Enterprise Server for WindowsR

ディプロイガイド

第 1 版
2003 年 5 月

このマニュアルでは、Interface Mapping Toolkit を使用してマップした既存の COBOL に必要なソフトウェアをディプロイする方法を説明します。既存の COBOL は、Web サービスや EJB などの外部インターフェイスの範囲でマップされている場合があります。

第 1 章 : 概要

ここでは、Interface Mapping Toolkit で生成されたインターフェイスをディプロイする方法を説明します。

1.1 サービスインターフェイスのディプロイ

Interface Mapping Toolkit を使用して生成したインターフェイスを実行するには、マッピング情報と、場合によってはその他の情報をエンタープライズサーバにディプロイする必要があります。

Interface Mapping Toolkit のディプロイツールは、次の作業ができます。

- 必要に応じて、WSDL ファイル、EJB、または COM オブジェクトを生成します。WSDL ファイルは、Web サービスを定義します。EJB は、Java アーカイブファイル (.jar) に、ディプロイディスクプリタのような他の必要なファイルとともにパッケージ化されます。COM オブジェクトは、すべてのマッピング情報を格納する .dll です。
- マッピング情報をエンタープライズサーバに、他の必要な COBOL ファイルやデータファイルとともにディプロイします。これは、Web サービスと EJB でのみ行い、COM オブジェクトでは行いません。
- マッピング情報と COBOL アプリケーションファイルを、COBOL アーカイブファイル (.car) にパッケージ化します。この .car は、再度ディプロイツールを使用しないで手作業で、Web サービスや EJB で別のエンタープライズサーバにディプロイすることもできます。

マッピング情報と COBOL をエンタープライズサーバにディプロイした後は、生成したインターフェイスのタイプによっては、さらにソフトウェアをディプロイします。

- Web サービスでは、これ以上ディプロイするものはありません。
- EJB の場合は、Java アーカイブファイル (.jar) を J2EE アプリケーションサーバにディプロイする必要があります。また、リソースアダプタもディプロイする必要があります。
- COM オブジェクトでは、コンポーネント .dll を Windows に登録する必要があります。また、COBOL アプリケーションを Application Server for Net Express にディプロイするか、または Net Express 内でアプリケーションを実行している場合は、COBDIR 環境変数を設定する必要があります。

  方法

最後に、マップしたサービスインターフェイスを使用するように、クライアントインターフェイスをディプロイできます。Web サービスの COBOL とのインターフェイスの場合には、クライアントプログラムは Web サービス要求をエンタープライズサーバに送り、エンタープライズサーバ側では公開された COBOL を呼び出します。このマニュアルでは、クライアントインターフェイスについては説明しません。

1.2 エンタープライズサーバのディプロイ

Interface Mapping Toolkit を使用して生成した Web サービスまたは EJB インターフェイスを実行するには、インターフェイスをエンタープライズサーバにディプロイする必要があります。インターフェイスはマッピング情報、COBOL プログラム、および必須データからなります。

インターフェイスは、Interface Mapping Toolkit のディプロイツールを使用しても、手動でもディプロイできます。

ディプロイツールを使用する場合は、ディプロイ設定で何をどのようにディプロイするかを定義できます。デフォルトでは、このツールによって ESDEMO と呼ばれるエンタープライズサーバにディプロイされます。異なるエンタープライズサーバを使用したい場合は、.mfdeploy ファイルで指定する必要があります。また、エンタープライズサーバが異なるマシン上で動作している場合は、mf-client.dat ファイルにリモートマシン名を指定する必要があります。

方法

ディプロイ設定と、上記のファイルにエンタープライズサーバを指定した場合は、[ディプロイ] をクリックし、必要事項をすべてディプロイします。

ディプロイメントについての情報は、次の場所で確認できます。

- 「ディプロイメントの進捗」ウィンドウ。ディプロイツールを使用している間表示されます。
- NetExpress\base\deploy\deploylog.txt。「ディプロイメントの進捗」ウィンドウに表示された内容の記録です。

ディプロイツールを使用しないで、手動でもディプロイできます。ディプロイするファイルは、COBOL アーカイブファイル myservice.car にまとめてパッケージ化します。このファイルは、ディプロイメントフォルダ myservice\Repos\myservice.deploy に格納されています。COBOL アーカイブの重要なファイルを、次に示します。

- インターフェイス定義テーブル (IDT)。たとえば、myservice.idt。
- 実装パッケージ。たとえば、myprogram.int および myprogram.idy。
- 必須データ。たとえば、mydata.idx および mydata.dat。
- マニフェストファイル manifest.mf。ディプロイメントプロセスを制御します。

サービスインターフェイスを手動でディプロイするには、次のことを行うために Enterprise Server Administration を使用します。

1. 実装パッケージを追加する。IDT、実行形式ファイル、およびデータファイルのパスと名前を指定します。
2. サービスを追加する。このサービスで使用するリスナーと、サービスの機能を提供する実装パッケージを指定します。

方法

第 2 章 : EJB とリソースアダプタ

ここでは、Interface Mapping Toolkit によって生成される EJB のディプロイと、リソースアダプタのディプロイの方法を説明します。

2.1 概要

EJB をディプロイする場合は、2 つの作業が必要です。COBOL 側をエンタープライズサーバにディプロイし、Java 側を J2EE アプリケーションサーバにディプロイします。

Interface Mapping Toolkit でディプロイツールを使用すると、COBOL 側がディプロイされます。また、ディプロイツールでは EJB が生成され、Java アーカイブファイル myservice.jar にパッケージ化されます。ディプロイツールの詳細については、『概要』の章にある『エンタープライズサーバのディプロイ』を参照してください。

Java 側では、次に示すファイルを J2EE 上で動作する、WebSphere や WebLogic などのサードパーティのアプリケーションサーバにディプロイします。

- Java アーカイブファイル myservice.jar。このファイルには、生成された EJB と必要なディプロイメントの情報が格納されています。

  .jar ファイルをアプリケーションアーカイブファイル (.ear) に、他のアーカイブファイルとともにパッケージ化し、.jar ファイルでなく、.ear ファイル全体をディプロイする必要がある場合もあります。

- 提供されているリソースアダプタの 1 つである mfcobol*.rar。リソースアダプタを使用すると、EJB が J2EE アプリケーションサーバ経由でエンタープライズサーバと通信できます。リソースアダプタは、J2EE コネクタとも呼ばれ、リソースアーカイブファイル (.rar) にパッケージ化されています。

そして、J2EE マシン上でサードパーティのアプリケーションサーバを構成する必要があります。アプリケーションサーバによって、ディプロイメント要件や使用するユーティリティは異なります。Net Express では、次のものをサポートします。

- J2EE 1.3.1 Reference Implementation (RI) サーバ (開発での使用のみ)
- WebLogic 7
- WebSphere 5

2.2 リソースアダプタ

Micro Focus 製のリソースアダプタを使用すると、J2EE 上にディプロイした EJB は エンタープライズサーバ上の COBOL と連携できるようになります。リソースアダプタは、J2EE コネクタとも呼ばれます。

次のリソースアダプタは、Net Express の base\bin ディレクトリに格納されています。また、Enterprise Server が収録されている CD の bin ディレクトリにも格納されています。

- mfcobol-notx.rar - トランザクションはサポートしません。
- mfcobol-localtx.rar - ローカルトランザクションをサポートします。
- mfcobol-xa.rar - XA トランザクションをサポートします。

EJB を Enterprise Server 上で動作する COBOL と連携させるには、Java 環境にリソースアダプタをディプロイする必要があります。一度にディプロイできるのは、1 つのリソースアダプタのみです。このリソースアダプタは、エンタープライズサーバに送られるすべてのバイナリ要求を処理します。

注:リソースアダプタは、Enterprise Server または Net Express のどちらかがインストールされたマシン上にディプロイする必要があります。

2.3 ディプロイメントディスクプリタ

ディプロイツールを使用して EJB を生成すると、結果の myservice.jar ファイルには他のファイルとともに、ディプロイメントディスクプリタ ejb-jar.xml と、WebLogic のディプロイメントディスクプリタ weblogic-ejb-jar.xml が含まれます。WebSphere や Sun 社の Reference Implementation では、特定のディプロイメントディスクプリタは必要ありません。

ディプロイメントディスクプリタには、Interface Mapping Toolkit でユーザが指定した設定が含まれます。次に、MapDemo デモを実行したときに作成され、NX\base\demo\Mapdemo\Mapdemo\repos\JMapServ.deploy\JMapServ.jar にパッケージ化されるディプロイメントディスクプリタ ejb-jar.xml の一部を示します (NX はインストールディレクトリ)。

```
<ejb-jar>
    <description />
    <display-name>JMapServ</display-name>
    <enterprise-beans>
        <session>
            <ejb-name>JMapServEJB</ejb-name>
            <home>com.mypackage.JMapServ.JMapServHome</home>
            <remote>com.mypackage.JMapServ.JMapServ</remote>
            <ejb-class>com.mypackage.JMapServ.JMapServBean</ejb-class>
            <session-type>Stateless</session-type>
            <transaction-type>Container</transaction-type>
            <resource-ref>
                <res-ref-name>CCIMFCobol_v1.0</res-ref-name>
                <res-type>javax.resource.cci.ConnectionFactory</res-type>
                <res-auth>Container</res-auth>
                <res-sharing-scope>Shareable</res-sharing-scope>
            </resource-ref>
```

```
        </session>
    </enterprise-beans>
    <assembly-descriptor>
        <container-transaction>
<method>
                <ejb-name>JMapServEJB</ejb-name>
                <method-name>Add</method-name>
<method>
            <trans-attribute>NotSupported</trans-attribute>
        </container-transaction>
   ...
```

2.4 マニフェストファイル

ディプロイツールを使用して EJB インターフェイスを生成すると、結果の myservice.jar ファイルには、マニフェストファイル manifest.mf が含まれます。このマニフェストファイルは、mfejblib.jar へのクラスパスを示します。これにより、Micro Focus 社は CustomRecord と RuntimeProperties のインターフェイスをサポートできます。

マニフェストファイルでは、次のようにクラスパスを指定することにより、mfejblib.jar がディプロイされたアーカイブファイルのルートにあることを指示します。

```
Class-path: mfejblib.jar
```

mfejblib.jar を別の位置に置く必要がある場合は、クラスパスをそれに従って変更する必要があります。Sun 社の Reference Implementation を使用する場合は、これを行う必要があります。詳細は、『EJB の J2EE RI へのディプロイ』を参照してください。

2.5 CustomRecord および他の EJB のサポート

EJB が CustomRecord または RuntimeProperties のインターフェイスを使用する場合は、EJB でこれらのインターフェイスを使用できるようにするために、サポートする必要があります。NX\base\bin\mfejblib.jar に、これらのサポートが用意されています (NX は Net Express のインストールディレクトリ)。

COBOL 集団項目を Java データ型にマップするとカスタムレコードが作成されます。集団項目の再利用可能なマッピングを設定すると、Interface Mapper でその集団項目をマップでき、複雑なインターフェイスフィールドに対応付けることができます。

EJB をディプロイする前に、ディプロイするアーカイブファイル (エンタープライズアーカイブ .ear ファイルまたは EJB の .jar ファイル) に mfejblib.jar ファイルをインクルードする必要があります。また、mfejblib.jar は、.ear または .jar ファイルの置かれている位置のルートに置く必要があります。これは、mfejblib.jar へのクラスパスが、EJB の .jar ファイルのマニフェストファイル manifest.mf のルートに設定されているためです。

2.6 J2EE 1.3.1 Reference Implementation

J2EE 1.3.1 Reference Implementation (RI) は、アプリケーションサーバとして機能します。これは実際の運用ではなく、開発の際に使用します。これは、J2EE Software Development Kit (SDK) の一部として提供されています。EJB とリソースアダプタをディプロイするには、これをインストールする必要があります。

J2EE RI に関連するのは、次のものです。

- J2EE RI サーバ自体。
- 管理ユーティリティ。これは、主にリソースアダプタを Java Naming and Directory Interface (JNDI) 名にバインドするために使用します。JNDI 名は、容易に識別できるようにリソースに付けた名前です。ヘルプを表示するには、次のコマンドを使用します。

  ```
  j2eeadmin -help
  ```

- ディプロイツールユーティリティ。これは、リソースアダプタと EJB をディプロイします。このユーティリティには、グラフィカルユーザインターフェイス (GUI) とコマンド行インターフェイスがあります。ヘルプを表示するには、次のコマンドを使用します。

  ```
  deploytool -help
  ```

サーバ上にディプロイしたモジュールを変更した場合は、Sun 社のマニュアルでは、サーバを停止し再開するよう推奨しています。

注:エンタープライズサーバを停止し再開すると、J2EE RI サーバも停止し再開する必要があります。

2.6.1 J2EE Reference Implementation のインストール

J2EE RI を使用するには、次に示すソフトウェアが必要です。これらは、Sun 社の Web サイトからダウンロードできます。

- Java Development Kit (JDK) 1.3.x 以降。インストールした後に、JAVA_HOME を JDK のインストールディレクトリに設定します。
- J2EE SDK。これには、J2EE 1.3.x RI のソフトウェアとマニュアルが含まれています。SDK をインストールした場合は、J2EE_HOME 環境変数を SDK のインストールディレクトリに設定します。

環境変数とパスを設定するサンプルスクリプトを、次に示します。SDK と JDK は、c: ドライブのデフォルト位置にインストールされていると仮定しています。

```
set java_home=c:\jdk1.3.1_01
set j2ee_home=c:\j2sdkee1.3.1
set path=c:\j2sdkee1.3.1\bin;c:\jdk1.3.1_01\bin;%path%
```

インストールの内容を検証するには、Web ブラウザを開き、URL http://localhost:8000 を入力します。インストールが成功した場合は、J2EE 1.3 のデフォルトのホームページが表示されます。

2.6.2 リソースアダプタの J2EE RI へのディプロイ

リソースアダプタのディプロイには、J2EE マシン上でサードパーティのアプリケーションサーバを、EJB、リソースアダプタ、および Enterprise Server がそれぞれ互いの位置を認識するように構成する作業が含まれます。

リソースアダプタを J2EE RI にディプロイするには、次を実行します。

1. リソースアダプタをディプロイツールユーティリティの -deployConnector コマンドを使用してディプロイする。
2. j2eeadmin ユーティリティの -addConnectorFactory を使用して接続ファクトリを追加する。これにより、リソースアダプタが Enterprise Server の JNDI 名 (eis/MFCobol_v1.0) にバインドされます。
3. EJB のリソースリファレンスで指定することで、リソースアダプタが EJB によって確実に使用されるようにする。『EJB の J2EE RI へのディプロイ』を参照してください。

次のサンプルスクリプトでは、デフォルトポート 9003 を使用して、リソースアダプタ mfcobol-notx.rar をローカルマシン localhost 上で動作する J2EE RI サーバにディプロイします。J2EE は c:\j2sdkee1.3.1 にインストールされていると仮定しています。

```
set RAR=mfcobol-notx.rar
copy "c:\NetExpress\base\bin\%RAR%" c:\j2sdkee1.3.1\lib\connector
call deploytool -deployConnector c:\j2sdkee1.3.1\lib\connector\%RAR%
localhost
call j2eeadmin -addConnectorFactory eis/MFCobol_v1.0 %RAR% Trace=true
call deploytool -listConnectors localhost
```

方法

2.6.3 EJB の J2EE RI へのディプロイ

EJB のディプロイには、J2EE マシン上でサードパーティのアプリケーションサーバを、EJB、リソースアダプタ、および Enterprise Server がそれぞれ互いの位置を認識するように構成する作業が含まれます。

EJB がカスタムレコードを使用する場合は、それらがサポートされているか確認する必要があります。COBOL 集団項目を Java データ型にマップするとカスタムレコードが作成されます。集団項目の再利用可能なマッピングを設定すると、Interface Mapper でその集団項目をマップでき、複雑なインターフェイスフィールドに対応付けることができます。

CustomRecord インターフェイスのサポートは、mfejblib.jar で提供されます。ディプロイツールにより service.jar ファイルが作成されると、マニフェストファイル manifest.mf が .jar ファイル

に追加されます。このマニフェストファイルには mfejblib.jar へのクラスパスが設定されています。mfejblib.jar への正しいパスが指定されていることを確認する必要があります。

EJB を J2EE RI にディプロイするには、ディプロイツールユーティリティを使用し、次を行う必要があります。

1. EJB のリソースリファレンスでリソースアダプタのコード名を指定する。

   リソースリファレンスは、ディプロイメントディスクリプタの 1 つの要素です。この場合は、リソースリファレンスで、接続ファクトリを示すリソースアダプタのコード名を定義する必要があります。接続ファクトリ eis/MFCobol_v1.0 のコード名は、CCIMFCobol_v1.0 です。

2. EJB がカスタムレコードを使用する場合は、マニフェストファイルで library\mfejblib.jar を指すクラスパスを指定する。

方法

2.7 WebLogic のディプロイ

WebLogic は開発と運用の際に使用するサードパーティのアプリケーションサーバです。インストールするには、WebLogic のインストールの指示書に従ってください。

WebLogic には、EJB とリソースアダプタのディプロイに使用する Server Console が付属しています。このコンソールで、ディプロイメントを手順に従って行います。また、オンライン情報を利用できます。このコンソールは、左側のペインにあるツリー構造をナビゲートし、対象の項目をクリックできます。

サーバ上にディプロイしたモジュールを変更した場合は、WebLogic のマニュアルでは、サーバを停止し再開するよう推奨しています。WebLogic マシンをリブートするたびに、WebLogic サーバも再開する必要があります。

2.7.1 WebLogic の設定

WebLogic を使用するには、その指示書に従ってソフトウェアをインストールする必要があります。

次に、ディプロイする EJB とリソースアダプタを保持するドメインを設定する必要があります。

最後に、WebLogic の環境を、起動スクリプトにクラスパス環境変数を記述することで設定する必要があります。サンプルスクリプトの一部を、次に示します。WebLogic と Net Express は、d: ドライブにインストールされていると仮定しています。

```
set java_home=d:\bea\jdk131_03
set mf_home=d:\NetExpress\base\bin
```

```
set
classpath=%JAVA_HOME%\lib\tools.jar;%MF_HOME%\mfconnector.jar;%MF_HOME%\mfcob
olpure.jar
```

方法

注：マシンをリブートするたびに、WebLogic サーバも再開する必要があります。[スタート] メニューから WebLogic サーバを起動する必要があります。Windows のコントロール パネルを使用して起動することはできません。またコントロール パネルで、マシンをリブートしたときに自動的に起動するように設定することはできません。

2.7.2 リソースアダプタの WebLogic へのディプロイ

WebLogic サーバのコンソールを使用してリソースアダプタをディプロイします。このコンソールでは、ディプロイメントを手順に従って行います。

主な手順は、Micro Focus COBOL リソースアダプタをロードすることです。

接続ファクトリを作成したり、リソースアダプタの JNDI 名を指定したりする必要はありません。これらは、すでに接続ファクトリのディプロイメントディスクリプタに指定されています。このディプロイメントディスクリプタは、リソースアダプタのアーカイブファイル mfcobol*.rar に含まれています。ディプロイメントディスクリプタ weblogic-ra.xml に関連する行を、次に示します。

```
<connection-factory-name>CCIMFCobol_v1.0</connection-factory-name>
<jndi-name>eis/MFCobol_v1.0</jndi-name>
```

このファイルは、アーカイブファイル mfcobol-notx.rar で確認できます。

方法

2.7.3 EJB の WebLogic へのディプロイ

WebLogic Server Console を使用して EJB をディプロイします。このコンソールでは、ディプロイメントを手順に従って行います。主な手順は、次のとおりです。

1. EJB が含まれる Java アーカイブファイル (.jar) を、アプリケーションのエンタープライズアーカイブファイル (.ear) にパッケージ化します。
2. EJB のディプロイメントディスクリプタを作成し、次を指定します。
    - EJB の JNDI 名。どのような名前も付けることができますが、アプリケーションの他の構成要素がその EJB を参照するときに使用する JNDI 名である必要があります。
    - リソースアダプタのリソースリファレンス名 CCIMFCobol_v1.0 および JNDI 名 eis/MFCobol_v1.0。ここに示した名前を使用する必要があります。

    たとえば、ディプロイメントディスクリプタには次の行を記述する必要があります。

```
<resource-description>
    <res-ref-name>CCIMFCobol_v1.0</res-ref-name>
        <jndi-name>eis/MFCobol_v1.0</jndi-name>
</resource-description>
...
<jndi-name>ejb/JMapServEJB</jndi-name>
```

3. .ear ファイルをインストールします。

方法

2.8 WebSphere のディプロイ

WebSphere は開発と運用で使用するサードパーティのアプリケーションサーバです。インストールするには、WebSphere のインストールの指示書に従ってください。

WebSphere には、EJB とリソースアダプタのディプロイに使用する Administrative Console が付属しています。このコンソールで、ディプロイメントを手順に従って行います。また、オンライン情報を利用できます。このコンソールは、左側のペインにあるツリー構造をナビゲートし、対象の項目をクリックできます。

サーバ上にディプロイしたモジュールを変更した場合は、WebSphere のマニュアルでは、サーバを停止し再開するよう推奨しています。

2.8.1 リソースアダプタの WebSphere へのディプロイ

WebSphere Administrative Console を使用してリソースアダプタをディプロイします。Administrative Console では、ディプロイメントを手順に従って行います。主な手順は、次のとおりです。

1. Micro Focus COBOL リソースアダプタをロードします。
2. 新しい J2C Connection Factory を追加します。

   次を指定します。

   - 接続ファクトリの名前 CCIMFCobol_v1.0
   - Enterprise Server の JNDI 名 eis/MFCobol_v1.0

方法

2.8.2 EJB の WebSphere へのディプロイ

WebSphere Administrative Console を使用して EJB をディプロイします。Administrative Console では、ディプロイメントを手順に従って行います。次の作業を実行する必要があります。

1. EJB が含まれる Java アーカイブファイル (.jar) を、アプリケーションのエンタープライズアーカイブファイル (.ear) にパッケージ化します。
2. .ear ファイルをインストールします。
3. EJB で使用する JNDI 名を指定します。どのような名前も付けることができますが、アプリケーションの他の構成要素がその EJB を参照するときに使用する JNDI 名である必要があります。
4. EJB 参照を EJB の JNDI 名にマップします。
5. EJB がリソースアダプタを見つけられるように、EJB リソースリファレンスをリソースアダプタの JNDI 名にマップします。Micro Focus 製のリソースアダプタの JNDI 名 は、eis/MFCobol_v1.0 です。

方法

2.9 管理非対象接続のディプロイ

管理非対象接続をディプロイするには、Micro Focus 製のリソースアダプタへのサポートを提供するクラスを含むようにクラスパスを更新するのみです。クラスは、mfcobolpure.jar と mfconnector.jar にパッケージ化されています。

方法

2.10 トラブルシューティング

ディプロイした EJB が正常に機能しない場合には、次のことを確認してください。

- アプリケーションサーバを停止し再開しましたか?ディプロイメントと構成の変更を有効にするために、これを行うことが必要な場合があります。WebLogic マシンをリブートするたびに、WebLogic サーバも [スタート] メニューから起動する必要があります。
- 必要なエンタープライズサーバが動作していますか?
- エンタープライズサーバに必要なサービスがディプロイされていますか?ディプロイに成功していますか?これらは、Enterprise Server Administration Console (http://locahost:86)、NX\base\deploy\...\deploylog.txt、および NX\base\workarea\myes\console.log を参照することで確認できます (NX は Net Express のインストールディレクトリ、myes は ESDEMO などのエンタープライズサーバ名)。
- リソースアダプタのログファイル c:\cobcon.log に情報が書き込まれていませんか?このログファイルは、Java システムの user.home 属性で定義したディレクトリに置かれています。このディレクトリは、デフォルトではユーザのプロファイルディレクトリと同じです。ログファイルに何の記述もない場合には、障害はリソースアダプタが起動する前に発生しています。障害は、アプリケーションサーバの設定時に発生した可能性があります。バックアップログファイル cobcon1.log および cobcon2.log が存在するか確認してください。
- ディプロイツールを使用して EJB を作成した場合に、アプリケーションサーバの J2EE コネクタクラスを使用しましたか?たとえば、WebLogic EJB の生成では WebLogic.jar、

WebSphere EJB の生成では j2ee.jar を使用する必要があります。これらのクラスはディプロイツールの Classpath 設定で指定します。

- リソースアダプタの JNDI 名 が eis/MFCobol_v1.0 に設定されていますか?この JNDI 名は、EJB のリソースリファレンスにマップされていますか?
- 各 EJB の JNDI 名は、それを参照する際に使用される名前と常に同じですか?EJB に JNDI 名を割り当てた場合は、その EJB を使用するすべてのクライアントや他のソフトウェアはその同じ JNDI 名を使用する必要があります。
- EJB とリソースアダプタは、アプリケーションサーバに適切にディプロイされましたか?

アプリケーションサーバのログファイルにどのような障害が記録されましたか?サポートされているすべてのアプリケーションサーバは、logs サブディレクトリを保有します。