数据。

— false：批量操作后不更新版本数据。

示例如下：

```
<property name="hibernate.jdbc.batch_versioned_data"  value="false" />
```

— 属性名：hibernate.jdbc.batch_versioned_data

描述：JDBC 执行批量操作时，是否同时更新版本数据。

可选值：

— true（默认）：执行批量操作 executeBatch()返回成功的记录数，并且更新版本数据。

— false：批量操作后不更新版本数据。

示例如下：

```
<property name="hibernate.jdbc.batch_versioned_data"  value="false" />
```

— 属性名：hibernate.jdbc.use_scrollable_resultset

描述：是否允许 Hibernate 使用 JDBC2 的可滚动结果集。

可选值：

— true（默认）：可使用可滚动结果集，只有在使用用户提供的 JDBC 连接时，才需要设置为启用。

— false：不可使用滚动结果集。

示例如下：

```
<property name="hibernate.jdbc.use_scrollable_resultset"  value="false" />
```

— 属性名：hibernate.jdbc.use_streams_for_binary

描述：是否 JDBC 以二进制方式读取。

可选值：

— true（默认）：以二进制方式读取。

— false：以二进制方式读取，而以序列化方式读取。

示例如下：

```
<property name="hibernate.jdbc.use_streams_for_binary"  value="false" />
```

— 属性名：hibernate.jdbc.use_get_generated_keys

描述：是否使用 JDBC3 插入记录时使用 PreparedStatement.getGeneratedKeys()生成主键。

可选值：

— true：使用 PreparedStatement.getGeneratedKeys()生成主键。

— false（默认）：使用 Hibernate 自定义的生成策略。

示例如下：

```
<property name="hibernate.jdbc.use_get_generated_keys"  value="true" />
```

— 属性名：hibernate.connection.isolation

描述：JDBC 事务隔离级别，请读者查阅 java.sql.Connection 文档了解各个级别的类型，例如 1、2、4（默认）、8。

示例如下：

```
<property name="hibernate.connection.isolation"  value="8" />
```

— 属性名：hibernate.connection.autocommit

描述：是否使用 JDBC 自动提交。

可选值：

— true（默认）：自动提交。

— false：不自动提交。

示例如下：<property name="hibernate.connection.autocommit" value="false" />

— 属性名：hibernate.connection.driver_class

描述：数据连接的驱动类的全称，不同的数据库实现类不同。

示例如下：

```
<property name="hibernate.connection.driver_class"  value="com.mysql.jdbc.Driver" />
```

— 属性名：hibernate.connection.url

描述：数据连接的 URL。

示例如下：

```
<property name="hibernate.connection.url"  value=" jdbc:mysql://localhost:3306/ jpademo" />
```

— 属性名：hibernate.connection.username

描述：数据连接的用户名。

示例如下：

```
<property name="hibernate.connection.username "  value="root" />
```

— 属性名：hibernate.connection.password

描述：数据连接的密码。

示例如下：

```
<property name="hibernate.connection.password "  value="123" />
```

— 属性名：hibernate.dialect

描述：指定不同的数据库，Hibernate 底层会根据不同的数据库生成的 SQL 进行优化，取值如表 14-3 所示。

表 14-3　Hibernate JPA 实现不同数据库相关配置的属性

属性名	描　述
DB2	org.hibernate.dialect.DB2Dialect
DB2 AS/400	org.hibernate.dialect.DB2400Dialect
DB2 OS390	org.hibernate.dialect.DB2390Dialect
PostgreSQL	org.hibernate.dialect.PostgreSQLDialect
MySQL	org.hibernate.dialect.MySQLDialect
MySQL InnoDB	org.hibernate.dialect.MySQLInnoDBDialect
MySQL with MyISAM	org.hibernate.dialect.MySQLMyISAMDialect
Oracle	org.hibernate.dialect.OracleDialect
Oracle 9i/10g	org.hibernate.dialect.Oracle9Dialect
Sybase	org.hibernate.dialect.SybaseDialect
Sybase Anywhere	org.hibernate.dialect.SybaseAnywhereDialect
Microsoft SQL Server	org.hibernate.dialect.SQLServerDialect
SAP DB	org.hibernate.dialect.SAPDBDialect
Informix	org.hibernate.dialect.InformixDialect
HypersonicSQL	org.hibernate.dialect.HSQLDialect

续表

属性名	描　述
Ingres	org.hibernate.dialect.IngresDialect
Progress	org.hibernate.dialect.ProgressDialect
Mckoi SQL	org.hibernate.dialect.MckoiDialect
Interbase	org.hibernate.dialect.InterbaseDialect
Pointbase	org.hibernate.dialect.PointbaseDialect

示例如下：

```
<property name=" hibernate.dialect "  value="org.hibernate.dialect.MySQLDialect" />
```

14.1.8　其他的常用配置

除了前面几节列举的配置外，Hibernate 的 JPA 实现还有一些常用的配置，如下所示。

— 属性名：hibernate.show_sql

描述：是否输出 SQL 语句。

可选值：

— true（默认）：输出 SQL，相当于日志中设置 org.hibernate.SQL 的类别值为 debug。

— false：不输出 SQL。

示例如下：

```
<property name="hibernate.show_sql"  value="false" />
```

— 属性名：hibernate.format_sql

描述：是否格式化输出 SQL 语句。

可选值：

— true（默认）：格式化输出 SQL。

— false：不格式化输出 SQL。

示例如下：

```
<property name="hibernate.format_sql "  value="false" />
```

— 属性名：hibernate.use_sql_comments

描述：是否输出 SQL 注释。

可选值：

- true（默认）：输出 SQL 注释，有助于调试。
- false：不输出 SQL 注释。

示例如下：

```
<property name="hibernate.use_sql_comments"  value="false" />
```

— 属性名：hibernate.generate_statistics

描述：是否收集与运行性能有关的参数。

可选值：

- true（默认）：收集与运行性能有关的参数。
- false：不收集与运行性能有关的参数。

示例如下：

```
<property name="hibernate.generate_statistics"  value="false"/>
```

— 属性名：hibernate.hbm2ddl.auto

描述：对 DDL 的自动生成方式。

可选值：

— create-drop：删除后重新创建。

— create：只创建新的。

— update：更新。

— validate：只进行验证。

示例如下：

```
<property name=" hibernate.hbm2ddl.auto "  value="create-drop"/>
```

— 属性名：hibernate.default_schema

描述：生成的 SQL 默认的 schema 名称。

示例如下：

```
<property name="hibernate.default_schema"  value="test"/>
```

— 属性名：hibernate.default_catalog

描述：生成的 SQL 默认的 catalog 名称。

示例如下：

```
<property name="hibernate.default_catalog"  value="test"/>
```

— 属性名：hibernate.max_fetch_depth

描述：一对一和多对一映射时，实体加载的最大深度，0 表示不抓取相关实体。建议值在 0～3 之间，默认为 1。

示例如下：

```
<property name="hibernate.max_fetch_depth"  value="2"/>
```

— 属性名：hibernate.default_batch_fetch_size

描述：加载相关联的实体集合时，所加载的相关实体个数。建议使用 2 的倍数值，例如 4、8（默认）、16。

示例如下：

```
<property name="hibernate.default_batch_fetch_size"  value="16"/>
```

大多数情况下，Hibernate JPA 中使用默认的设置就能基本满足需求，但当需求变化时，需要进行一些特殊的优化时，就可以通过自定义的一些属性来实现目标。