第119回 夜な夜な! なにわオラクル塾

ORACLE

WebLogic Serverを基礎から学ぶシリーズ第2弾

解説! JDBCデータソースと接続プール

WebLogic Server勉強会へようこそ

2014年2月12日
日本オラクル株式会社

ORACLE
FUSION MIDDLEWARE

なにわ
オラクル
倶楽部

「WebLogic Server勉強会」とは

- **By developers, for developers**
  - ユーザ企業、システムインテグレータ、ベンダーのそれぞれの立場を超えたWebLogic開発者同士の繋がり
  - WebLogic Serverアプリケーション開発者のための勉強会
- **スキルアップ**
  - 先人の成功・失敗談を共有
- **開発者間のネットワーキング**
  - Team WebLogic: 「やっぱり、WebLogic！」コミュニティ@Facebook
  - 「WebLogicつながり」の輪を広げよう！
- **最新情報**
  - Oracle Technology Network:
  - http://www.oracle.com/technetwork/jp/middleware/weblogic/community/index.html

本日のアジェンダ

18:30-18:40	オープニング
18:40-19:15	「WebLogic ServerのJDBCデータソースの基礎」 WebLogic Serverのデータベース接続サービスであるJDBCデータソースや接続プールの基本について説明します。 日本オラクル Fusion Middleware事業統括本部 ソリューション本部 野邊 哲男
19:15-19:20	休憩
19:20-20:00	「WebLogic ServerのJDBCデータソースの監視方法、設定上の注意点」 ツールを用いてWebLogic ServerのJDBCデータソースの動作状況を監視する方法について、デモを交えて説明します。また、WebLogic ServerのJDBCデータソースの設定上の注意点について説明します。 日本オラクル Fusion Middleware事業統括本部 ソリューション本部 野邊 哲男
20:00-20:30	Quiz, Q&A

次回のご案内

2014年5月14日予定

WebLogic Serverを基礎から学ぶシリーズ
第3弾！

再び、お目にかかれるのを
楽しみにしています。

ORACLE

WebLogic Serverを基礎から学ぶシリーズ第2弾

解説! JDBCデータソースと接続プール

日本オラクル株式会社
Fusion Middleware事業統括本部
2014年2月12日

ORACLE
FUSION MIDDLEWARE
WEBLOGIC SERVER

なにわ
オラクル
倶楽部

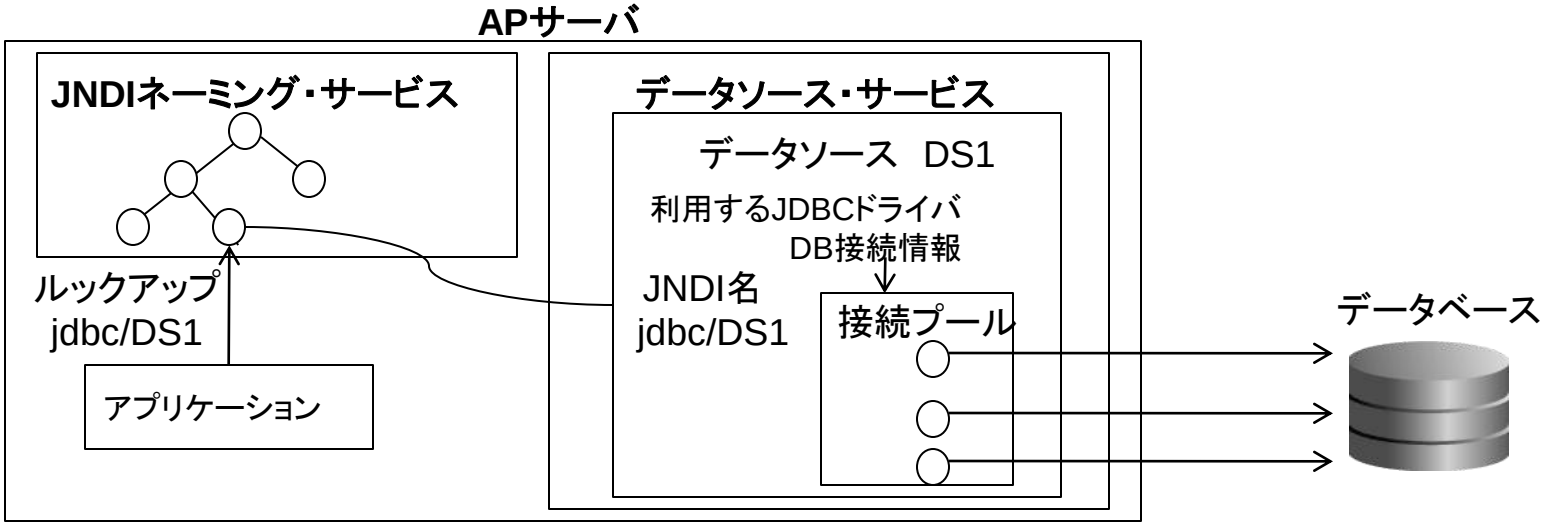
Program Agenda

- JDBCデータソースと接続プールについて
- WebLogic Server のデータソース
- WebLogic Server のデータソースの動作
- WebLogic Server のデータソースの監視
- WebLogic Server のデータソースにおける推奨/注意事項

JDBCデータソースと接続プールについて

JDBCデータソースと接続プール

- JDBCデータソースは、アプリケーション実行環境（APサーバ）において、アプリケーションにデータベース接続サービスを提供する機能。（当資料では、以後「データソース」と表記します。）
- アプリケーションは、DB接続に必要な物理的な情報（DBホスト名、DBユーザIDやパスワードなど）を意識せずにデータベース接続を行える。
- 一般的にデータソースは接続プールを提供し、アプリケーションからDB接続、切断処理のオーバーヘッドを削減。



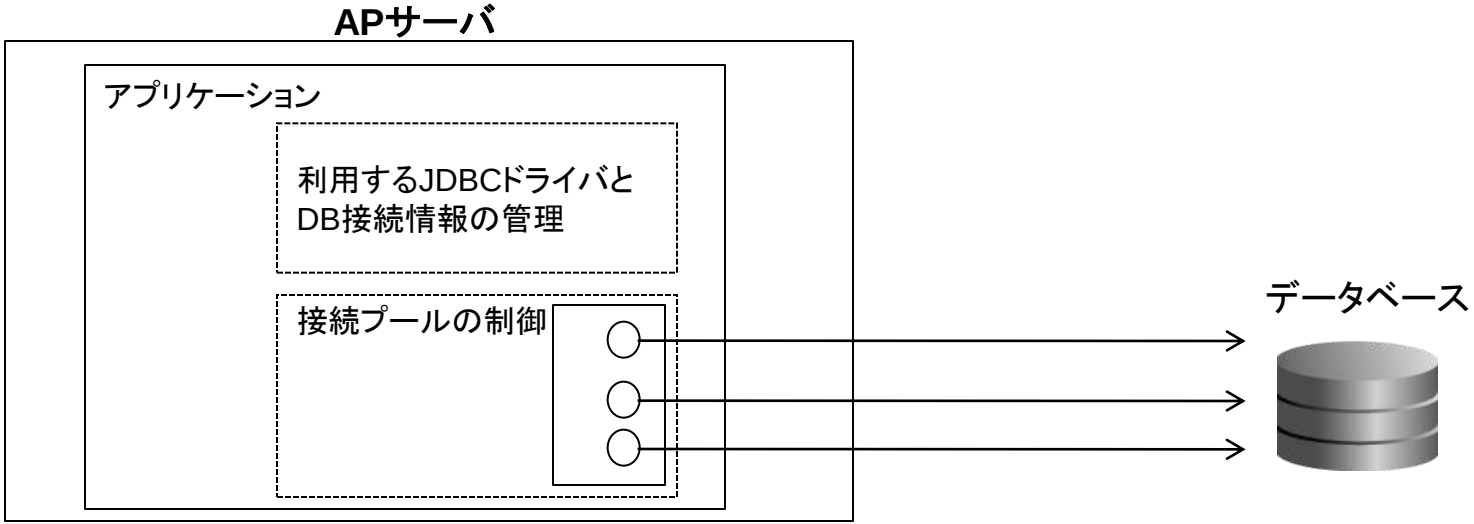
データソースを利用する場合のコード例

- 下記は、データソースを利用してConnection オブジェクトを取得する場合のコード例。

```
import javax.naming.*;
import javax.sql.*;
import java.sql.*;

...(中略）
String sql = “select * from emp”                    //実行するSQL文
Context ic = new InitialContext();                  //JNDIルックアップのための初期コンテキスト取得
DataSource ds = (DataSource)ic.lookup(“jdbc/DS1”);  //データソースオブジェクト取得
Connection conn = ds.getConnection();               //コネクション取得
Statement stmt = conn.createStatement();            //Statementの作成
ResultSet rset = stmt.executeQuery(sql);            //結果セットの取得

... ..(中略）

rset.close();  //結果セットクローズ処理
stmt.close();  //Statementクローズ処理
conn.close();  //コネクションクローズ処理
```

データソースを利用しない場合

- アプリケーションがAPサーバのデータソース・サービスを利用せず、JDBCドライバをロードし、そのAPIを使用してDB接続を行うことも不可能ではない。
- ただしその場合、DB接続情報や接続プールの管理をすべてアプリケーション側で管理・制御する必要がある。



データソースを利用しない場合のコード例

- データソースを利用しない場合、利用するJDBCドライバやDBの物理接続情報をアプリケーション側で管理する必要がある。
- 下記は、それらの情報をハードコーディングしてしまっている悪い例。

```
import java.sql.*;

...(中略）
String sql = “select empno, ename from emp”                          //実行するSQL文
DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());    // 利用ドライバの指定
String url = “jdbc:oracle:thin:@host1:1521:SID”;
Connection conn =DriverManager.getConnection(url,”SCOTT”, “TIGER”);  //コネクション取得
Statement stmt = conn.createStatement();                                  //Statementの作成
ResultSet rset = stmt.executeQuery(sql);                                  //結果セットの取得

... ..(中略）

rset.close();  //結果セットクローズ処理
stmt.close();  //Statementクローズ処理
conn.close();  //コネクションクローズ処理
```

WebLogic Serverのデータソース

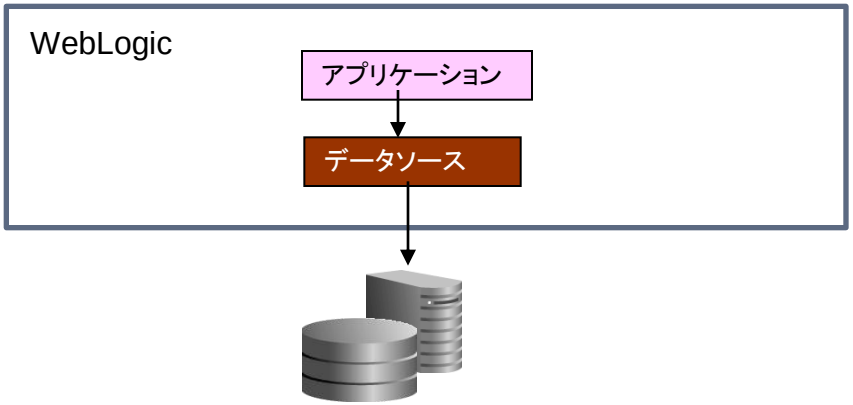
WebLogic Serverのデータソースの種類

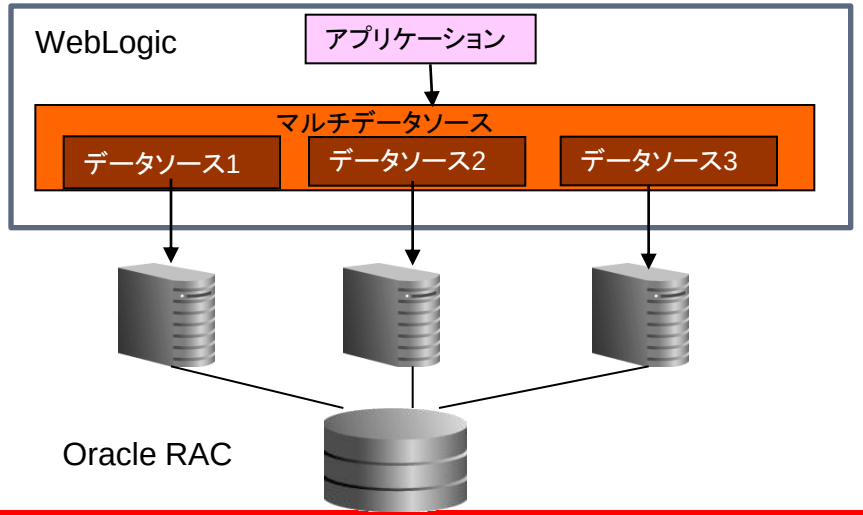
- WebLogic Server の11g (10.3.4)以降のバージョンでは、大別して3種類のデータソース機能を提供。
- 単体のDB接続には、「汎用データソース」を、Oracle RACには「マルチデータソース」か「GridLinkデータソース」を利用。

種類	用途	概要
汎用データソース （または単に、 「データソース」ともいう）	DB単体用	単体のDBインスタンスに特定のDBユーザで接続する。
マルチデータソース （当資料では詳細説明は割愛しています。）	複数のDBインスタンス用 （主にOracle RACで利用）	複数のデータソースをまとめて１つのデータソースとして利用することで接続分散と可用性向上を可能に。
Grid Link データソース <WebLogic Suiteで利用可能。ただしExalogic上であれば全エディション利用可能。当資料では詳細説明は割愛しています。>	Oracle RAC用	Oracle RAC側から死活情報、負荷情報を取得し、 アプリからの論理接続時に、それらの情報を考慮した上で最適なConnectionを利用させる。

[参考]汎用データソースとマルチデータソース

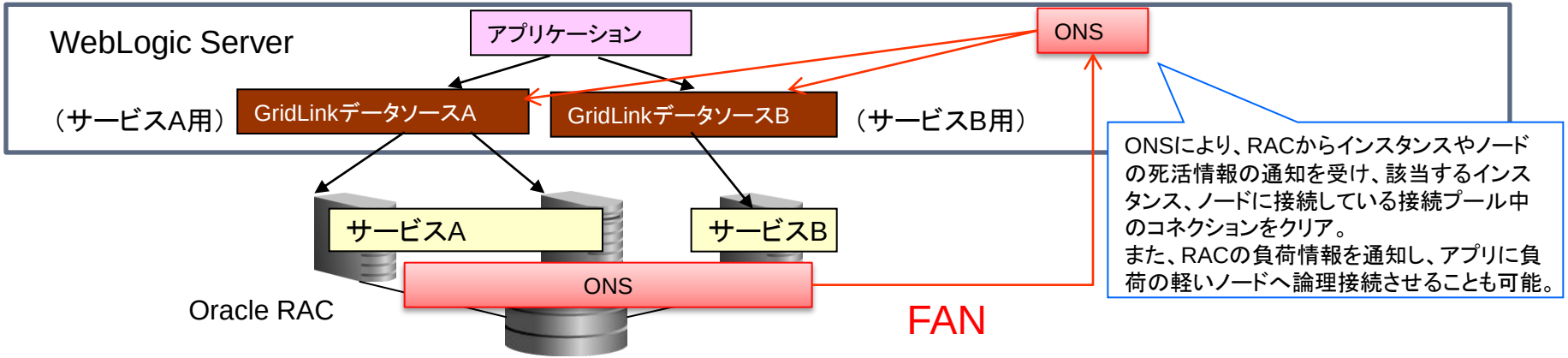
- 汎用データソースは単体のDBインスタンスに特定ユーザで接続
- マルチデータソースは複数のデータソースをまとめて1つのデータソースとする。
  - これにより、アプリからの接続要求を分散したり、1つのデータソースの接続先DBインスタンスに障害が発生した場合に、そのデータソースをアプリに利用させないように自動制御が可能。
  - つまり、Oracle RACのようなDBクラスタ環境での適用を前提としている。





[参考] GridLinkデータソース (Active GridLink for RACを行うためのデータソース)

- RACやOracle Netの機能と連携したGridLinkデータソース機能を提供。
- GridLinkデータソースでは、RACサービス単位で構成させる。
- アプリケーションはRACサービスに対応したGridLinkデータソースを指定して接続する。
- 接続要求はRACサービス設定基づき分散される。
- Oracle Notification Service(ONS)により、RACの死活情報の通知を受け、接続プール中のコネクションで、ダウンしているRACノードへのコネクションだけをクリアできる。(Fast Connection Failover)さらにRACの負荷情報を通知し、アプリが接続要求時、負荷の軽いノードへの接続を使用させることも可能。(Runtime Connection Load Balancing)



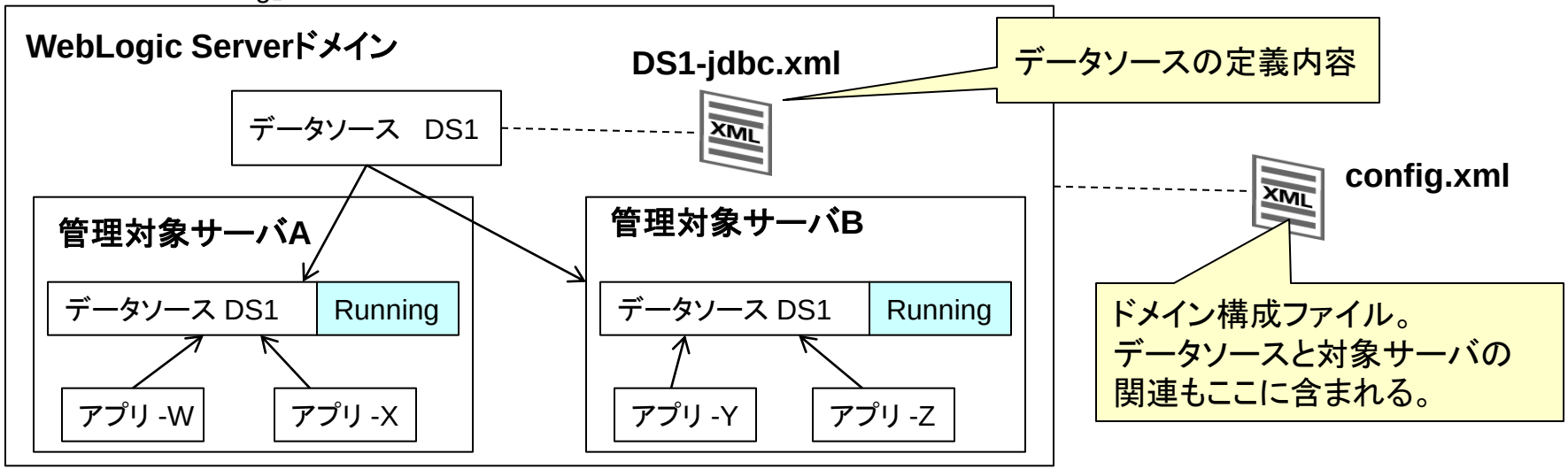
WebLogic Serverのデータソースの構成や監視について

- WebLogic Serverのデータソースは、主に管理コンソールやWLSTを用いて構成や監視を行う。
- 個々のデータソースや関連リソースに対応するJMX MBean(Configuration MBean)により構成、監視が可能。
- また、WebLogic Serverのデータソースは JSR-77(Java EE Management Model)をサポートしており、実行時の情報をMBean(Runtime MBean)を用いてモニタリングすることも可能。
- なお、WebLogic Server12c(12.1.2)よりFusion Middleware Controlを用いてデータソースの構成が可能。ただしFusion Middleware Controlを使用するために別途 Infrastructureのインストールやドメインの拡張等が必要になる。

ツール	作成・削除	変更	監視
管理コンソール	○	○	○
WLST	○	○	○
Fusion Middleware Control 12c 12.1.2 New	○	○	○
JMX　APIや任意のJMX MBeanツール		○	○

データソースの構成内容について

- WebLogic Serverにてデータソースを作成すると…….
  - DOMAIN/config/jdbc配下にデータソース毎の定義ファイル（データソース名-jdbc.xml）が生成される。
  - JDBCデータソースの対象サーバの情報はドメイン構成ファイル(config.xml)で保持される。
  - データソースを削除すると、上記XMLやconfig.xmlのエントリも削除される。
  - 対象サーバを指定してデータソースは作成すると、初期化され状態が「Running」になる。
  - 状態が「Running」になると、対象サーバ上のすべてのアプリケーションから利用可能になる。



WebLogic Serverに含まれるJDBCドライバ

- WebLogic Server12c(12.1.2)では、標準で下記のJDBCドライバを提供。
- ドライバの実体は、WL_HOME/oracle_common/modules 以下の該当ディレクトリに格納されている。
- 下表以外のJDBCドライバを利用する場合は、WebLogicに適用するCLASSPATHの先頭に追加するドライバのライブラリを追加する。
- **Oracle Database12cと共に提供されるOracle Thin Driver12c(ojdbc7.jar)は、WebLogic Server12c(12.1.2)には標準で含まれていないが、上記の設定で利用する事は可能。**

ドライバ	ファイル名	備考
Oracle Thin Driver 11g	ojdbc6.jar , ojdbc6_g.jar , ojdbc6dms.jar	11.2ベース、TYPE4
MySQL5.1 JDBCドライバ	mysql-connector-java-commercial-5.1.22-bin.jar	
WebLogic Type4 JDBCドライバ	DB2用：wldb2.jar MS SQL Server用：wlsqlserver.jar Informix用：wlinformix.jar Sybase用：wlsybase.jar	DataDirectのOEM供給

その他、WebLogic Serverに含まれるサンプル用にDerbyのDBMSとJDBCドライバが WL_HOME/common/derbyに格納される。

データソース作成時に必要なパラメータ

- 下表はデータソースに対する設定項目。接続プールに対する詳細設定は、データソース作成後に行う。

項目（＊は必須項目）	概要	Oracle Databaseの場合の例
データソース名＊	WebLogicでの管理名	DS1
JNDI名＊	JNDIツリーへのバインド名	jdbc/DS1
データベースのタイプ＊	OracleやDB2など　※1	Oracle
データベース・ドライバ＊	データベース種類に応じて指定 ※1	Oracle Thin Instance-Connection
トランザクション・オプション	非XAドライバの場合に指定	1phase commit
データベース名＊	データベースのID	ORCL
ホスト名＊	データベースのホスト名	localhost
ポート＊	データベースへの接続ポート	1521
データベース・ユーザ名	データベースユーザ名	SCOTT
パスワード	データベースユーザのパスワード	TIGER
追加の接続プロパティ:oracle.jdbc.DRCPConnectionClass	DB12cのDRCP利用時に必要に応じて値を設定	CRM_APP
ドライバ・クラス名＊	JDBCドライバのクラス名※2	oracle.jdbc.OracleDriver
URL＊	JDBC URL※2	jdbc:oracle:thin:@localhost:1521:ORCL
プロパティ	接続の作成時にJDBCドライバに渡すプロパティ	
システム・プロパティ	ドライバ・プロパティのセット。値は指定されたシステム・プロパティから実行時に導出	
テスト対象の表名（or SQL)	接続テストに利用する表またはSQL	SQL SELECT 1 FROM DUAL
ターゲット	WLSドメイン中、どのサーバで利用するか	Server1

※1 管理コンソールの場合リストBOXで選択
※2 管理コンソールの場合、自動入力

接続プールの主要パラメータ

- 下表の主要パラメータは、WLS11g(10.3.6)、WLS12c(12.1.1)、WLS12c(12.1.2)で共通。

項目	概要	デフォルト値
初期容量	接続プール作成時に作成する物理接続の数。この数の接続を作成できない場合、データ・ソースの作成は失敗する	1
最大容量	接続プール中に作成可能な最大接続数	15
最小容量	初期化後にこの接続プールに含めることのできる物理接続の最小数。互換性のため 初期容量が設定されてない場合のみ適用される。 データ・ソースが中断/再開された後は、最小容量または初期容量の大きい方の値を適用	1
予約時に接続をテスト	アプリケーションが接続要求を行った際に、接続の有効性をテストするか否か。 （テスト対象の表名の指定が必須）	false
テスト頻度（秒）	接続プール中の未使用接続に対する接続テストの実行間隔。テスト失敗時はその接続を無効化して再度接続を作成 （テスト対象の表名の指定が必須）	120
アイドルプール接続を信頼する秒数	ここで指定した時間内に正常性が確認された接続のテストはスキップする。	10
縮小頻度（秒）	接続プール内の接続数を縮小させるまでの間隔	900
非アクティブ接続タイムアウト（秒）	アプリケーションで使用中の接続が非アクティブの場合、ここで指定した秒数が経過すると接続プールに自動復旧する	0（無効）
接続予約のタイムアウト（秒）	アプリケーションが接続要求時、接続を得るまで待機できる秒数	10
接続作成の再試行間隔（秒）	接続プール中の接続作成が失敗した場合に再作成を試行する間隔	0
文タイムアウト	JDBCドライバに対して実行中のSQL文をタイムアウトする時間の指定. JDBCドライバのStatement.setQueryTimeoutのメソッド実装に依存	-1

Oracle JDBC Driverによるデータソース作成

- WebLogic Serverに含まれるOracle のJDBCドライバを利用してデータソースを作成する場合、ドライバのタイプと接続先指定方法などにより下表7パターンから1つを選択する。
- Instance Connection：オラクルのSIDを指定する方法　（例：jdbc:oracle:thin:@localhost:1521:XE)
- Service Connection：オラクルのservice_nameを指定する方法（例：jdbc:oracle:thin:@localhost:1521/XE)
- RAC Service-Instance Connection：Oracle RACにマルチデータソースで接続する際に使用
- Application Continuity: DB障害後の復旧時にトランザクションを自動的に再実行する機能の利用時に使用(12.1.2の新機能)

ドライバ種類	ドライバ・クラス	接続先指定方法	備考
XA Thin ドライバ	oracle.jdbc.xa.client.OracleXADataSource	Instance Connection	
		Service Connection	
		RAC Service-Instance Connection	
Thin ドライバ	oracle.jdbc.OracleDriver	Instance Connection	
		Service Connection	
		RAC Service-Instance Connection	
	oracle.jdbc.replay.OracleDataSourceImpl 12.1.2 New	Application Continuity	下記での利用を推奨 対象のDBが12cかつ利用する JDBC ドライバをojdbc7.jar

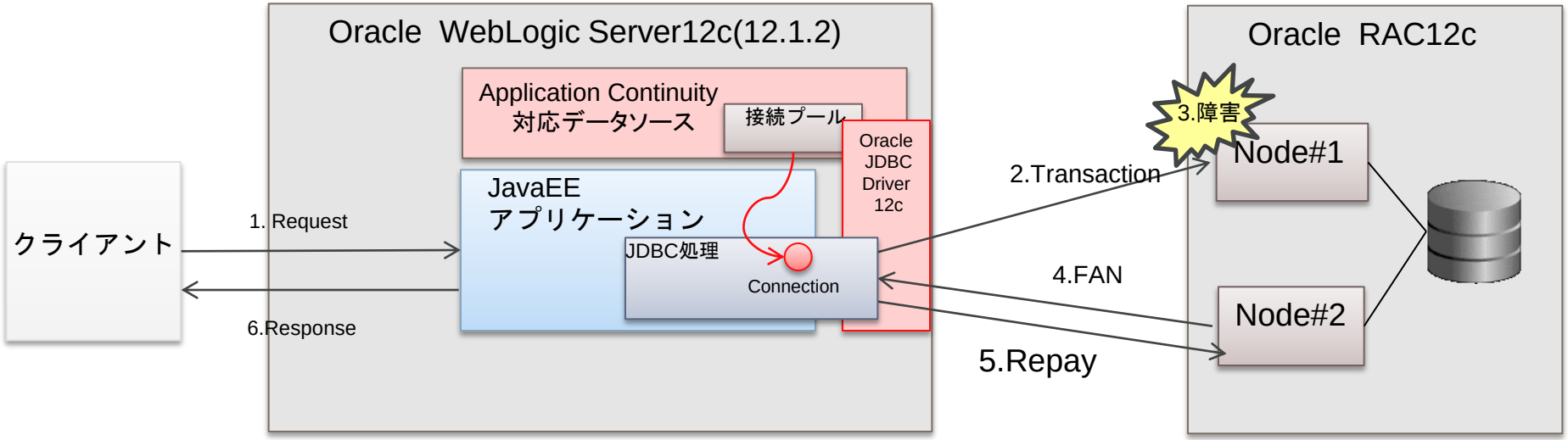
【参考】Application Continuityとは

- Application Continuityは、データベース障害時にアプリケーションのデータベース処理のフェイルオーバー(Replay)を透過的に実行する機能。
- これは、Oracle Database12cとOracle JDBC Driver12cにより提供される機能で、JDBCを使用するアプリケーションやWebLogic12(12.1.2)上で動作するアプリケーションに適用できる。
- 基本、Oracle RACとの併用が効果的だが、Active-Standbyのデータベースやシングル・データベース環境でも利用することができる。

メリット	説明
データベース処理の可用性向上	アプリケーションがデータベースに対して処理を実行している最中にデータベースに障害が発生した場合でも、その復旧後や、RACであれば残存インスタンスにフェイルオーバーして処理を継続することができます。
アプリケーションでの対応不要	上述の機能を利用するために、特にアプリケーション側で特別なコードを記述する必要はないため、既存のアプリケーションのデータベース処理の可用性を容易に向上する事ができます。

【参考】WebLogicデータソースのApplication Continuity対応

- WebLogic Server12c(12.1.2)では Application Continuityに対応したデータソースを構成できる。
- これにより、WebLogicで動作するアプリケーションがDB障害発生時に透過的にJDBC処理をフェイルオーバー(Replay)させることが可能。
- 下図ではDBがRACだが、シングルのデータベースでも利用可能。



WebLogic Serverのデータソースの動作

データソースの動作

- 下表のデータソースを前提に、WebLogic ServerのJDBCデータソースの動作について次のスライドより説明する。

データソース

項目	設定値
データソース名	DS1
JNDI名	jdbc/DS1
データベース種類	Oracle Database12c
ドライバ種類	Oracle Thin Service-Connection
ターゲット	Server1

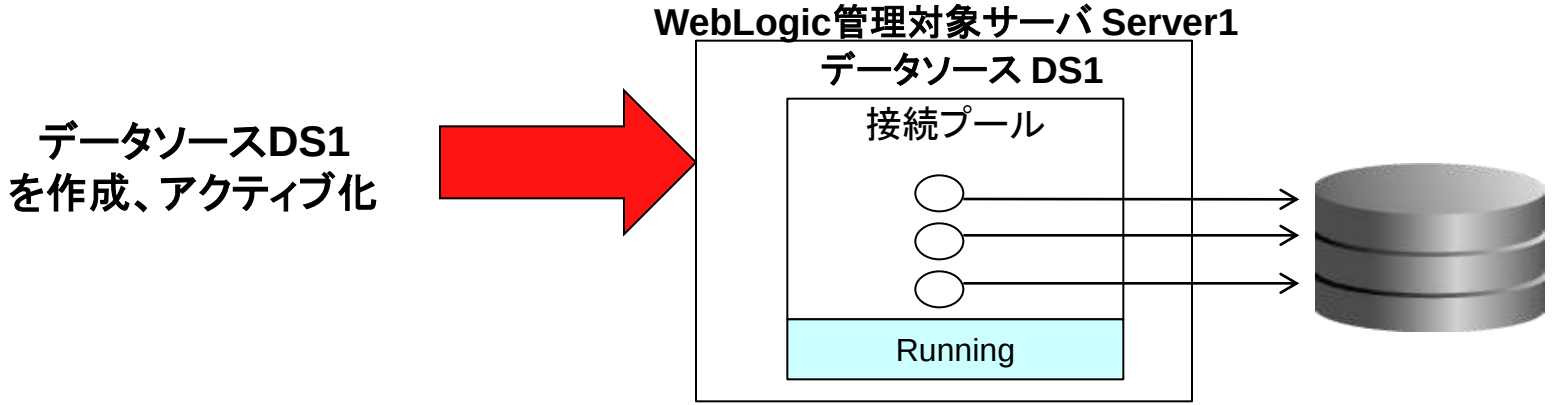
接続プール

項目	設定値
初期容量	3
最大容量	5
最小容量	3
予約時にテスト	False
テスト頻度	0
接続予約のタイムアウト（秒）	10
接続作成の再試行間隔	0
縮小頻度	900

データソース作成

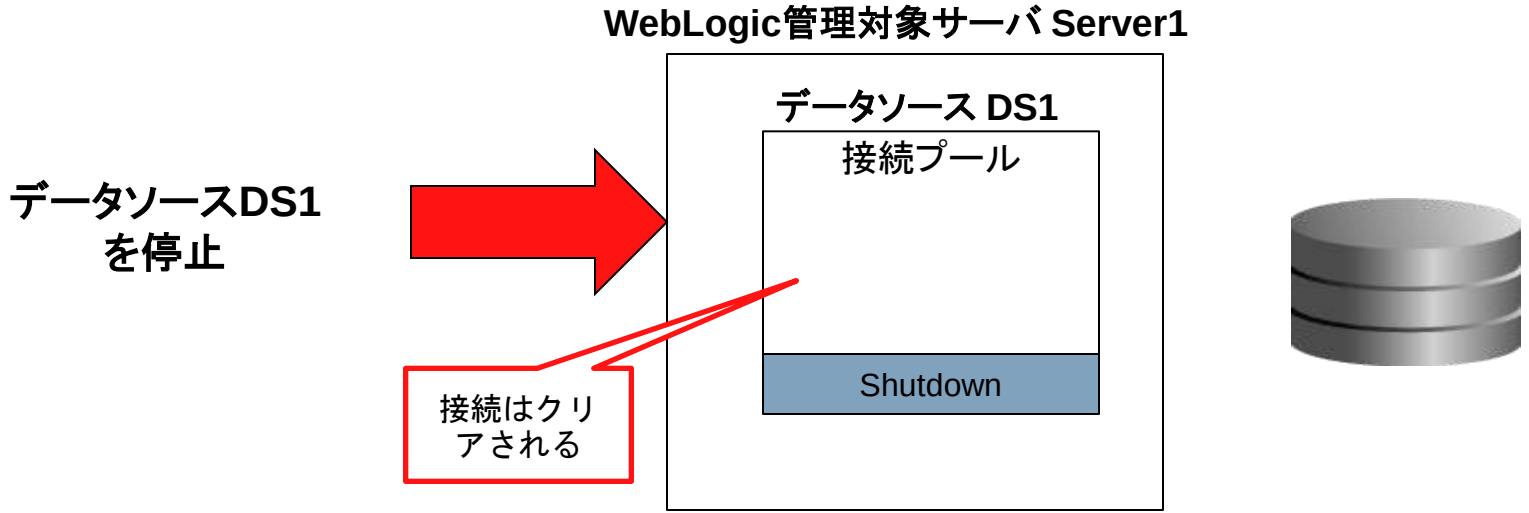
- 管理コンソール等で、「ターゲット」となるWebLogicサーバ名を指定した上でデータソースの作成、アクティブ化を行うと、データソースが初期化され、接続プール中に「**初期容量**」で指定した数の接続が作成され、すぐにアプリケーションから利用可能になる。

  (今回、「初期容量」=「最小容量」にしている。「初期容量」NOT = 0 AND 「初期容量」<「最小容量」の場合は「最小容量」で指定した数の接続が作成される。)
- 「ターゲット」のサーバ指定無しでも作成・アクティブ化だけは可能。
- 管理対象サーバの再起動は不要。
- データソース作成時には、データベースは利用可能状態である必要がある。



データソース停止

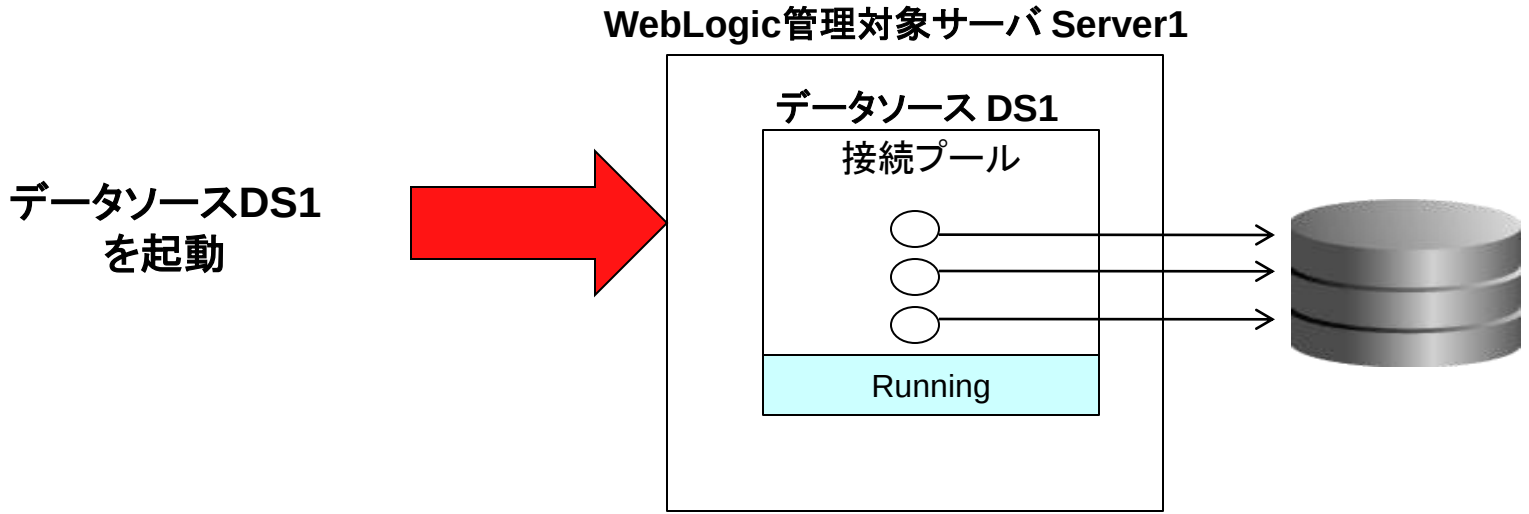
- 管理コンソール等で、データソースに対して停止操作を行うと、「状態」が「Shutdown」になり、接続プールはクリアされ、結果、データベースとの物理接続もクリアされる。
- データソースを停止中にアプリケーションが接続要求を行うと下記例外が発生する。
- weblogic.jdbc.extensions.PoolDisabledSQLException: weblogic.common.resourcepool.ResourceDisabledException: Data Source DS1 is not active, cannot allocate connections to applications



データソース起動

- データソース停止時、起動すると、接続プール中に「**初期容量**」で指定した数の接続が作成され、すぐにアプリケーションから利用可能になる。

  (今回、「初期容量」=「最小容量」にしている。「初期容量」NOT = 0 AND「初期容量」<「最小容量」の場合は「最小容量」で指定した数の接続が作成される。)

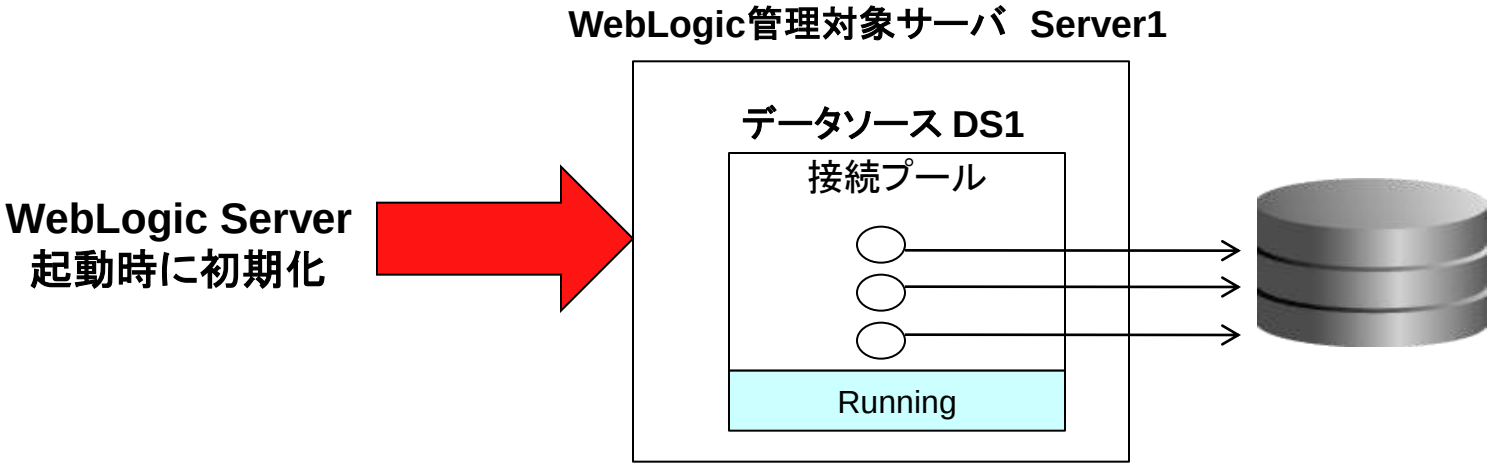


WebLogic Server起動時

- WebLogic Server起動時に、データソースが自動的に初期化され、接続プール中に「**初期容量**」で指定した数の接続が作成され、アプリケーションから利用可能になる。

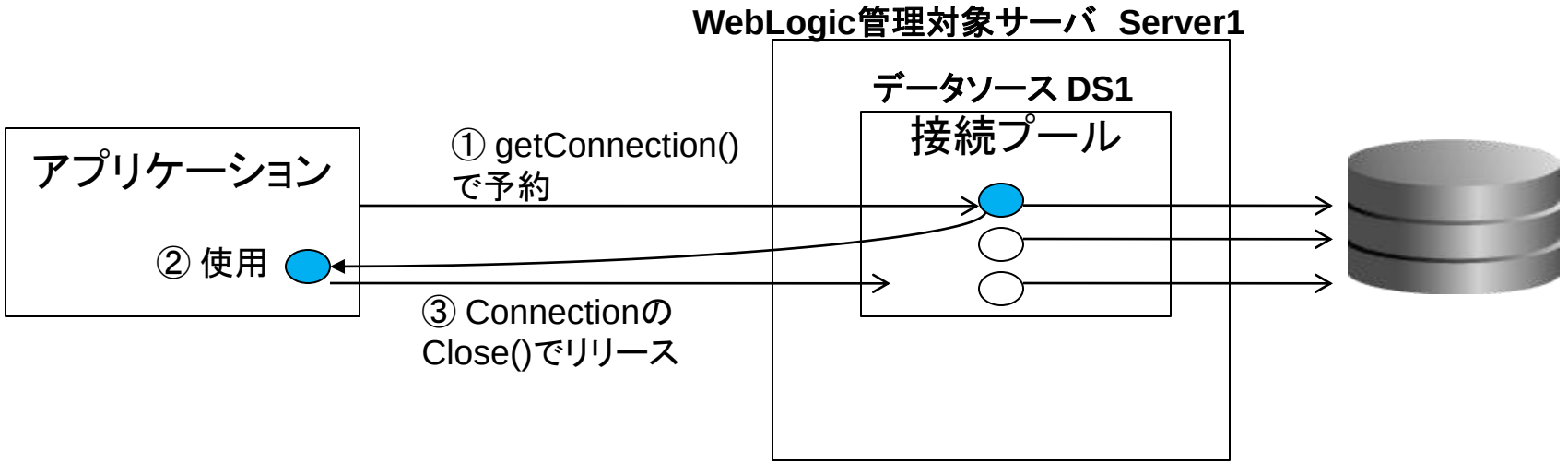
  (今回、「初期容量」＝「最小容量」にしている。「初期容量」NOT = 0 AND 「初期容量」＜「最小容量」の場合は「最小容量」で指定した数の接続が作成される。)

- WebLogic Server起動時は、原則としてデータベースは起動している必要がある。



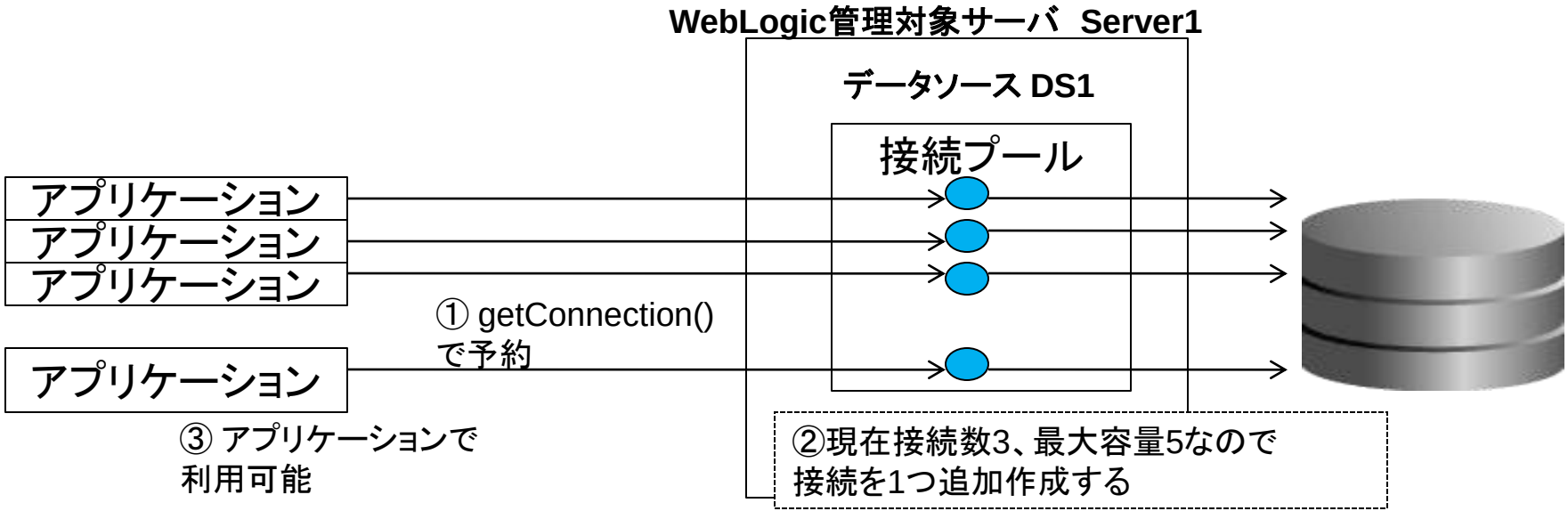
アプリケーション接続要求時

- アプリケーションがデータソースを利用して接続を要求した時、データソースの接続プールに未使用接続があれば、それを「予約」してアプリケーションが利用する。
- アプリケーションは利用後接続をリリースすると、接続プールに接続が戻される。



アプリケーション接続要求時に未使用接続が無い場合①

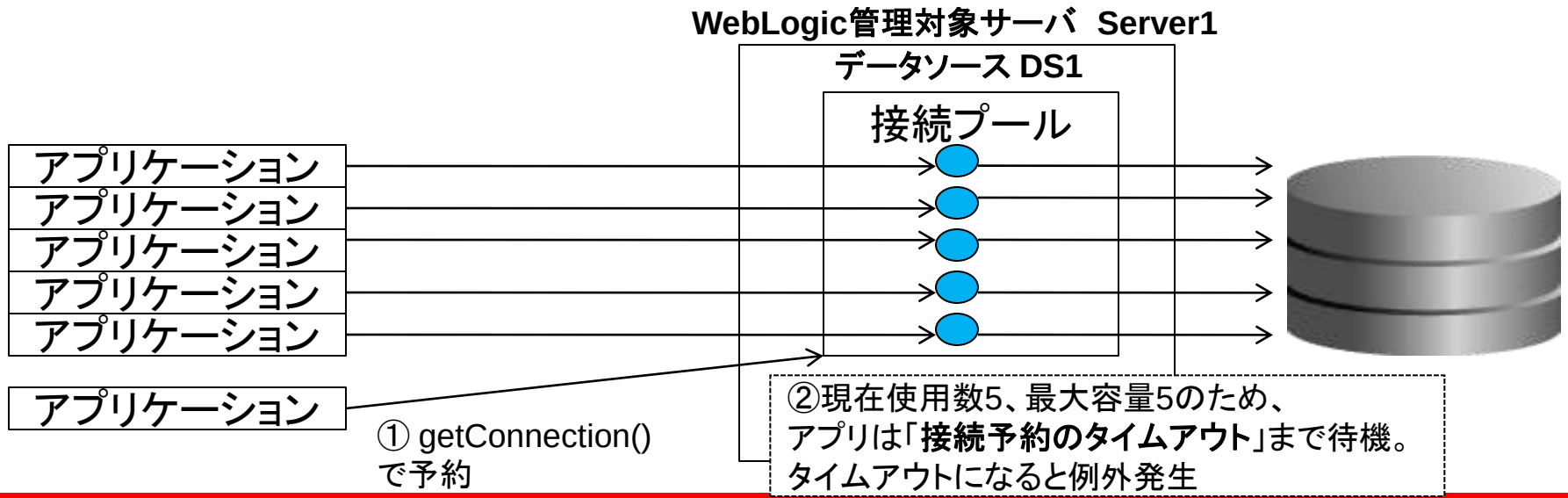
- 接続プール中の接続数が**最大容量**に達していない場合は、新たな接続が生成され、アプリケーションはそれを予約、使用することが可能。
- (アプリ接続要求後、「**接続予約のタイムアウト**」まで待たなくても、新たな接続が生成される。)



アプリケーション接続要求時に未使用接続が無い場合②

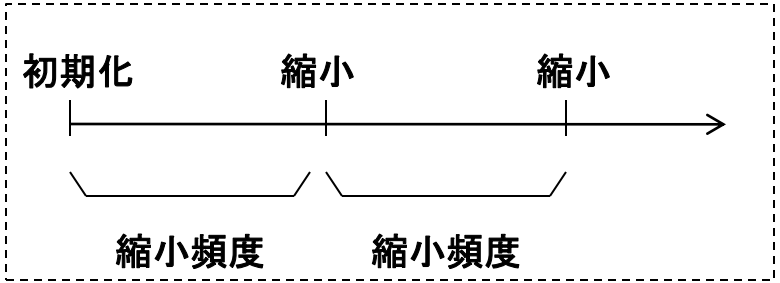
- 接続プール中の接続数が**最大容量**に達している場合は、接続プールに接続が戻るまで「**接続予約のタイムアウト**」で指定した秒数分だけ待機する。
- 「**接続予約のタイムアウト**」で待機しても接続を得られなかった場合、下記例外が発生する。
- weblogic.jdbc.extensions.PoolLimitSQLException: weblogic.common.resourcepool.ResourceLimitException:

No resources currently available in pool DS1 to allocate to applications, please increase the size of the pool and retry..

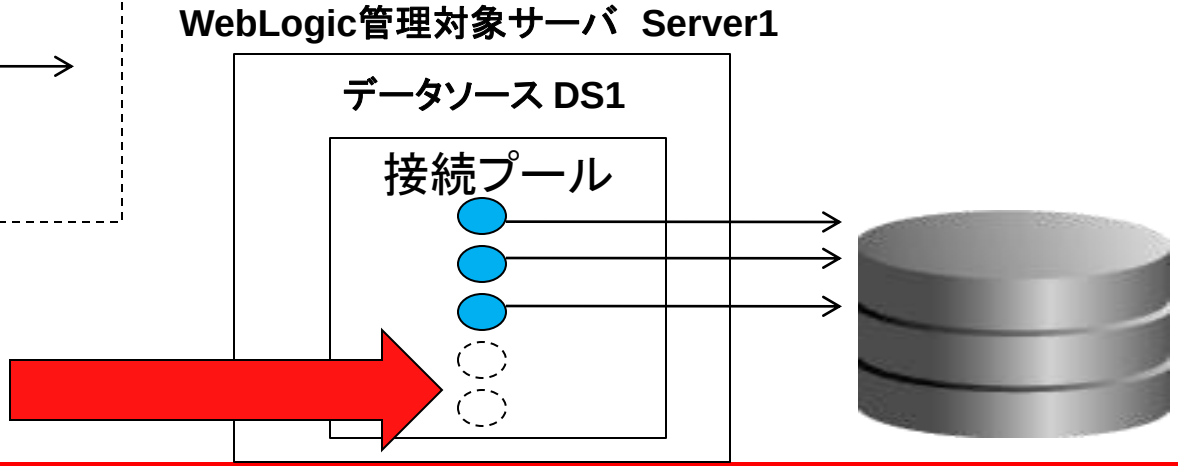


接続プールの縮小

- WebLogic Serverが起動し、接続プールが初期化された後、「**縮小頻度**」で設定した時間が経過したとき、接続プールが「**最小容量**」以上に増加していた場合、接続プール中の未使用の接続数は、「**最小容量**」値まで段階的に縮小される。（状況により一気に「**最小容量**」値まで縮小するケースもある）
- 使用中の接続は縮小対象にはならないが、それにより「**最小容量**」まで縮小できない場合は使用後に即縮小される。（縮小間隔もリセットされる。）
- 「**最小容量**」＝「**最大容量**」の場合は、当然ながら縮小されない。
- 縮小処理は、管理コンソールから手動で行うことも可能。



データソース初期化後
「縮小頻度」900秒の間隔で
「最小容量」3まで縮小



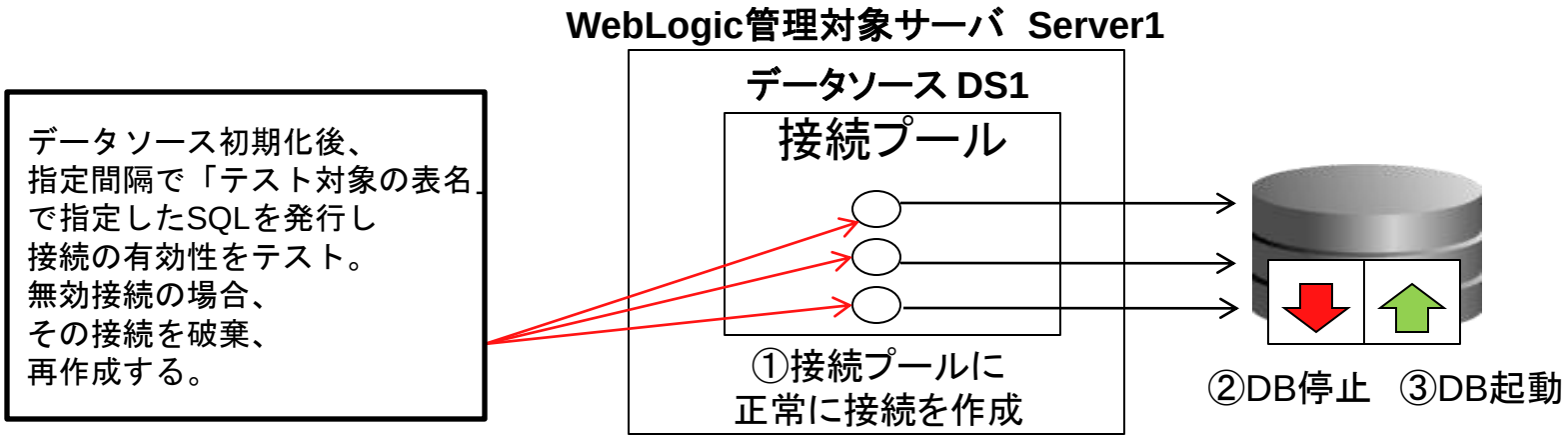
接続プール中の接続の有効性チェックに関連するパラメータ

- データベースの再起動など、接続プール中の接続が無効化されるようなケースにおいては下表のようなパラメータを活用し、接続を再度有効化させる、もしくはアプリケーションがこれ以上無効な接続を使わないようにさせることができる。
- 各パラメータの詳細は、次のスライドから説明。

カテゴリ	パラメータ	MBeanの属性名	デフォルト
接続プールのテスト機能	テスト頻度	JDBCConnectionPoolParamsBean.TestFrequencySeconds	120
	予約時に接続をテスト	JDBCConnectionPoolParamsBean.TestConnectionsOnReserve	false
接続プールのテストや接続の再作成が失敗した場合の制御機能	フラッシュされるまでのテストの失敗数	JDBCConnectionPoolParamsBean.CountOfTestFailuresTillFlush 12.1.2 New	2
	無効化されるまでのリフレッシュに失敗した接続数	JDBCConnectionPoolParamsBean.CountOfRefreshFailuresTillDisable 12.1.2 New	2

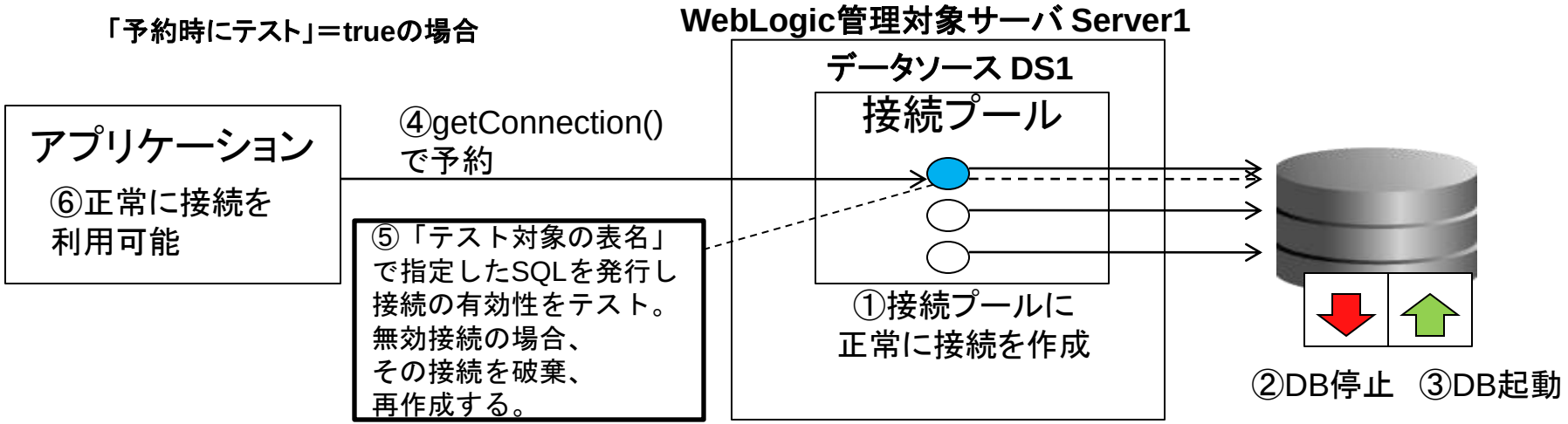
接続プールのテスト機能①（定期的なテスト）

- 接続プールのパラメータで、「**テスト頻度**」>0（秒数）に設定し、「**テスト対象の表名**」に有効なSQLまたは表名を設定する。その場合、データソースの初期化後、指定した値（秒数）毎に設定したSQLを発行し接続テストが行われる。
- もしSQLが成功しない場合はその接続を破棄し、新たに接続を作成する。



接続プールのテスト機能②（予約時のテスト）

- 接続プールのパラメータで、「**予約時にテスト**」をTrueにし、「**テスト対象の表名**」に有効なSQLまたは表名を設定する。
- 「**予約時にテスト**」をTrueにすると、アプリケーションが接続要求時に、接続プール中の接続が有効か実際にSQLを発行してテストを行う。もしSQLが成功しない場合はその接続を破棄し、新たに接続を作成する。（アプリケーションでは例外が発生せず、正常に接続を取得できる。）
- ただし、接続要求都度にSQLを発行するため、理論上、DB側の負荷が幾ばくか高まる。



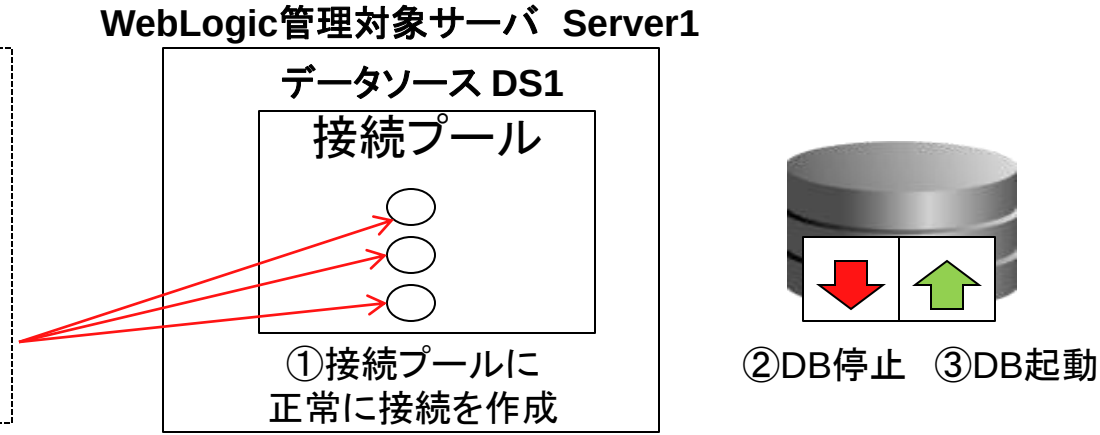
接続プールのテスト失敗に対応した制御機能

12.1.2 New

フラッシュされるまでのテストの失敗数
(JDBCConnectionPoolParamsBean.CountOfTestFailuresTillFlush)

- 接続プールのテスト機能で行われたテストの連続失敗数が「**フラッシュされるまでのテストの失敗数**」で設定した数に達した場合、接続プールの全ての接続を閉じ、5秒単位で接続の再作成が行われる。
- 再作成は、「最小容量」に達するまで行われる。

④接続プールのテスト機能によるテストが連続失敗して「**フラッシュされるまでのテストの失敗数**」に達した場合、全ての接続を閉じ5秒間隔で接続の再作成処理が始まり、「**最小容量**」に達するまでコネクション確立を行う。



接続プールの接続再作成失敗時に対応した制御機能

12.1.2 New

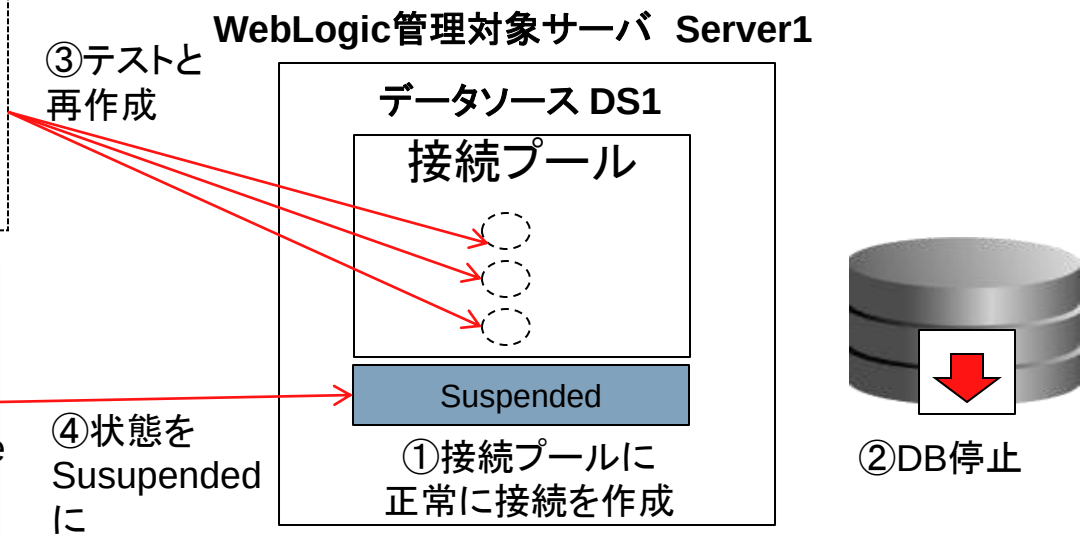
無効化されるまでリフレッシュに失敗した接続の数

(JDBCConnectionPoolParamsBean.CountOfRefreshFailuresTillDisable)

- 接続プールのテスト機能で行われたテストの連続失敗数と、前述の「**フラッシュされるまでのテストの失敗数**」に達した場合に開始される5秒間隔の再作成処理数の合計が「**無効化されるまでリフレッシュに失敗した接続の数**」に達した場合、データソース自体の有効性がFalseとなり、「状態」はSuspendedになる。その場合、 5秒間隔でのコネクション再作成処理も停止する。

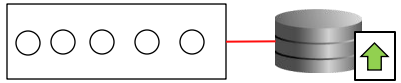
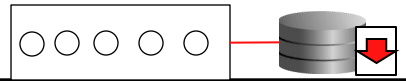
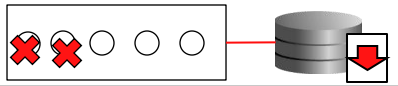
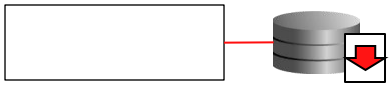
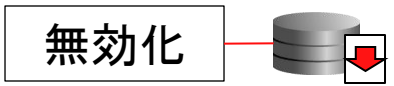
③テストの連続失敗数が「**フラッシュされるまでのテストの失敗数**」に達した場合、全ての接続を閉じ5秒間隔でコネクションの再作成処理が始まる

④テスト連続失敗数 + 5秒間隔での再作成処理の失敗数が「**無効化されるまでリフレッシュに失敗した接続の数**」に達した場合、データソースの「有効化」がFalseとなり、「状態」がSuspendedとなる。



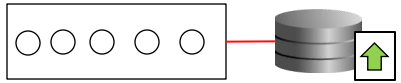
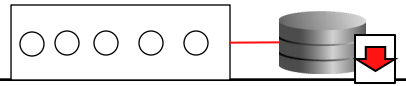
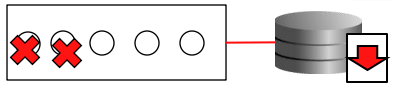
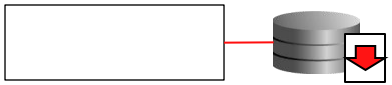
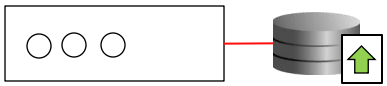
接続プールのテストとコネクション再作成のシーケンス①

- 下表は、DBが使用不能状態が続いている場合のシーケンス。

順	処理	イメージ
①	接続プールに正常にコネクションを確立。	
②	DBが停止。コネクションは無効状態になる。	
③	テスト機能によりテストされ失敗する。	
④	③での連続失敗数が「フラッシュされるまでのテストの失敗数」に達するとフラッシュ開始	
⑤	5秒毎にコネクション再作成が行われる	
⑥	③+ ⑤の失敗回数が「無効化されるまでリフレッシュに失敗した接続の数」に達するとデータソース自体が無効化（Suspended状態に）される	

接続プールのテストとコネクション再作成のシーケンス②

- 下表は、DBが使用不能状態から回復した場合のシーケンス。

順	処理	イメージ
①	接続プールに正常にコネクションを確立。	
②	DBが停止。コネクションは無効状態になる。	
③	テスト機能によりテストされ失敗する。	
④	③での連続失敗数が「フラッシュされるまでのテストの失敗数」に達するとフラッシュ開始	
⑤	5秒毎にコネクション再作成が行われる(DBは回復)	
⑥	③+ ⑤の失敗回数が「無効化されるまでリフレッシュに失敗した接続の数」に達する前に「最小容量」で指定した数までコネクションが再作成されると再作成処理は停止	

WebLogic Serverのデータソースの監視

データソースの監視

- WebLogic Serverでは、データソースの実行時の情報を監視、診断する仕組みを提供している。
- 下表は、データソースを監視・診断するための主要な方法をまとめたもの。

主要な方法	概要
JMX MBean による監視	主に、サーバやクラスタにデプロイされた JDBC データ ソースデータソースの実行時の統計情報を提供するJDBCDataSourceRuntimeMBeanを使い監視できる。MBeanは、管理コンソールやWLST、監視ダッシュボード、JMX APIや任意のMBeanブラウザ等を利用して監視できる。
プロファイル による診断	プロファイルの機能を利用し、特定の問題にフォーカスして情報を取得する事ができる。例えばコネクション・リークなどの問題が発生した場合、コネクションリーク用プロファイル情報を出力させることで問題の診断に役立てることができる。
デバッグログ出力による診断	データソースの動作に関してデバッグレベルのログを出力させ、データソースの動作の問題を分析する必要がある場合などに役立てることができる。

[参考] JDBCDataSourceRuntimeMBeanの統計情報①

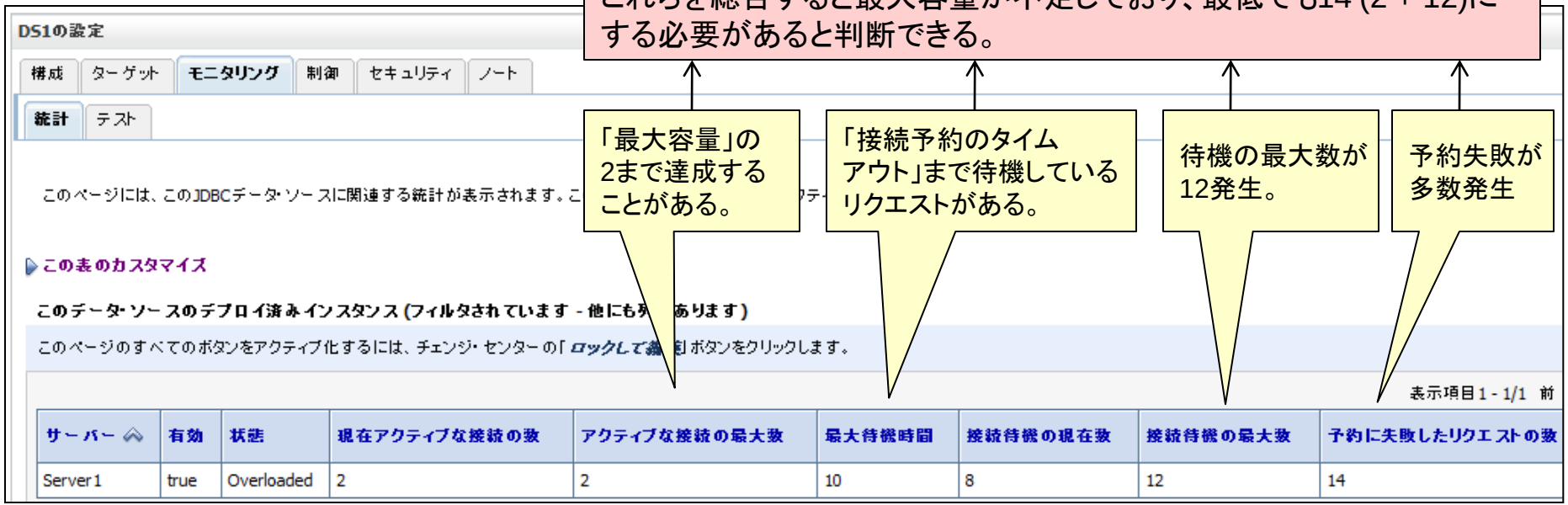
管理コンソールの監視項目名	JDBCDataSourceRuntimeMBeanの属性名	説明
アクティブな接続の平均数	ActiveConnectionsAverageCount	使用中接続の平均数
現在アクティブな接続の数	ActiveConnectionsCurrentCount	現在使用中の接続数
アクティブな接続の最大数	ActiveConnectionsHighCount	同時に使用された接続の最大数
接続遅延時間（msec)	ConnectionDelayTime	物理接続の作成に要した平均時間
接続の総数	ConnectionsTotalCount	データ・ソースで作成されたデータベース接続の累計数
現在の容量	CurrCapacity	接続プール中の接続数
予約に失敗した要求の数	FailedReserveRequestCount	アプリが接続予約に失敗した数
再接続の失敗数	FailuresToReconnectCount	データソースが物理接続のリフレッシュに失敗した回数
リークした接続数	LeakedConnectionCount	アプリがcloseしなかった接続数
使用可能数	NumAvailable	接続プール中の使用可能な接続数
使用不可数	NumUnavailable	接続プール中の未使用の接続数
予約された要求の数	ReserveRequestCount	接続要求の現在の累積数。

[参考] JDBCDataSourceRuntimeMBeanの統計情報②

管理コンソールの監視項目名	JDBCDataSourceRuntimeMBeanの属性名	説明
プリペアド・ステートメント・キャッシュのアクセス数	PrepStmtCacheAccessCount	文キャッシュにアクセスされた累計数
プリペアド・ステートメント・キャッシュの追加数	PrepStmtCacheAddCount	文キャッシュに追加された文の現在の累積数
プリペアド・ステートメント・キャッシュの現在サイズ	PrepStmtCacheCurrentSize	文キャッシュの現在の数
プリペアド・ステートメント・キャッシュの削除数	PrepStmtCacheDeleteCount	キャッシュから削除された文の数
プリペアド・ステートメント・キャッシュのヒット数	PrepStmtCacheHitCount	文キャッシュが使用された数
プリペアド・ステートメント・キャッシュの失敗数	PrepStmtCacheMissCount	文キャッシュが使用されなかった数
最大待機時間(秒)	WaitSecondsHighCount	接続待機の最大時間
接続待機の現在数	WaitingForConnectionCurrentCount	接続待機している現在の要求数
接続待機の失敗総数	WaitingForConnectionFailureTotal	接続待機後、接続予約に失敗した総数
接続待機の最大数	WaitingForConnectionHighCount	接続待機した要求の最大数
接続待機の成功総数	WaitingForConnectionSuccessTotal	接続待機後、接続を予約できた総数
接続待機の総数	WaitingForConnectionTotal	接続待機した要求数の総数

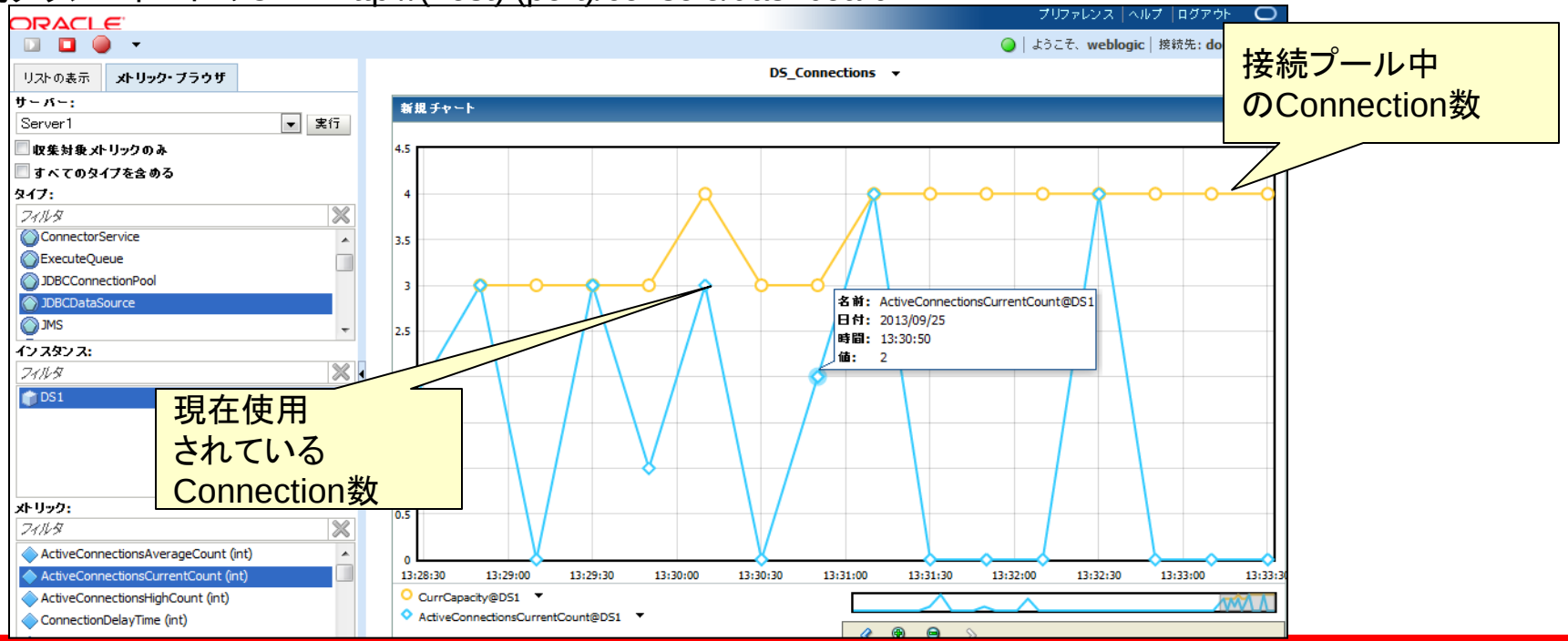
管理コンソールによるMBean監視例

- 接続プールの「最大容量」値が適切かどうかを監視する例
- 下記例では、「最大容量」を2、「接続予約のタイムアウト」を10秒に設定



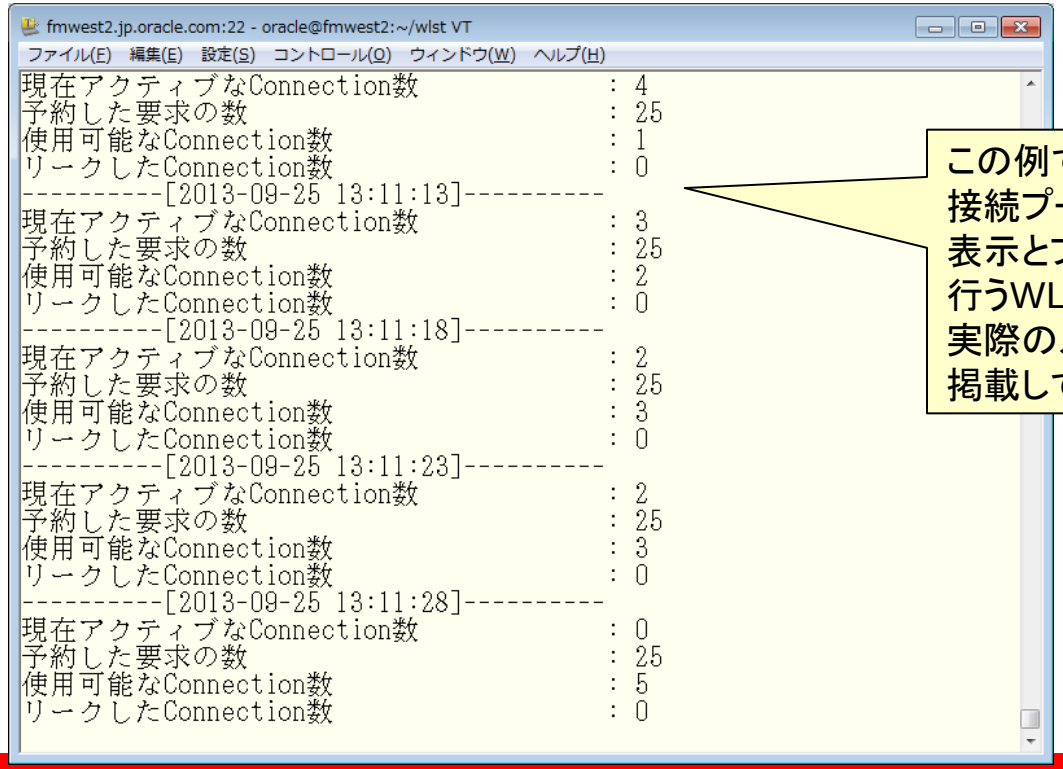
監視ダッシュボードによるMBean監視例 (WebLogic Server11g10.3.3~)

- 接続プールの状況を自動的にリフレッシュさせつつ、時系列に確認したい場合などは管理コンソールの監視ダッシュボードが便利
- 監視ダッシュボードのURL: http://(host):(port)/console/dashboard



WLSTのスクリプトによるMBean監視例

- WLSTを活用し、データソースの接続プールの監視項目を定期的にチェックしてファイルに書き込んだり標準出力に表示するような処理を行うスクリプトを作成できる。



この例では、5秒毎に
接続プールの状況を取得し
表示とファイル出力を同時に
行うWLSTのスクリプトを実行している。
実際のスクリプトは次のスライドに
掲載している。

WebLogic Server データソースにおける推奨・注意事項

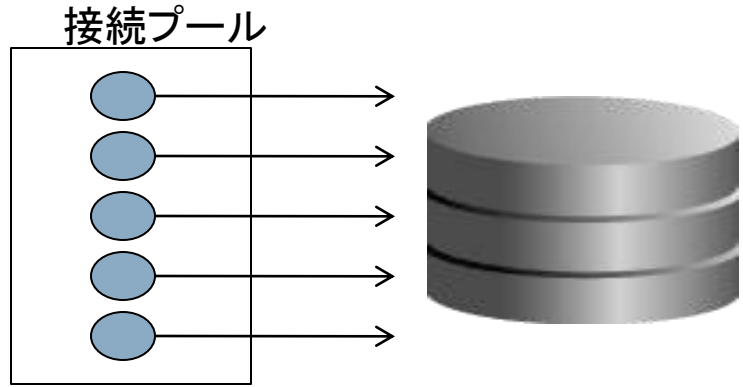
1. 接続プールの「初期容量」と「最大容量」について

- 接続プールの容量における推奨事項。

接続プールの「初期容量」と「最大容量」の値は同じにする。

- これは、下記理由によるもの。
  - 接続プールの初期容量と最大容量が異なる場合、同時リクエスト数によっては接続プール内で新たにDBへの物理接続処理が発生し得る。また接続プールの縮小処理も発生し得る。このDBへの物理接続や縮小時の切断処理は決して軽くない。
  - そのため運用中に接続プール中の接続が増減しないように「**初期容量**」や「**最大容量**」の値は同じにすることを推奨する。

初期容量5、最大容量5の場合、
WebLogic起動時に接続プールに
5つのConnectionオブジェクトが
作成され、その後増加も減少もしない。
つまり、その分、物理接続や切断の
オーバーヘッドが無い



2. 接続プールの「初期容量」と「最小容量」について

- 接続プールの容量における推奨事項。
- WebLogic Server11g(10.3.6)以降では、接続プールに「最小容量」というパラメータが追加されている。
- 「最小容量」の意味は「初期容量」とほぼ同じだが、「初期容量」の値と比較し、値が大きい場合に適用される。
- 例外として、データソース作成時に接続プールの「初期容量」を0に設定した場合は初期容量の値が優先される。ただし、作成済みのデータソースの接続プールの「初期容量」を0に変更した場合は「初期容量」の値と比較し、値が大きい場合に適用される。
- **運用の複雑性を回避するため、初期容量と最小容量は同じ値に設定する事を推奨する。**

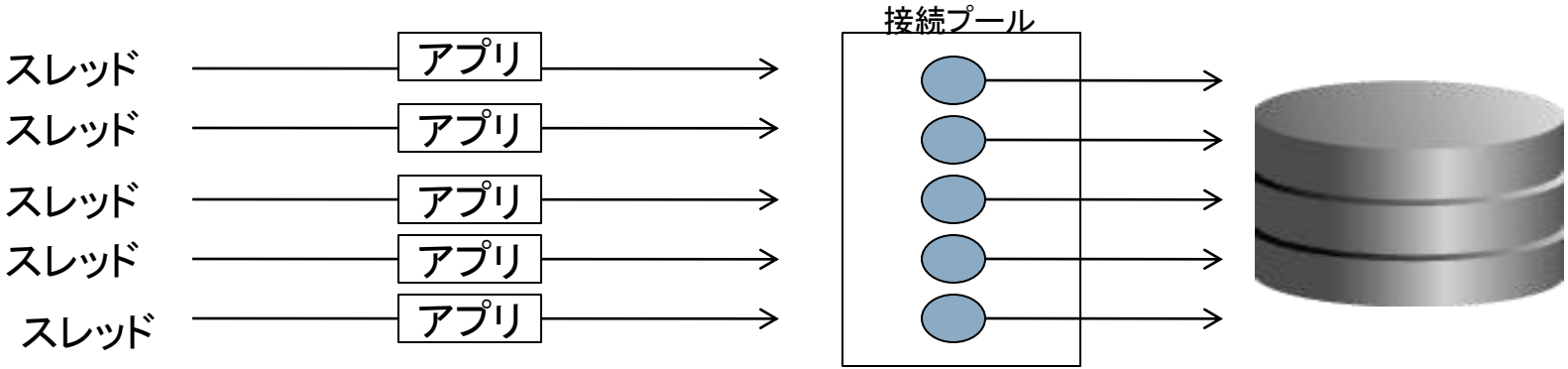
接続プールのパラメータ		接続プール中のコネクション数					
初期容量	最小容量	接続プール作成直後	サーバ起動時	中断→再開	停止→起動	DB再起動後(自動Suspend->Running)	初期容量から6に増加し、縮小後
5	5	5	5	5->5	0->5	5	6->5
5	3	5	5	5->5	0->5	3	6->3
3	5	5	5	5->5	0->5	5	6->5
0	5	0	0	0->0	0->0	0	6->5
5	0	5	5	5->5	0->5	1	6->3->1->0

3. 接続プールの容量とスレッド数について

- 接続プールの容量における推奨事項。

  WebLogic Serverで同時実行される最大スレッド数 <= 接続プールの「最大容量」にする。

- これは、下記理由によるもの。
  - WebLogic上のアプリケーションはスレッドにて処理されるが、同時実行スレッド数が多くなると、それに対応できる接続プール中の接続が無ければ接続予約時でスレッドの待機が発生してしまい、性能に影響が出る。
  - （ただしリクエストによりスレッドがDB接続を行わないこともあるという場合は、この限りではない）
  - WebLogicで同時実行される最大スレッド数はワークマネジャー機能でサーバまたはアプリケーション別に指定できる。その際に、最大スレッド数＝特定の接続プールの最大容量として指定することも可能。

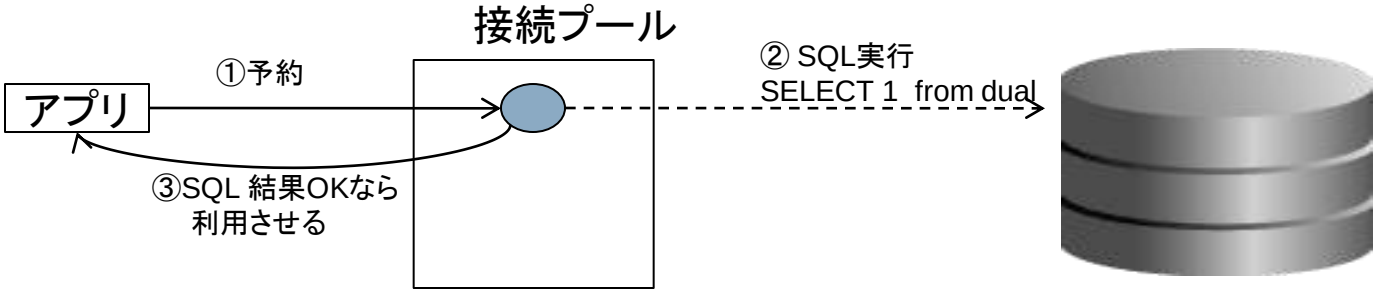


4. 接続プールのテストについて

- 接続プールのテストにおける推奨事項

  接続プールでテスト機能を設定するか否かは、データベース障害時の対応における要件と、データベース・サーバ側の負荷状況を鑑みて判断する。

- WebLogic Serverのデータソースにはテスト機能が存在し、アプリケーションが予約時、または指定頻度で指定したSQLを自動実行し、その結果でDBの死活状況をチェック可能。
- データベース・サーバ側の負荷（CPU使用率など）が非常に高い場合、接続プールのテスト機能は極力使用しない方がよい。
- （シングル構成の）データベースが運用中に、障害等で再起動してしまった場合で、その際に管理対象サーバまたはデータソースの再起動などの手動操作を行うことが許容されない場合は、**「接続予約時のテスト」**をtrueにし、かつ**「テスト頻度」**も設定し、接続テストを実施させる。

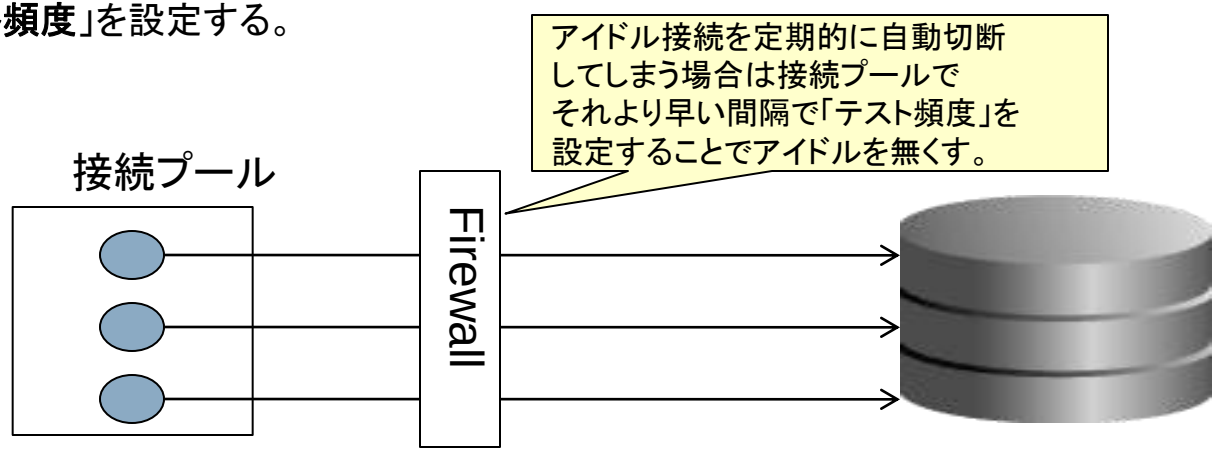


5. WebLogicとDB間のファイアーウォールについて

- WebLogicとDB間にファイアウォールが存在する場合の注意事項。

WebLogicとデータベースの間にFirewall等が設置されている場合、Firewallが接続プール中の接続とDB間のアイドルな通信を一定間隔で自動切断させないようにする必要がある。

- Firewallによっては、アイドルとなった通信を自動的に無効化する機能を提供するが、接続プールは、常時DBとの通信を保持するため、この影響を受ける可能性がある。
- その場合、Firewall側でその機能を無効化する。できない場合はFirewallがアイドル通信を自動切断するより早い間隔で、接続プールの「**テスト頻度**」を設定する。



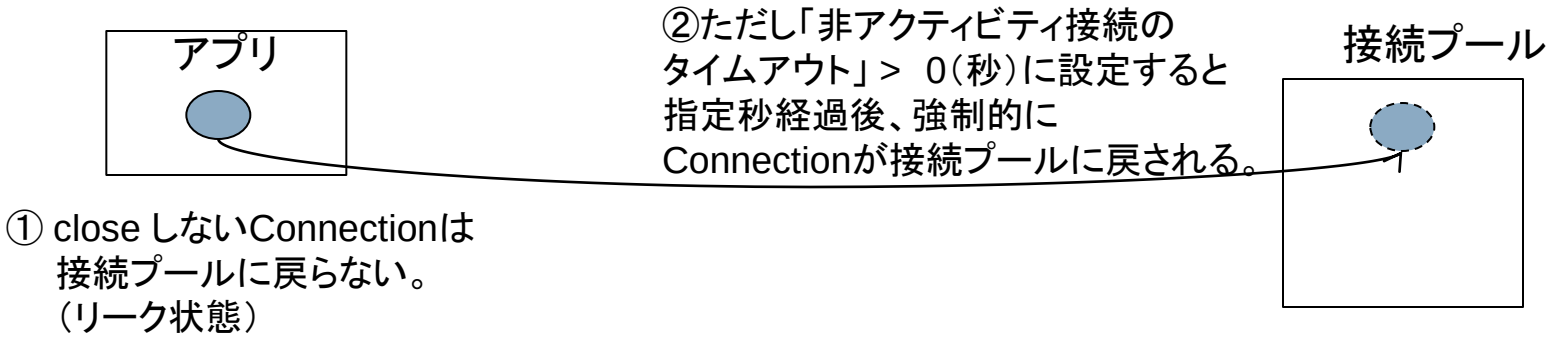
6. Connection リークへの対応について

- Connectionリークにおける注意事項。

  リークが発生してしまった場合、WebLogic側で強制解放する方法はあるが、

  そもそも、Connectionリークを発生させないようにする事が重要で、それは原則、アプリ側の責任。

- アプリケーション側で、利用したConnectionオブジェクトに対してclose( ) メソッドを実行しなかった場合、Connectionリークが発生する。
- WebLogicでは、データソースに「**非アクティビティ接続のタイムアウト**」を0よりも大きい値（秒）を指定すると、Connectionがリークした後、指定秒経過後に強制的に接続プールに戻すことができる。
- ただし、この機能に頼ることなくアプリ側でcloseをきちんと行う必要がある。



7. ステートメント・キャッシュについて①

- ステートメント・キャッシュにおける注意事項。

- **「ステートメント・キャッシュ数」×接続プールの「最大容量」×接続プール数がデータベース側で許容されるオープン・カーソル数を超えないようにする必要がある。**

- ステートメント・キャッシュ機能により、PreparedStatementがキャッシュされデータベースのカーソルをオープンしたままの状態で維持される。（データベース実装に依存）
- 「ステートメント・キャッシュ数」は、一つの接続当りのキャッシュ数になるため最大同時キャッシュ数がデータベースの許容オープンカーソル数を超えないようにする。

8. ステートメント・キャッシュについて②

- ステートメント・キャッシュにおける注意事項。

- **ステートメント・キャッシュが有効な状態では、運用中のデータベースの表定義を変更（列追加など）は、基本的に避ける。**

- ステートメント・キャッシュが有効な状態（「ステートメント・キャッシュ数」が0より大きい場合）で、キャッシュされたPreparedStatementで解析対象のSQL内に含まれるテーブルなどの定義を変更すると、次回キャッシュしたPreparedStatementを使用時にSQL Exceptionが発生するため。
- その場合、管理コンソールで「文キャッシュのクリア」の操作が必要。
- （例えばOracle Database 10gでは、下記のような例外が発生する。）

  java.sql.SQLException: プロトコル違反です。

  at oracle.jdbc.driver.SQLStateMapping.newSQLException

Hardware and Software

ORACLE

Engineered to Work Together


ORACLE®