##############################错题记录#############################

1.Statement、PreparedStatement和CallableStatement都是接口(interface)。   
2.Statement继承自Wrapper、PreparedStatement继承自Statement、CallableStatement继承自PreparedStatement。   
3.   
Statement接口提供了执行语句和获取结果的基本方法；   
PreparedStatement接口添加了处理 IN 参数的方法；   
CallableStatement接口添加了处理 OUT 参数的方法。   
4.   
a.Statement:   
普通的不带参的查询SQL；支持批量更新,批量删除;   
b.PreparedStatement:   
可变参数的SQL,编译一次,执行多次,效率高;   
安全性好，有效防止Sql注入等问题;   
支持批量更新,批量删除;   
c.CallableStatement:   
继承自PreparedStatement,支持带参数的SQL操作;   
支持调用存储过程,提供了对输出和输入/输出参数(INOUT)的支持;   
  
Statement每次执行sql语句，数据库都要执行sql语句的编译 ，   
最好用于仅执行一次查询并返回结果的情形，效率高于PreparedStatement。   
  
PreparedStatement是预编译的，使用PreparedStatement有几个好处   
1. 在执行可变参数的一条SQL时，PreparedStatement比Statement的效率高，因为DBMS预编译一条SQL当然会比多次编译一条SQL的效率要高。   
2. 安全性好，有效防止Sql注入等问题。   
3.  对于多次重复执行的语句，使用PreparedStament效率会更高一点，并且在这种情况下也比较适合使用batch；   
4.  代码的可读性和可维护性。

**事务属性的种类：   传播行为、隔离级别、只读和事务超时**

**a)**   **传播行为定义了被调用方法的事务边界。**

|  |  |
| --- | --- |
| **传播行为** | **意义** |
| **PROPERGATION\_MANDATORY** | **表示方法必须运行在一个事务中，如果当前事务不存在，就抛出异常** |
| **PROPAGATION\_NESTED** | **表示如果当前事务存在，则方法应该运行在一个嵌套事务中。否则，它看起来和PROPAGATION\_REQUIRED**  **看起来没什么俩样** |
| **PROPAGATION\_NEVER** | **表示方法不能运行在一个事务中，否则抛出异常** |
| **PROPAGATION\_NOT\_SUPPORTED** | **表示方法不能运行在一个事务中，如果当前存在一个事务，则该方法将被挂起** |
| **PROPAGATION\_REQUIRED** | **表示当前方法必须运行在一个事务中，如果当前存在一个事务，那么该方法运行在这个事务中，否则，将创建一个新的事务** |
| **PROPAGATION\_REQUIRES\_NEW** | **表示当前方法必须运行在自己的事务中，如果当前存在一个事务，那么这个事务将在该方法运行期间被挂起** |
| **PROPAGATION\_SUPPORTS** | **表示当前方法不需要运行在一个是事务中，但如果有一个事务已经存在，该方法也可以运行在这个事务中** |

**b)**   **隔离级别**

**在操作数据时可能带来 3 个副作用，分别是脏读、不可重复读、幻读。为了避免这 3 中副作用的发生，在标准的 SQL 语句中定义了 4 种隔离级别，分别是未提交读、已提交读、可重复读、可序列化。而在 spring 事务中提供了 5 种隔离级别来对应在 SQL 中定义的 4 种隔离级别，如下：**

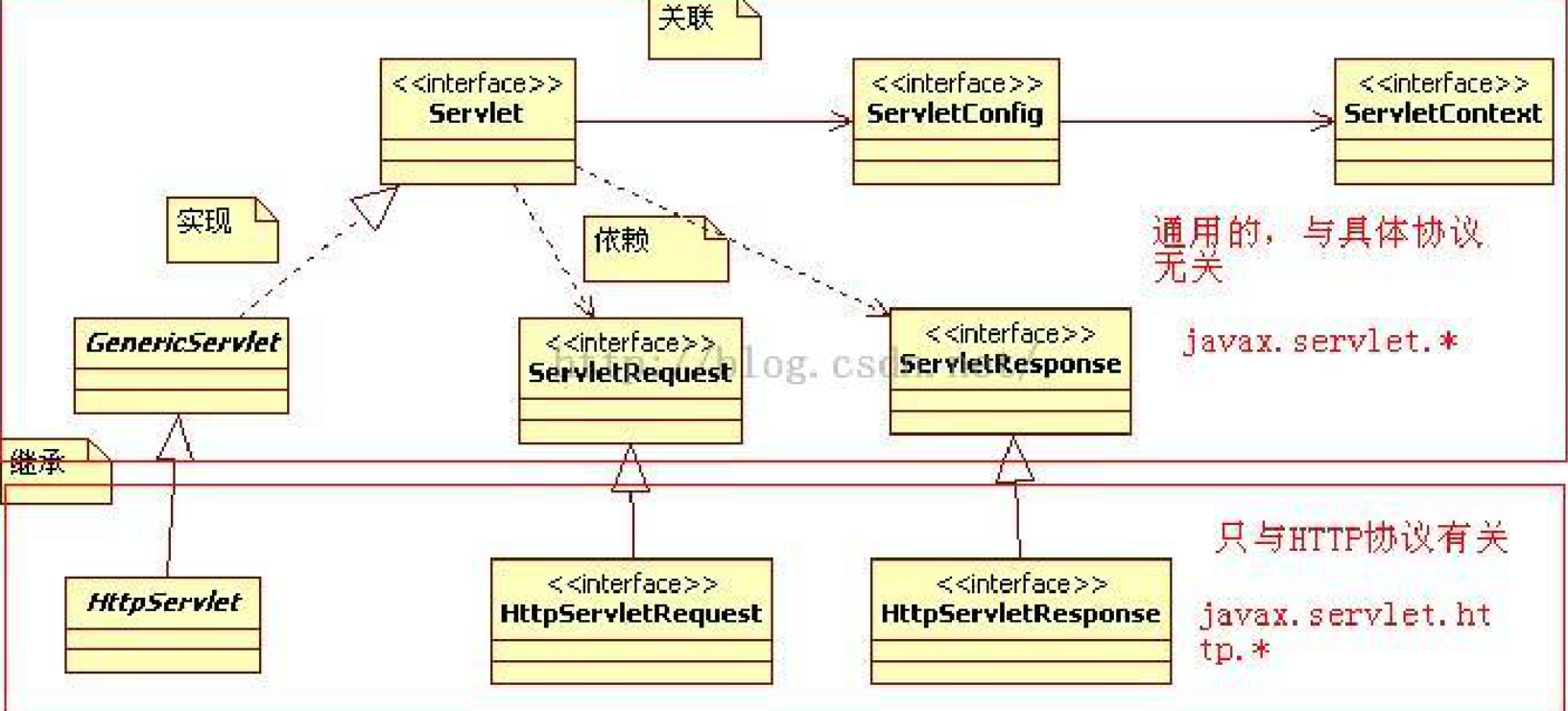
|  |  |
| --- | --- |
| **隔离级别** | **意义** |
| **ISOLATION\_DEFAULT** | **使用后端数据库默认的隔离级别** |
| **ISOLATION\_READ\_UNCOMMITTED** | **允许读取未提交的数据（对应未提交读），可能导致脏读、不可重复读、幻读** |
| **ISOLATION\_READ\_COMMITTED** | **允许在一个事务中读取另一个已经提交的事务中的数据（对应已提交读）。可以避免脏读，**  **但是无法避免不可重复读和幻读** |
| **ISOLATION\_REPEATABLE\_READ** | **一个事务不可能更新由另一个事务修改但尚未提交（回滚）的数据（对应可重复读）。**  **可以避免脏读和不可重复读，但无法避免幻读** |
| **ISOLATION\_SERIALIZABLE** | **这种隔离级别是所有的事务都在一个执行队列中，依次顺序执行，而不是并行（对应可序列化）。**  **可以避免脏读、不可重复读、幻读。但是这种隔离级别效率很低，因此，除非必须，**  **否则不建议使用。** |

**c)**    **只读**

**如果在一个事务中所有关于数据库的操作都是只读的，也就是说，这些操作只读取数据库中的数据，而并不更新数据，那么应将事务设为只读模式（ READ\_ONLY\_MARKER ） , 这样更有利于数据库进行优化 。**

**因为只读的优化措施是事务启动后由数据库实施的，因此，只有将那些具有可能启动新事务的传播行为 (PROPAGATION\_NESTED 、 PROPAGATION\_REQUIRED 、 PROPAGATION\_REQUIRED\_NEW) 的方法的事务标记成只读才有意义。**

**如果使用 Hibernate 作为持久化机制，那么将事务标记为只读后，会将 Hibernate 的 flush 模式设置为 FULSH\_NEVER, 以告诉 Hibernate 避免和数据库之间进行不必要的同步，并将所有更新延迟到事务结束。**

**d)**   **事务超时**

**如果一个事务长时间运行，这时为了尽量避免浪费系统资源，应为这个事务设置一个有效时间，使其等待数秒后自动回滚。与设**

**置“只读”属性一样，事务有效属性也需要给那些具有可能启动新事物的传播行为的方法的事务标记成只读才有意义**

Servlet的生命周期

1.加载：容器通过类加载器使用Servlet类对应的文件来加载Servlet

2.创建：通过**调用Servlet的构造函数来创建一个Servlet实例**

3.初始化：通过调用Servlet的init()方法来完成初始化工作，**这个方法是在Servlet已经被创建，但在向客户端提供服务之前调用。**

4.处理客户请求：Servlet创建后就可以处理请求，当有新的客户端请求时，Web容器都会**创建一个新的线程**来处理该请求。接着调用Servlet的

Service()方法来响应客户端请求（Service方\*\*\*根据请求的method属性来调用doGet（）和doPost（））

5.卸载：**容器在卸载Servlet之前**需要调用destroy()方法，让Servlet释放其占用的资源。

如果在类中定义成员变量，而在service中根据不同的线程对该成员变量进行更改，那么在并发的时候就会引起错误。最好是在方法中，定义局部变量，而不是类变量或者对象的成员变量。由于方法中的局部变量是在栈中，彼此各自都拥有独立的运行空间而不会互相干扰，因此才做到线程安全。

AWT ：是通过调用操作系统的native方法实现的，所以在Windows系统上的AWT窗口就是Windows的风格，而在Unix系统上的则是XWindow风格。 [AWT](https://www.baidu.com/s?wd=AWT&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YYm1n1Pjm1mHfsPW03mWcd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1T3nW6krjTL) 中的图形函数与 [操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YYm1n1Pjm1mHfsPW03mWcd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1T3nW6krjTL) 所提供的图形函数之间有着一一对应的关系，我们把它称为peers。 也就是说，当我们利用 [AWT](https://www.baidu.com/s?wd=AWT&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YYm1n1Pjm1mHfsPW03mWcd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1T3nW6krjTL) 来构件图形用户界面的时候，我们实际上是在利用 [操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YYm1n1Pjm1mHfsPW03mWcd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1T3nW6krjTL) 所提供的图形库。由于不同 [操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YYm1n1Pjm1mHfsPW03mWcd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1T3nW6krjTL) 的图形库所提供的功能是不一样的，在一个平台上存在的功能在另外一个平台上则可能不存在。为了实现Java语言所宣称的"一次编译，到处运行"的概念，AWT 不得不通过牺牲功能来实现其平台无关性，也就是说，AWT 所提供的图形功能是各种通用型操作系统所提供的图形功能的交集。由于AWT 是依靠本地方法来实现其功能的，我们通常把AWT控件称为重量级控件。

Swing ：是所谓的Lightweight组件，不是通过native方法来实现的，所以Swing的窗口风格更多样化。但是,Swing里面也有heaveyweight组件。比如JWindow，Dialog,JFrame

Swing是所谓的Lightweight组件，不是通过native方法来实现的，所以Swing的窗口风格更多样化。但是,Swing里面也有heaveyweight组件。比如JWindow，Dialog,JFrame

Swing由纯Java写成，可移植性好，外观在不同平台上相同。所以Swing部件称为轻量级组件（ Swing是由纯JAVA CODE所写的，因此SWING解决了JAVA因窗口类而无法跨平台的问题，使窗口功能也具有跨平台与延展性的特性，而且SWING不需占有太多系统资源，因此称为轻量级组件！！！）

1. Java只支持单继承，实现多重继承三种方式：（1）直接实现多个接口 （2）扩展(extends)一个类然后实现一个或多个接口  （3）通过内部类去继承其他类

2、java的4类流程控制语句

循环语句：while，for，do while

选择语句（分支语句）：if,switch

跳转语句：break,continue,break,label

异常处理语句：try catch finally，throw

Java表达式转型规则**由低到高转换**：

1、所有的byte,short,char型的值将被提升为int型；

2、如果有一个操作数是long型，计算结果是long型；

3、如果有一个操作数是float型，计算结果是float型；

4、如果有一个操作数是double型，计算结果是double型；

5、被fianl修饰的变量不会自动改变类型，当2个final修饰相操作时，结果会根据左边变量的类型而转化。