Connection Pooling Gelişmiş Ayarlar

Temel Kavramlar

Idle Connection: Connection havuzundayken, bir uygulama tarafından kullanıldıktan sonra tekrar havuza dönüp yeniden kullanılmayı bekleyen connection'dur.

Stale Connection: Havuzda bulunan veya borrowed alınmış connection'un, kullanılamaz halde olmasıdır. Bunun sebebi bağlantının remotely olarak kapatılması, veritabanının yeniden başlatılması vs gibi durumlardır.

Abandoned Connection: Uygulama **uzun süre** boyunca connection'u havuza göndermezse bu connection abandoned olur. Bunun nedeni, uygulama **close()** metodu ile connection'u kapatmadığında veya uygulama **uzun süren** sql işlemleri yaptığında meydana gelir.

Inactive Connection Timeout / idle-timeout-minutes: Bir connection'un kapatılıp havuzdan atılmadan önce ne kadar süre **idle** olarak kalacağını belirler.

Connection Wait Timeout: Uygulamanın, connection havuzunda connection kalmadığında, connection'u havuzdan request ettiği zaman beklemesi gereken süreyi temsil eder. Genelde bu değer sonsuz yapılır.

Ayarlar

Not: Aşağıda belirtilen özellikler **Apache DBCP** (http://commons.apache.org/proper/commons-dbcp/) connection havuzunun özellikleridir. Apache Tomcat, connection havuzu olarak Apache **DBCP** 'yi bazı özelliklerini değiştirerek kullanmaktadır. Bundan dolayı Apache Tomcat server kullanıyorsanız yukarıdaki ayarlar aynı isimde kullanılır.

Temel Ayarlar

maxActive: Aynı anda kaç tane aktif connection'unun havuzda tutulacağını belirler. **Default** değeri 100'dür.

maxIdle: Eğer **maxActive 100** olarak tanımlanmışsa, **maxIdle** değeri **80** ise, veritabanına herhangi bir request yoksa ve **validationQuery** set edilmişse, sadece **80** connection test edilecektir ve aktif kalacaktır. Geri kalan **20** connection kapatılacaktır. Fakat **maxActive 100** yaptık ve **100** connection'da sürekli kullanılıyorsa o zaman **maxIdle** değerini set etmenin bir faydası yoktur. Kısacası, bu değer connection havuzunda her an/her zaman(at all times) ne kadar connection'un var olacağını belirler. Idle connection'lar, aktif edilmişse, periyodik bir şekilde taranır ve bu connection'lar **minEvictableIdleTimeMillis** değerinden daha uzun süredir idle durumda iseler, kapatılırlar.

Not: maxIdle değeri **default** olarak **maxActive** değerine eşit olur.

minIdle: Connection havuzunda her a/her zaman(at all times) minimum ne kadar connection tutulacağını belirler. Eğer validation query'iler başarısız olursa, connection havuzundaki connection sayısı bu değerden daha aza düşebilir. **Default** değer **initialSize:10** değerine eşittir.

initialSize: Connection havuzu başlatıldığı zaman, minimum ne kadar connection yaratılacağını belirler. **Default** değeri 10'dur.

maxWait: Milisaniye cinsinden değer alır. Connection havuzunun, kullanıcı tarafından kullanılan bir connection'u ne kadar süre bekleyeceğini ifade eder. **Default** değeri **30000** milisaniyedir. Yani **30** saniyedir.

Not: maxWait değerini bir sql query işlemi minimum kaç saniye sürüyorsa o değere eşitlemek mantıklı olacaktır. Aksi taktirde query işlemi yarıda kalır.

Gelişmiş Ayarlar

1. Validate Connections:

testOnBorrow: true değerine sahip olduğu zaman, connection havuzundan bir connection borç(borrow) alındığında, yani bağlantı request edilince, validation işlemine tabi tutulacaktır.

testOnReturn: true olduğu zaman, uygulama tarafından borrowed/request edilen connection havuza dönmeden önce **validation query** run edilir. Varsayılan(default) değeri **false** olarak belirtilmiştir. Genelde bu özellik kullanılmaz.

testWhileIdle: true olduğu zaman, **idle** connection havuzdan request edilince **validation query** çalışır.

validationInterval: Validation işlemi aktif hale getirildiği zaman, bir connection borrow edildiği durumda **validation query** run edilir. Bunun sonucunda

aşağıda belirtildiği gibi performans kaybı oluşur. Bunu önlemek için sadece belirli aralıklarla validation işleminin yapılması mantıklı olacaktır. Bu özelliğin aldığı değer milisaniye cinsindendir. Örneğin **30000** değerine sahip olduğunda **30** saniyede bir validation yapılacaktır.

validationQuery: validation işlemi bu elementin aldığı değere göre yapılır. Eğer bu değer set edilmezse **validation yapılmaz.**
Örneğin Mysql veritabanına bağlanmışsak, **validationQuery** = **"SELECT 1"** şeklinde kullanmalıyız. Diğer veritabanları için aldığı değerler aşağıdaki gibidir:

- **hsqldb** - `select 1 from INFORMATION_SCHEMA.SYSTEM_USERS`
- **Oracle** - `select 1 from dual`
- **DB2** - `select 1 from sysibm.sysdummy1`
- **mysql** - `select 1`
- **microsoft SQL Server** - `select 1` (tested on SQL-Server 9.0, 10.5 [2008])
- **postgresql** - `select 1`
- **ingres** - `select 1`
- **derby** - `values 1`
- **H2** - `select 1`
- **Firebird** - `select 1 from rdb$database`

`Yukarıdaki özellikler şu tarz durumlar olduğunda aktif hale getirilir:`

a. Firewall gibi programlar tarafından veritabanı ile uygulamamız arasındaki connection işlemleri kontrol ediliyorsa, firewall uzun süre açık kalan bağlantıları kapattığı zaman oluşabilecek connection problemlerini gidermek için kullanılır.
b. Veritabanı **bakım** için, **backup** alırken vs kapatıldığında connection pool bu kapatılma işleminden **haberdar olmaz.** Böyle durumda tüm connection'lar kullanılamaz hale gelir.
c. Uzun süre açık kalan bağlantılar veritabanı tarafından kapatıldığında yeniden bağlantı açılması için kullanılır.

Not: validationQuery set edilmezse **testOnBorrow true** olsa bile ihmal edilir. Çünkü validation işlemi sql cümlesine göre yapılır. Ayrıca sql cümlesi **SELECT** cümlesi olmak zorundadır.

Not: Yukarıda belirtilen durumlar gerçekleştiğinde yani borrow aldığımız connection veritabanına bağlanamazsa, validation işlemi için kullandığımız query başarısız olur. Böyle bir durumda connection havuzdan atılıp, yerine yenisi yaratılır ve havuza eklenir.

Not: validationInterval değeri belirlenirken şuna dikkat etmeliyiz: Büyük değerler daha fazla performans sağlarken, daha küçük değerler **stale connection** (kullanılamaz halde olan connection) ihtimalini azaltır. Örneğin, **60** saniye olarak belirtildiği zaman her **60** saniye de bir **validation query** çalışacaktır. Ancak, connection **60.** saniyede validate edildikten sonra **60** saniye boyunca validate edilmeyeceği için, bu süre zarfında veritabanı **kapanırsa** veya **restart** olursa, bu süre içerisinde uygulama tarafından connection request/borrow edildiğinde, **connection hatası** meydana gelecektir. Bundan dolayı bu özelliği set ederken bu durumları da göz önünde bulundurup en uygun değeri belirlemek gereklidir.

Not: validationInterval değeri set edilmezse, her connection borrowed olduğunda **validation query** çalışır. Bu ise performans kaybına neden olacaktır. Örneğin, connection havuzundan saniyede **10** tane connection request ediliyorsa **30** saniye içerisinde **300** kez **validation query** run edilir. Bunun yerine **validationInterval** değeri **30** saniye yapılırsa request/borrowed edilen connection sayısı **10** olduğu için sadece **10** kez run edilir. Sonuç olarak fazladan **290** kez validation query run **edilmemiş** olur.

2. Connection Leaks:

Connection leak'leri önlemek için aşağıdaki özellikleri kullanmak gereklidir:

removeAbandoned: **true** değerine sahip olduğu zaman, **abandoned** onnection havuzdan atılır. Bir connection **removeAbandonedTimeout** değerinden daha uzun süredir kullanılıyorsa, örneğin uzun süren bir sql cümlesi buna neden olabilir, bu connection havuzdan atılır.

removeAbandonedTimeout: Saniye türünden değer alır. Örneğin **54** değerine sahip olursa **54** saniye içerisinde connection havuzdan atılabilir anlamına gelir. Bundan dolayı **en uzun sql** işlem süresine göre bu değer belirlenmelidir. Aksi taktirde, sql işlemi tamamlanamadan bağlantı kapatılıp havuzdan atılacağı için işlem yarıda kalabilir.

validationQuery: connection leak olmaması için bu özellik de kullanılmalıdır. Önemi yukarıda bahsedildi.

3. Validation/Cleaner Thread:

Aşağıdaki üç şart gerçekleştiği zaman **Pool Sweeper** şeklinde adlandırılan bir **thread** aktif hale gelir:

1. minEvictableIdleTimeMillis > 0
2. removeAbandoned = **true**
3. removeAbandonedTimeout > 0

Bu thread, arka planda çalışan, **idle** connection'ları **test** eden ve connection havuzunu **resize** eden bir thread'tir. Ayrıca connection leak'leri **detect** etmekten de sorumludur. Bu thread aktif hale geldikten sonra, borrowed/request edilen bir connection **minEvictableIdleTimeMillis** değerinden daha uzun süredir **idle** ise, bu connection havuzdan atılması için işaretlenir.

4. Diğer

defaultTransactionIsolation: Bu özellikle ilgili detaylı bilgi için lütfen **tıklayınız** (http://www.softwarevol.com/en/tutorial/Transaction-Isolation-Levels)