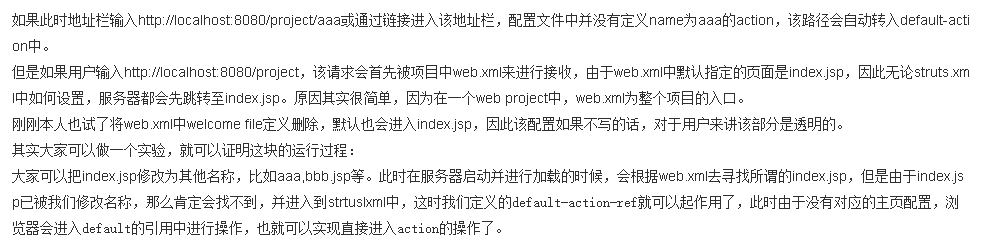
Struts2

1. Struts2 对于default-action-ref配置的解析思路





Hibernate4

hibernate提供了产生自动增长类型主键的多种策略

Generator 为每个 POJO 的实例提供唯一标识。一般情况，我们使用“native”。class 表示采用由生成器接口net.sf.hibernate.id.IdentifierGenerator 实现的某个实例，其中包括： 

“assigned”   
主键由外部程序负责生成，在 save() 之前指定一个。

(没怎么用过。一般不会用手工方式赋值主键，除非有特殊的需求。)

“hilo”   
通过hi/lo 算法实现的主键生成机制，需要额外的数据库表或字段提供高位值来源。 

“seqhilo”   
与hilo 类似，通过hi/lo 算法实现的主键生成机制，需要数据库中的 Sequence，适用于支持 Sequence 的数据库，如Oracle。 

“increment”   
主键按数值顺序递增。此方式的实现机制为在当前应用实例中维持一个变量，以保存着当前的最大值，之后每次需要生成主键的时候将此值加1作为主键。这种方式可能产生的问题是：不能在集群下使用。

(最好不要用。如果有其他程序访问、修改数据库，那就恐怖了。)

“identity”   
采用数据库提供的主键生成机制。如DB2、SQL Server、MySQL 中的主键生成机制。 

“sequence”   
采用数据库提供的 sequence 机制生成主键。如 Oralce 中的Sequence。

（这种地方可以解决上面用native时候的问题，但是需要堆数据库做一些其他配置。）

“native”   
由 Hibernate 根据使用的数据库自行判断采用 identity、hilo、sequence 其中一种作为主键生成方式。

（我最常用的。可以保证多个数据库之间的可移植性。但是有可能有时候会有问题：因为不能控制id值，在数据倒表的时候可能无法满足业务需要。  
）

uuid

理论上可以保证多个数据库生成的ID在一个系统里唯一，有时候挺有用。但是效率稍微低点（其实都无所谓）。

“uuid.hex”   
由 Hibernate 基于128 位 UUID 算法生成16 进制数值（编码后以长度32 的字符串表示）作为主键。 

“uuid.string”   
与uuid.hex 类似，只是生成的主键未进行编码（长度16），不能应用在 PostgreSQL 数据库中。 

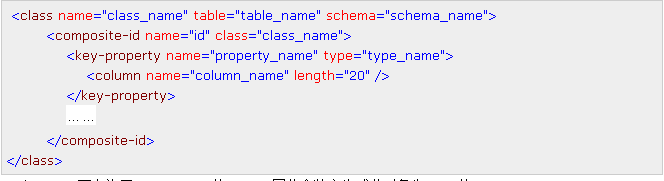
“foreign”   
使用另外一个相关联的对象的标识符作为主键。

(在one-to-one的时候可能会用到。)

 在使用数据库自动生成主键的时候，SQL语句会有所不同：有些数据库不许你填主键，有些要求你该字段必须为null，有些会完全忽略你写的主键的值。

Hibernate逆向工程问题

Hibernate逆向生成无主键的数据表时产生的hbm如：



Hibernate不允许无PrimaryKey的ORM，因此会独立生成此对象为ORM的PrimaryKey

  当需要查询此表，如果此对象中有任意一个字段为NULL，则整条记录都为NULL

  因为Hibenrate约束主键不能为空。

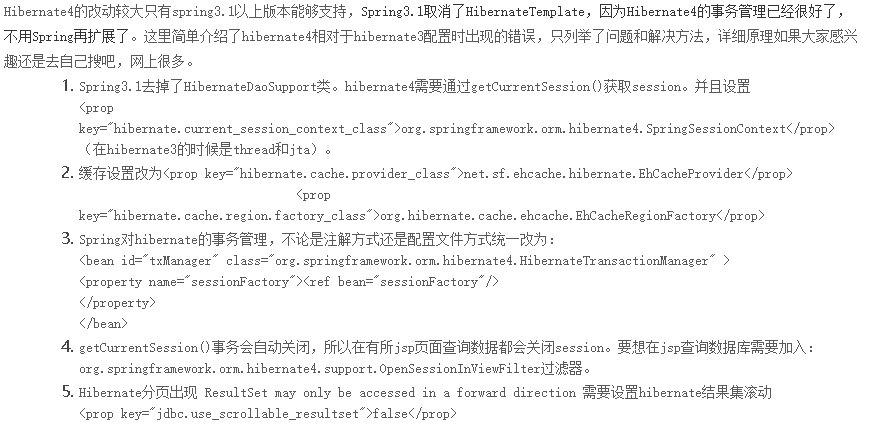
  解决办法：可将composite-id标签里可能会为空的字段提取到composite-id外，  
  将数据表所对应的POJO添加字段，字段为提取到composite-id标签外的字段属性

  HQL查询时直接用对应的类索引属性查询，无需经过第三方生成类。

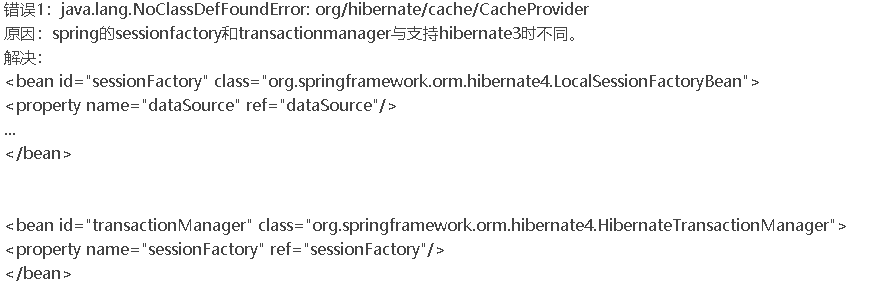
NoClassDefFoundError异常解决：

1. jar包中查看是否有该类，没有的话重新下载；
2. 查看jar包是否发布到webapps对应的项目中
3. Jar包冲突

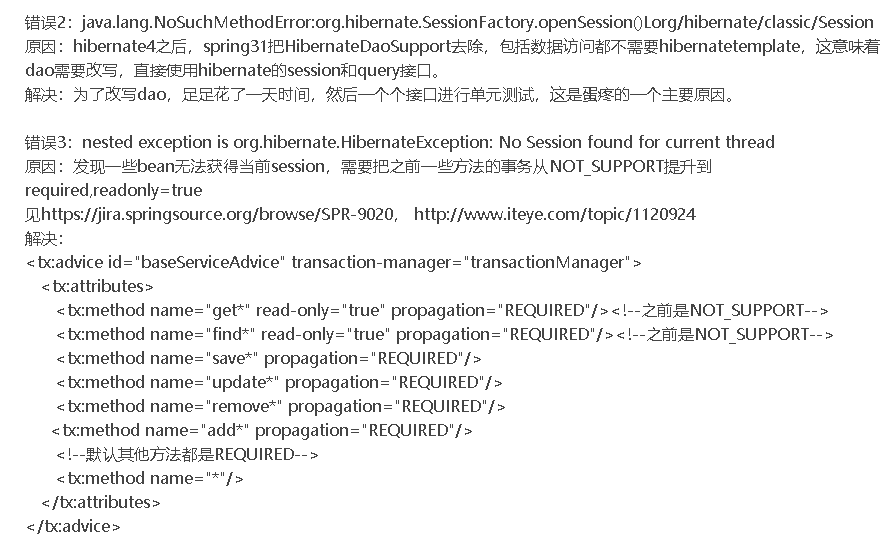
注意Hibernate4在开发当中的一些改变



错误1



错误2、3



错误4



总结



对web.xml文件、spring配置文件中的classpath的理解



[commons-logging 和 log4j 之间的关系](http://zachary-guo.iteye.com/blog/361177)

真正的记录日志的工具是 log4j 和 sun 公司提供的日志工具。而 commons-logging 把这两个(实际上，在 org.apache.commons.logging.impl 包下，commons-logging 仅仅为我们封装了 log4j 和 sun logger)记录日志的工具重新封装了一遍(Log4JLogger.java 和 Jdk14Logger.java)，可以认为 org.apache.commons.logging.Log 是个傀儡，它只是提供了对外的统一接口。因此我们只要能拿到 org.apache.commons.logging.Log，而不用关注到底使用的是 log4j 还是 sun logger。

配置 mybatis的 log4j.properties

### 设置Logger输出级别和输出目的地 ###

log4j.rootLogger=debug,stdout,logfile

### 把日志信息输出到控制台 ###

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

#log4j.appender.stdout.Target=System.err

log4j.appender.stdout.layout=org.apache.log4j.SimpleLayout

### 把日志信息输出到文件：jbit.log ###

log4j.appender.logfile=org.apache.log4j.FileAppender

log4j.appender.logfile.File=jbit.log

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

log4j.appender.logfile.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %F %p %m%n

###显示SQL语句部分

log4j.logger.com.ibatis=DEBUG

log4j.logger.com.ibatis.common.jdbc.SimpleDataSource=DEBUG

log4j.logger.com.ibatis.common.jdbc.ScriptRunner=DEBUG

log4j.logger.com.ibatis.sqlmap.engine.impl.SqlMapClientDelegate=DEBUG

log4j.logger.java.sql.Connection=DEBUG

log4j.logger.java.sql.Statement=DEBUG

log4j.logger.java.sql.PreparedStatement=DEBUG

手动加载启动日志（当log4j.properties文件不在src目录下时应用程序不能自动加载）

log4jPath=System.getProperty("user.dir")+"/config/log4j.properties";

PropertyConfigurator.configure(log4jPath);

java生成UUID通用唯一识别码 (Universally Unique Identifier)

UUID 是指在一台机器上生成的数字，它保证对在同一时空中的所有机器都是唯一的。通常平台会提供生成的API。按照开放软件基金会(OSF)制定的标准计算，用到了以太网卡地址、纳秒级时间、芯片ID码和许多可能的数字

　　UUID由以下几部分的组合：

　　（1）当前日期和时间，UUID的第一个部分与时间有关，如果你在生成一个UUID之后，过几秒又生成一个UUID，则第一个部分不同，其余相同。

　　（2）时钟序列

　　（3）全局唯一的IEEE机器识别号，如果有网卡，从网卡MAC地址获得，没有网卡以其他方式获得。

UUID的唯一缺陷在于生成的结果串会比较长。关于UUID这个标准使用最普遍的是微软的GUID(Globals Unique Identifiers)。而标准的UUID格式为：xxxxxxxx-xxxx-xxxx-xxxxxx-xxxxxxxxxx (8-4-4-4-12)，

使用UUID的好处在分布式的软件系统中（比如：DCE/RPC, COM+,CORBA）就能体现出来，它能保证每个节点所生成的标识都不会重复，并且随着WEB服务等整合技术的发展，UUID的优势将更加明显。根据使用的特定机制，UUID不仅需要保证是彼此不相同的，或者最少也是与公元3400年之前其他任何生成的通用惟一标识符有非常大的区别。

实例：

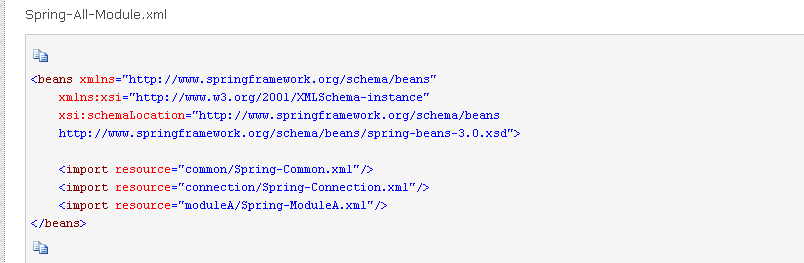


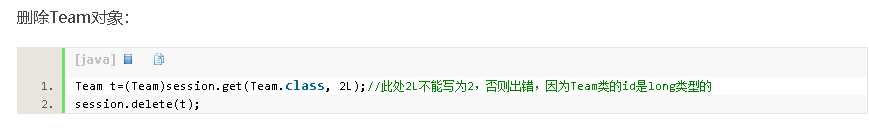
Spring4系列-多个配置文件的整合

代码中加载以上3个xml配置文件

ApplicationContext context = new ClassPathXmlApplicationContext(new String[] {"Spring-Common.xml","Spring-Connection.xml","Spring-ModuleA.xml"});

但是这种方法不易组织并且不好维护，最好的方法是在一个单独的xml的配置文件中组织其他所有的xml配置文件。例如，可以创建一个Spring-All-Module.xml文件，然后将其他的xml配置文件导入到Spring-All-Module.xml中，就像下边这样，





[Session.evict()方法](http://***/forum/blogPost/list/3771.html;jsessionid=53CBD06464EB9607E75084664E24711F" \o "Hibernate学习笔记 - Session.evict(user)方法)



Hibernate注册服务对象





