Tomcat から WebOTX AS への
移行ガイド

WebOTX V8 編

2008.12.18
第 3.1 版

NEC
第二システムソフトウェア事業部

前書き

【本書の位置付け】

本書は、Apache Tomcat プロジェクトで開発が進められているサーブレット・コンテナである Tomcat を利用して構築したシステムを NEC 製のアプリケーションサーバである WebOTX に移行する際に参考となる情報を集めたガイドブックです。

移行後のトラブルやチューニングに関しては、別資料の「WebOTX AS Web コンテナ チューニングとトラブルシューティング」の資料もご覧ください。

対象とする移行パターンは以下のとおりです。

Tomcat 3.x → WebOTX V8.x

Tomcat 4.x → WebOTX V8.x

Tomcat 5.0.x → WebOTX V8.x

Tomcat 5.5.x → WebOTX V8.x

Tomcat 6.x → WebOTX V8.x

改版履歴

版数	年月日	改訂内容	備考
3.0	2008/10/30	第 3.0 版	
3.1	2008/11/18	第 3.1 版	
3.1	2008/11/18	WebOTX V6, V7 から V8 へ移行する場合について記載	
3.1	2008/12/18	2.6.3 章 nec-web.xml の記述内容を修正	

目次

1. はじめに

Web システム構築において、コスト削減の面からオープンソース導入が注目されています。しかし、保守運用までを見据えた場合、本当にコスト削減が達成できるのか、という疑問の声もよく耳にします。

WebOTX V8 シリーズでは、サン・マイクロシステムズ社が中心に仕様策定した、エンタープライズ領域の Java 仕様「Java Platform, Enterprise Edition 5 (Java EE 5)」に準拠した製品を提供しています。WebOTX AS は、その機能群の中の Web 層として、Apache Tomcat プロジェクトのオープンソース Tomcat 6.0 をベースに Servlet や JSP などの Web アプリケーション実行環境(Web コンテナ)を組み込んでいます。

WebOTX V8.1 以降で提供する Web コンテナは、Apache Tomcat 6.0.16 をベースに品質の改良、機能の追加、性能の向上、運用性の強化などを施しています。このため、現在 Apache Tomcat (以下、略して Tomcat と呼びます) 上で動作している Web アプリケーションを少ない労力で WebOTX に移行でき、かつ、システムの信頼性、運用性、性能等の向上を手に入れることができます。

対象読者

本書は、既に Tomcat を利用して何らかのシステムを運用している方や、利用経験のある方、Tomcat の知識を有している方などを対象に、既存の Tomcat による構築済みシステムを WebOTX に移行する際の作業を支援するガイドブックです。

対象とする移行パターンは以下のとおりです。

Tomcat 3.x → WebOTX V8.x

Tomcat 4.x → WebOTX V8.x

Tomcat 5.0.x → WebOTX V8.x

Tomcat 5.5.x → WebOTX V8.x

Tomcat 6.x → WebOTX V8.x

表記について

パス名表記

本書ではパス名の表記については特に OS を限定しない限りセパレータはスラッシュ'/'で統一しています。Windows 環境においては'¥'に置き換えてください。

環境変数表記

インストールディレクトリやドメインルートディレクトリなど環境によって値の異なるものについては環境変数を用いて表します。

${*env*} または $(*env*)で表しています。

例)

${AS_INSTALL} : インストールディレクトリ

${ INSTANCE_ROOT}: ドメインルートディレクトリ

1.1. ソフトウェア条件

WebOTX がサポートするソフトウェアについて説明します。

WebOTX に移行することにより、J2SE SDK やデータベースのバージョンを変更する場合、関連するソフトウェアのバージョンアップの必要性を確認してください。

詳細については、「WebOTX マニュアル　セットアップガイド」ー「1. 使用上の条件」をご覧ください。

1.1.1. 対象プラットフォーム

WebOTX V6.5 がサポートする OS と J2SE SDK のバージョンは次の表のとおりです。

OS	J2SE 1.4.2	J2SE 5.0
Windows 2000 Server Windows Server 2003	○	○
Windows Server 2003(IPF 版)	○※	×
Windows Server 2003 x64 Edition	×	○ (Update 4 以降)
HP-UX 11i v1	○※ (1.4.2.05 以降)	○※
HP-UX 11i v2	○ (1.4.2.05 以降)	○ (5.0.02 以降)
Solaris 8 Solaris 9	○※	○※
RHEL AS/ES 4.0	○	○

※WebOTX V6.31 でのサポート

その他のバージョンの対応状況については、WebOTX の Web サイトをご覧ください。

WebOTX V7 がサポートする OS と J2SE SDK のバージョンは次の表のとおりです。

OS	J2SE 1.4.2	J2SE 5.0	J2SE 6.0
Windows 2000 Server	○	○	○
Windows Server 2003		○	
Windows Server 2003(IPF 版)	○	×	×
Windows Server 2003 x64 Edition	×	○ (Update 4 以降)	○
HP-UX 11i v1	○ (1.4.2.05 以降)	○	×
HP-UX 11i v2	○ (1.4.2.05 以降)	○ (5.0.02 以降)	×
Solaris 9 Solaris 10	○	○	○
RHEL AS/ES 4.0	○	○	○

その他のバージョンの対応状況については、WebOTX の Web サイトをご覧ください。

WebOTX V8 がサポートする OS と J2SE SDK のバージョンは次の表のとおりです。

OS	J2SE 5.0	J2SE 6.0
Windows 2000 Server Windows Server 2003	○	○
Windows Server 2003(IPF 版)	×	×
Windows Server 2003 x64 Edition	○ (Update 4 以降)	○
HP-UX 11i v1	○	×
HP-UX 11i v2	○ (5.0.02 以降)	×
Solaris 9 Solaris 10	○	○
RHEL AS/ES4.0 MIRACLE LINUX V4.0	○	○

その他のバージョンの対応状況については、WebOTX の Web サイトをご覧ください。

1.1.2. Web サーバ

WebOTX がサポートする外部 Web サーバは次の表のとおりです。

Web サーバ	バージョン
WebOTX Web サーバ	1.3.x、2.0.x ※詳細は下表参照
Apache HTTP Server	1.3.x、2.0.x ※詳細は下表参照
Internet Information Server (IIS)	5.0、6.0、7.0
Sun Java System Web Server 6.1	6.1
Sun ONE Web Server	6.0 ※HP-UX 11i v2 は未サポート

WebOTX の各バージョンでサポートする WebOTX Web サーバ、Apache HTTP Server の各バージョンは次のとおりです。

WebOTX	WebOTX Web サーバ、Apache HTTP Server バージョン
WebOTX V6.1	1.3.31 以降、2.0.52 以降
WebOTX V6.2	1.3.33 以降、2.0.54 以降
WebOTX V6.3	1.3.34 以降、2.0.55 以降
WebOTX V6.4	1.3.36 以降、2.0.58 以降
WebOTX V7.1	1.3.37 以降、2.0.59 以降
WebOTX V8.1	1.3.41 以降、2.0.63 以降

1.1.3. データベース

WebOTX のサポート対象となるデータベースは、次の表のとおりです。また、JDBC データソース、Transaction サービス(JTA)をご

ールする必要があります。

その他の製品についても、JDBC 2.0 または、JDBC 3.0 の仕様に準拠している JDBC ドライバであれば、使用することができます。

WebOTX V6.5 のサポート対象のデータベース

JDBC ベンダ	JDBCドライバ・タイプ	サポートするデータベース・サーバ	備考
Oracle	Type 2、4	Oracle8i R8.1.6 (JTA 連携を行う場合は、Oracle R8.1.7 以降)	(*1)
		Oracle R8.1.7	
		Oracle9i Database Release 1 (9.0.1)	(*2)
		Oracle9i Database Release 2 (9.2.0)	
		Oracle Database 10g Release 1 (10.1.0)	
		Oracle Database 10g Release 2 (10.2.0)	
IBM	Type 4	DB2 Universal Database 8.1.4	
Microsoft	Type 4	Microsoft SQL Server 2000	
		Microsoft SQL Server 2005	
Sybase	Type 4	Sybase Adaptive Server Enterprise 12.5	
DataDirect	Type 4	「Connect for JDBC 3.3 以降」経由による Oracle 接続	
	Type 3	「SequeLink for JDBC 5.0」経由による Oracle 接続	
PostgreSQL Development Group	Type 4	PostgreSQL 7.3.2 (JDBC ドライバ 7.3-113) ～ 8.1.2 (JDBC ドライバ 8.1-404) (JTA 連携を行う場合は、バージョン 8.1.0 以降)	(*3)
Cloudscape	Type 4	Cloudscape 3.0.3 (Sun J2EE 1.3.1 SDK にバンドルされるもの)	

(*1) Oracle8i R8.1.6 を使用して 2 フェーズコミットを行う場合には、JDBC データソースのデータソースタイプに「JDBC」を指定して JDBC データソースの JTA 機能を利用してください。 または、JDBC ドライバとして SequeLink を使用してください。

(*2) Oracle の Real Application Cluster(RAC)と X/Open XA の機能を利用して 2 フェーズコミットを行うためには、必ず次のパッチを適用してください。
- PSR 10.1.0.3
- PSR 9.2.0.7

パッチの詳細については Oracle 社の情報をご参照ください。

(*3) PostgreSQL を使用して 2 フェーズコミットを行う場合には、PostgreSQL 8.1 と、バージョン 8.1-404 の JDBC ドライバを使用してください。

WebOTX V7.1 のサポート対象のデータベース

JDBC ベンダ	JDBCドライバ・タイプ	サポートするデータベース・サーバ	備考
Oracle	Type 2、4	Oracle R8.1.7	
		Oracle9i Database Release 1 (9.0.1)	(*1)
		Oracle9i Database Release 2 (9.2.0)	
		Oracle Database 10g Release 1 (10.1.0)	
		Oracle Database 10g Release 2 (10.2.0)	
IBM	Type 4	DB2 Universal Database 8.1.4	
		DB2 V9.1	
Microsoft	Type 4	Microsoft SQL Server 2000	
		Microsoft SQL Server 2005	
Sybase	Type 4	Sybase Adaptive Server Enterprise 12.5	
DataDirect	Type 4	「Connect for JDBC 3.3 以降」経由による Oracle 接続	
	Type 3	「SequeLink for JDBC 5.0」経由による Oracle 接続	
PostgreSQL Development Group	Type 4	PostgreSQL 7.3.2 (JDBC ドライバ 7.3-113) ～ 8.1.2 (JDBC ドライバ 8.1-404) (JTA 連携を行う場合は、バージョン 8.1.0 以降)	(*2)
Cloudscape	Type 4	Cloudscape 3.0.3 (Sun J2EE 1.3.1 SDK にバンドルされるもの)	
Apache Derby	Type 4	Apache Derby 10.2.2.0	

(*1) Oracle の Real Application Cluster(RAC)と X/Open XA の機能を利用して 2 フェーズコミットを行うためには、必ず次のパッチを適用してください。
- PSR 10.1.0.3
- PSR 9.2.0.7

パッチの詳細については Oracle 社の情報をご参照ください。

(*2) PostgreSQL を使用して 2 フェーズコミットを行う場合には、PostgreSQL 8.1 と、バージョン 8.1-404 の JDBC ドライバを使用してください。

JDBC ベンダ	JDBCドライバ・タイプ	サポートするデータベース・サーバ	備考
Oracle	Type 2、4	Oracle9i Database Release 1 (9.0.1)	(*1)
		Oracle9i Database Release 2 (9.2.0)	
		Oracle Database 10g Release 1 (10.1.0)	
		Oracle Database 10g Release 2 (10.2.0)	
		Oracle Database 11g Release 1 (11.1.0)	
IBM	Type 4	DB2 Universal Database 8.1.4	
		DB2 V9.1	
Microsoft	Type 4	Microsoft SQL Server 2000	
		Microsoft SQL Server 2005	
Sybase	Type 4	Sybase Adaptive Server Enterprise 12.5	
DataDirect	Type 4	「Connect for JDBC 3.3 以降」経由による Oracle 接続	
	Type 3	「SequeLink for JDBC 5.0」経由による Oracle 接続	
PostgreSQL Development Group	Type 4	PostgreSQL 7.3.2 (JDBC ドライバ 7.3-113) ～ 8.3.1 (JDBC ドライバ 8.3-603)	(*2)
Cloudscape	Type 4	Cloudscape 3.0.3 (Sun J2EE 1.3.1 SDK にバンドルされるもの)	
Apache Derby	Type 4	Apache Derby 10.2.2.0	

(*1) Oracle の Real Application Cluster(RAC)と X/Open XA の機能を利用して 2 フェーズコミットを行うためには、必ず次のパッチを適用してください。
- PSR 10.1.0.3
- PSR 9.2.0.7

パッチの詳細については Oracle 社の情報をご参照ください。

(*2) PostgreSQL を使用して 2 フェーズコミットを行う場合には、PostgreSQL 8.1 と、バージョン 8.1-404 の JDBC ドライバを使用してください。

1.1.4. 同梱するライブラリ

WebOTX には、次のライブラリが同梱されています。

WebOTX に同梱しているライブラリとバージョン

名称	V6.1	V6.22	V6.31	V6.4	V6.5.02	V7.1	V8.12
Java Beans Activation Framework (JAF)	1.0.2	1.0.2	1.0.2	1.0.2	1.0.2	1.0.2	1.0.2
JavaMail	1.3.1	1.3.1	1.3.1	1.3.1	1.3.1	1.3.1	1.3.1
JAXP	1.2.5	1.2.5	1.2.5	1.2.5	1.2.5	1.3.04	1.3.04
Ant	1.5.4	1.5.4	1.5.4	1.5.4	1.5.4	1.6.5	1.7.0
Jakarta Commons BeanUtiles	1.6.1	1.6.1	1.7.0	1.7.0	1.7.0	1.7.0	1.7.0
Jakarta Commons Codec	1.2	1.3	1.3	1.3	1.3	1.3	1.3
Jakarta Commons Collections	2.11	2.11	2.11	2.11	2.11	2.11	2.11
Jakarta Commons Digester	1.5	1.6	1.7	1.7	1.8	1.8	1.8
Jakarta Commons Discovery	0.2	0.2	0.2	0.2	0.4	0.4	0.4
Jakarta Commons EL	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Jakarta Commons FileUpload	1.0	1.0	同梱なし	同梱なし	同梱なし	同梱なし	同梱なし
Jakarta Commons Launcher	1.0-dev	1.1	1.1	1.1	1.1	1.1	1.1
Jakarta Commons Logging	1.0.4	1.0.4	1.0.4	1.0.4	1.1	1.1	1.1.1
Jakarta Commons Modeler	1.1	1.1	1.1	1.1	2.0	2.0	2.0
Jakarta Regexp	1.3	1.3	1.4	1.4	1.4	1.5	1.5
Log4j	1.2.8	1.2.8	1.2.13	1.2.13	1.2.14	1.2.14	1.2.15
XML Xerces 2 Java Parser	2.5.0	2.5.0	2.5.0	2.5.0	2.5.0	2.9.0	2.9.1
XML Xalan Java 2	2.5.2	2.5.2	2.5.2	2.5.2	2.5.2	2.7.0	2.7.1
The Web Services Description Language for Java Toolkit (WSDL4J)	1.4	1.4	1.4	1.4	1.4	1.4	1.4

1.1.5. クラスのロードについて

Web アプリケーションで利用する Java のクラスを実行するためにロードを行う機能としてクラスローダがあります。クラスローダには階層があります。上位のクラスローダでロードされたクラスからは下位のクラスローダでロードされたクラスは参照することができません。逆に、下位のクラスローダでロードされたクラスからは、上位のクラスローダでロードされたクラスは参照することができます。

WebOTX のクラスローダの階層は次のようになります。

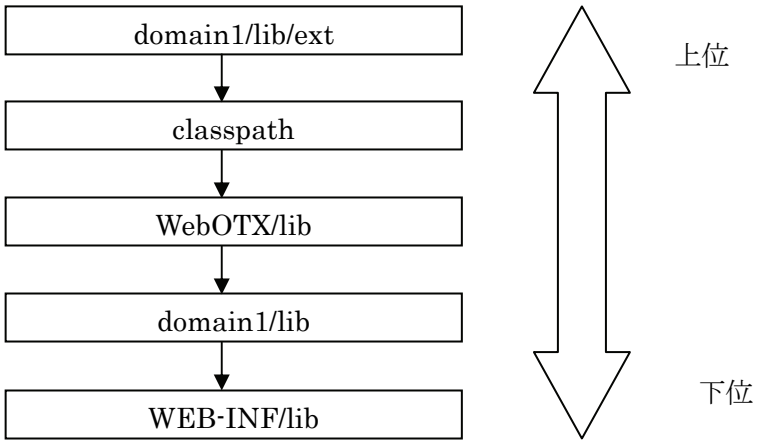


図 1 WebOTX のクラスローダの階層

また、nec-web.xml の delegate の設定により、ロードする優先順位を決定することができます。同じ名前のライブラリがあると、競合が発生し、優先順位の高いパスに含まれるのライブラリが利用されます。ロードされるライブラリの優先順位は delegate=true の場合は以下のようになります。delegate の設定は WebAP の WEB-INF/nec-web.xml で設定します。

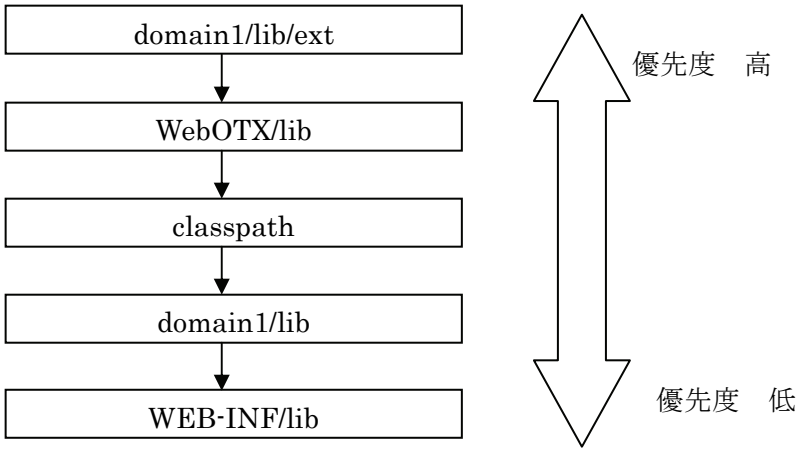


図 2 delegate=true の場合の優先順位

delegate=false の場合は以下のようになります。

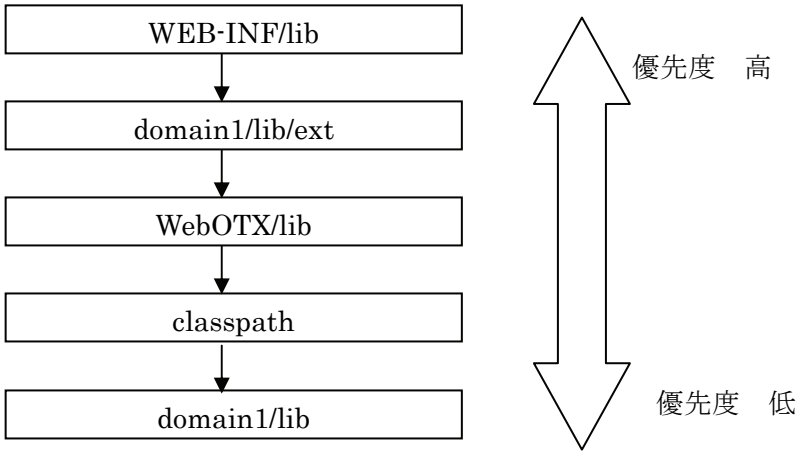


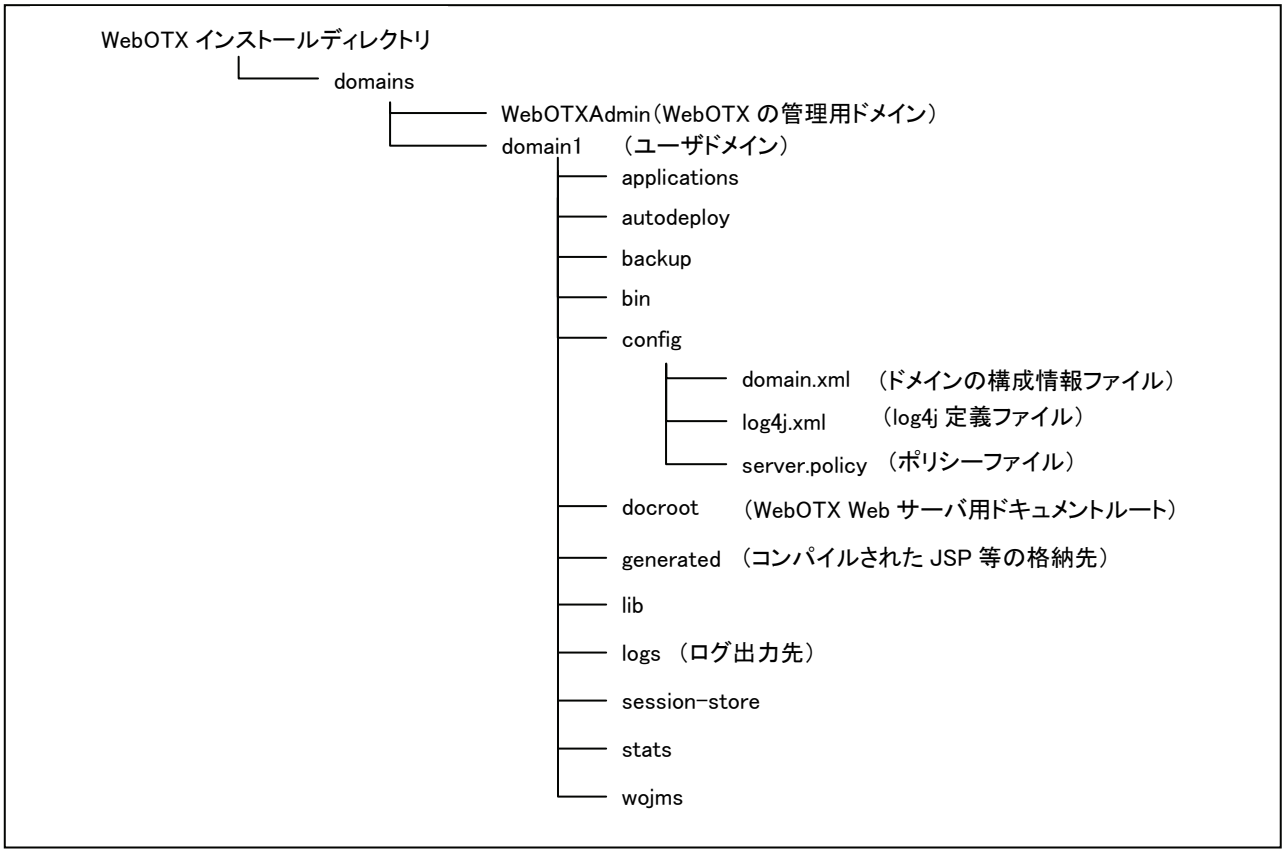
図 3 delegate=false の場合の優先順位

2.移行作業

本章では、Tomcat から WebOTX に移行する際に必要となる作業について説明します。

2.1. ディレクトリ構成

WebOTX のドメインに関連する主要なディレクトリは次のようになっています。



2.2. 移行作業一覧

Tomcat の各バージョンから WebOTX に移行するための作業と、本書での説明の対応は以下のとおりです。

○：作業要　△：条件により作業要

移行作業	Tomcat 3.x	Tomcat 4.x	Tomcat 5.0.	Tomcat 5.5	Tomcat 6.x	本書での説明
パッケージの設定	△	△	△	△	△	2.3 章
JDBC データソースの設定	△	△	△	△	△	2.4章
log4j の設定	△	△	△	△	△	2.5章
nec-web.xml の設定	○	○	△	△	△	2.6 章
セキュリティポリシーの設定	○	○	○	○	○	2.7章
Tomcat の差異による対応	×	×	×	△	△	2.8 章
JavaVM オプションの設定	△	△	△	△	△	4.2 章

パッケージの設定

既存の JSP コードにおいて、他のクラスをインポート(import)している場合にはパッケージを指定しなければなりません。これは JDK 1.4 から Javac コンパイラが Java 言語仕様へ厳密にチェックするように変わったことに起因します。J2SE 1.4.0 より前のバージョンでは、パッケージの指定は必要ありませんでしたが、J2SE 1.4.0 以降ではパッケージの指定が必要になっています。

JDBC データソースの設定

Tomcat では、「conf/server.xml」ファイルを直接修正することで JDBC データソースの設定を行いますが、WebOTX では、JMX (Java Management Extensions) に対応した統合運用管理ツール/Web 版統合運用管理コンソールや運用管理コマンドで設定します。

WebOTX では、データベース・サーバへの接続をファクトリ化し、JNDI API 経由で接続オブジェクト参照を得る仕組みとして JDBC データソースを提供しています。JDBC データソースは、分散トランザクションやコネクション・プーリングなどを考慮しているため、単に JDBC ドライバのインタフェースを呼び出す場合よりも機能拡張されています。

既存の Web アプリケーションにおいて JDBC データソースを使用している場合、JDBC データソースを利用するための設定を行う必要があります。

log4j の設定

WebOTX は、Apache Logging Server プロジェクトの log4j をパッケージ名を変えてバンドルしており WebOTX 自身のロギング機能に log4j を利用しています。このため、既存の Web アプリケーションで log4j を利用してログ出力している場合、WebOTX の定義を気にする事なく利用可能です。

nec-web.xml の設定

既存の Web アプリケーションにおいて、Struts などのライブラリ JAR ファイルや、WebOTX 以外の CORBA ライブラリを WAR ファイル内の「WEB-INF/lib」配下に格納している場合、また、CORBA ライブラリについては、WebOTX に含まれるライブラリと重なる場合、Web アプリケーションを配備できないなどの問題が発生することがあります。Web アプリケーションを、そのままのファイル構成でご利用いただくには、nec-web.xml でライブラリのロードに関する設定を行います。

セキュリティポリシーの設定

WebOTX は、インストール・デフォルトの定義により Java セキュリティ・マネージャが動作します。Java セキュリティ・マネージャが働くことで、Web アプリケーションの実行中に、システム上のファイルや他システムへの接続などの資源に対する不用意なアクセスを制限できます。このため、現行システムでセキュリティポリシーを設定していない場合は、WebOTX 移行に際してはセキュリティポリシーの設定が必要となります。 また、現行システムでセキュリティポリシーを設定している場合は、既存のセキュリティポリシー設定を移行する必要があります。

Tomcat の差異による対応

Tomcat の各バージョンの差異により WebOTX に移行する際に対応が必要となる場合があります。

JavaVM オプションの設定

WebOTX は Tomcat と同様に JavaVM のオプション設定をサポートしています。このため、既存の Tomcat システムで JavaVM オプションの設定を行っている場合は WebOTX にも同様の設定を行う必要があります。

詳細は 4.2 章 Java VM オプションのサポートを参照ください。

2.3. パッケージの設定

JSP で他のクラスをインポート（import）している場合には、パッケージの指定が必要です。

これまで Java2 SDK 1.3.1 以前を使用していた場合は、その Javac コンパイラの構文チェックが緩かったために問題とはなりませんでした。しかし、J2SE 1.4.0 からは構文チェックが厳しくなり、パッケージを持たない JSP コードは、JSP を WebOTX に配備した後に行われる JSP コンパイルでエラーとなります。なお、WebOTX V8 は、JDK1.5 か 1.6 のみをサポートします。

J2SE 1.3.1 以前のバージョンから移行する場合には、import のパッケージの指定を確認してください。

次のようにパッケージの指定のない import はエラーとなります。

```
<%@ page import="Converter,ConverterHome" %>
```

import するクラスにパッケージの指定を追加して、import の指定をそれにあわせて修正してください。

2.4. JDBC データソースの設定

WebOTX の JDBC データソースを利用するためには、WebOTX 運用環境製品で提供している「統合運用管理ツール」か、標準で備わる「運用管理コマンド（otxadmin)」を利用します。ここでは、運用管理コマンドを利用する設定について説明します。

設定内容の詳細や運用機能の詳細は、WebOTX マニュアルの中の「運用管理コマンドリファレンスマニュアル」ー「1. 運用管理エージェント運用管理コマンド」、「運用編（コンフィグレーション）」をご参照ください。

2.4.1. JDBC データソースの設定方法

運用管理コマンドより JDBC データソースを JNDI に登録します。

以下は、WebOTX をインストールした際にデフォルトで作成されるドメイン「domain1」上で Oracle データベース・サーバを利用する JDBC データソースを登録するコマンド例です。

```
otxadmin> create-jdbc-datasource --user admin --password adminadmin
        --port 6212 --host localhost
        --dataSourceType JDBCEX_Oracle
        --jdbcMajorVersion 3 --maxPoolSize 10
        --jdbcUserName scott --jdbcPassword tiger
        --dataSourceName "jdbc:oracle:thin:@hostname:1521:ORCL"
        jdbc/MyOracle
```

※ 実際には 1 行で実行してください。

※ 各引数の詳細は、WebOTX マニュアルの「運用管理コマンドリファレンスマニュアル」ー「1. 運用管理エージェント運用管理コマンド」の "create-jdbc-datasource" を参照してください。

※ *"jdbc:oracle:thin:@hostname:1521:ORCL"* と、*jdbc/MyOracle* の部分は環境に合わせて変更してください。

登録が成功すると次のように表示されます。

```
> Command create-jdbc-datasource executed successfully.
```

現在の設定を確認する場合は、次のコマンドを実行します。

```
otxadmin> list-jdbc-datasources --user admin --password adminadmin
        --port 6212
```

2.4.2. クラスパスの設定

Oracle データベース・サーバに付属する JDBC ドライバ（ojdbc14.jar）が配置されたディレクトリにクラスパスを設定します。

以下は domain1 ドメインにクラスパスを設定するコマンド例です。

現状のクラスパスを確認します。

```
otxadmin> get --user admin --password adminadmin
            --port 6212 --host localhost
            server.java-config.classpath-suffix
```

※ 実際には 1 行で実行してください。

現状のクラスパスに JDBC ドライバのクラスパスを追加設定します。

```
otxadmin> set --user admin --password adminadmin
            --port 6212 --host localhost
            server.java-config.classpath-suffix="…;/temp/ojdbc14.jar"
```

※ *"/temp/ojdbc14.jar"* 部分は環境に合わせて変更してください。

※ 実際には1行で実行してください。

※ set コマンドは指定の値で上書きしますので、既存の設定値を含めて設定してください。

設定が完了したらドメインを再起動してください。

2.5. log4j の設定

Tomcat では、Web アプリケーション毎にログ出力方法を制御する場合は、各 Web アプリケーションの WEB-INF/classes 以下に log4j.properties または log4j.xml という名前で設定ファイルを配置します。そのほかに、システム全体で設定ファイルを指定する方法もあります。この場合、Tomcat3.x ならば TOMCAT_OPTS 、Tomcat4.x 以降ならば CATALINA_OPTS 環境変数へ指定します。

以下は、Tomcat4.x で、システム全体で log4j の設定を行っている場合の移行の例です。WEB-INF/classes 配下に log4j の定義ファイルを含めている場合はこの設定は不要です。

【log4.configuration を使用した設定例（tomcat4.x の例）】

環境変数への設定（unix の場合）

```
export CATALINA_OPTS="-Dlog4j.configuration=file:///home/foo/log4j.xml"
```

【運用管理コマンドでの設定(WebOTX の例)】

```
otxadmin > create-jvm-options --user admin --password adminadmin --port
6212 --host localhost "-Dlog4j.configuration=
file¥:///${com.nec.webotx.instanceRoot}${file.separator}config ${file.separator}.xml"
```

Web アプリケーションで log4j API を呼び出して、利用する log4j 定義ファイルを読み込む方法についての説明は、ここでは省略します。

前述の通り、WebOTX V8.1 ではログ出力に WebOTX 独自の logging モジュールを利用しています。よって、ユーザアプリケーションでは、WEB-INF/classes 配下に定義ファイルを置くか、システムプロパティに設定するかのどちらかで log4j を利用することが可能です。

2.6. 追加になった設定ファイル(nec-web.xml)の設定

Web アプリケーションには、Servlet 仕様にしたがって配備記述子 (Deployment Descriptor) と呼ばれる実行の振る舞いを宣言する XML 形式の設定ファイル (web.xml) を付属させます。WebOTX V8 では、その標準の Servlet 仕様に規定された web.xml 要素を拡張した、nec-web.xml を追加しています。

ライブラリのロードの設定を変更したい場合や、BASIC/FORM 認証をする際のユーザ名や権限の設定を追加したい場合は、nec-web.xml に該当する情報を設定しなければなりません。

2.6.1. BASIC 認証、FORM ベース認証を利用する場合の設定

WebOTX では、BASIC 認証か FORM ベース認証を利用する場合、次の箇所を設定します。

- 使用するレルムにユーザ名、パスワード、権限を登録します。
- 必要に応じて login.conf、domain.xml ファイルに使用するレルムを登録します。
- web.xml ファイルでレルムおよびロールを指定します。
- nec-web.xml で、web.xml で指定したロール名に対応するユーザ名(principal)および権限(group)を指定します。

【web.xml の記述例(抜粋)】

```
<security-constraint>
   <display-name>Server Configuration Security Constraint</display-name>
      <web-resource-collection>
         <web-resource-name>Protected Area</web-resource-name>
         <!-- Define the context-relative URL(s) to be protected -->
         <url-pattern>*.jsp</url-pattern>
         <url-pattern>*.do</url-pattern>
         <url-pattern>*.html</url-pattern>
         <url-pattern>/WebAPManage/*</url-pattern>
      </web-resource-collection>
      <auth-constraint>
        <!-- Anyone with one of the listed roles may access this area -->
         <role-name>stRole</role-name>
      </auth-constraint>
</security-constraint>


<!-- Login configuration uses form-based authentication -->
```

```
<login-config>
    <auth-method>FORM</auth-method>
      <realm-name>file</realm-name>

      <form-login-config>
          <form-login-page>/login.jsp</form-login-page>
           <form-error-page>/error.jsp</form-error-page>
      </form-login-config>
  </login-config>

  <!-- Security roles referenced by this web application -->
  <security-role>
      <description>
           The role that is required to log in to the Administration Application
      </description>
      <role-name>stRole</role-name>
  </security-role>
```

nec-web.xml のロールを指定します

【nec-web.xml の記述例(抜粋)】

```
<context-root>BasicAuthServlet24</context-root>

<security-role-mapping>
  <role-name>stRole</role-name>
  <principal-name>admin3</principal-name>
  <group-name>stGroup</group-name>
</security-role-mapping>
```

web.xml で設定したロール名

ログインを許可するユーザ名を指定します。

ログインを許可するグループ名を指定します
※データベースのロールテーブルに登録するロール名に相当します。

※　ロール名などを nec-web.xml に指定する方法は、「WebOTX 運用(コンフィグレーション)マニュアル」ー「6.3.2 Web アプリケーションの設定」をご覧ください。

2.6.2. クラスロード優先順位の設定

nec-web.xml に”delegate=false”の設定をすることにより、クラスロードの優先順位の変更を行うことができます。

nec-web.xml ファイルの格納場所およびファイルの内容は次のとおりです。

${INSTANCE_ROOT}/applications/j2ee-modules/Web アプリケーション毎のディレクトリ/WEB-INF

【nec-web.xml】

```
<?xml version="1.0" encoding="UTF-8"?>
<nec-web-app xmlns="http://java.sun.com/xml/ns/j2ee">
  <context-root>context_name</context-root>
                                ・・・Web アプリケーションのコンテキスト名
  <class-loader delegate="false"/>
                                ・・・委譲モデル（ロード要求を受けた場合、まずは要求を親の
                                クラスローダに委譲）を使用しない場合は、false を指定
</nec-web-app>
```

※ nec-web.xml はアプリケーション配備時にアプリケーションディレクトリ内（"<ドメインディレクトリ>/applications/j2ee-module/<アプケーション名>/WEB-INF" 配下）に自動的には生成されません。"<ドメインディレクトリ>/generated/xml/<アプリケーション名>/WEB-INF" 配下に生成されますので、必要に応じてコピーして使用してください。nec-web.xml が含まれない場合、Servlet のバージョンに関わらず delegate のデフォルト値は常に true になります。

nec-web.xml を変更した場合は、変更した nec-web.xml を、Web アプリケーションの WEB-INF 配下に再配置します。

WEB-INF 下に配置後、次のいずれかの方法で配備を行ってください。

・WAR ファイルを作成して配備

・ディレクトリ指定で配備

配備については、「3.4 章 配備方法」をご覧ください。

2.6.3. 利用する場合に設定が必要なパッケージ

Web アプリケーション内に配置したクラスのうちパッケージ名が以下で始まるクラスは参照することができません。この仕様は Servlet 仕様での勧告（javax などの J2SE、J2EE クラスを WAR 内のライブラリでオーバライドさせるべきではない）に基づくものです。詳細は Servlet 2.4 の 9.7.2 節を参照してください。

それにより、WEB-INF/lib 配下のこれらのファイルを読み込む場合は以下の設定が必要になります。

対象パッケージ名

"javax"
"sun"
"org.xml.sax"
"org.w3c.dom"
"org.apache.xerces"
"org.apache.xalan"
"org.apache.taglibs.standard"
"com.sun.faces"

【nec-web.xml での設定方法】
<nec-web-app>タグの"property"に"overrideablejavaxpackages"を追加します。複数設定する場合はカンマで区切って指定します。反映させるには Web アプリケーションを更新して、更新したアプリケーションを配備し直してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<nec-web-app>
```

```
    <context-root>context_name</context-root>
    <class-loader delegate="true">
      <property name="overrideablejavaxpackages" value="javax"/>
    </class-loader>
  </nec-web-app>
```

【運用管理コンソールでの設定方法】

JavaVM オプションに"com.nec.webotx.enterprise.overrideaclejavaxpackages"の設定を行います。詳細は 4.2. JavaVM オプションのサポートを参照してください。

※ JAXB2.0 を利用しているアプリケーションを動作させるためにもこの設定が必要となります。以下の例を参考に nec-web.xml を追加してください

【nec-web.xml】

```
<?xml version="1.0" encoding="UTF-8"?>
<nec-web-app>
  <context-root>context_name</context-root>
  <class-loader delegate="false">
    <property name="overrideablejavaxpackages" value="javax,sun,org.
apache" />
  </class-loader>
 </nec-web-app>
```

2.7. セキュリティポリシーの設定

Tomcat は、オプションなしで起動すると、Java セキュリティ・マネージャを使用しない定義で動作します。セキュリティ・マネージャを使用しない場合、実行時にアクセス制御を行わないため、高速に処理できます。その反面、社会インフラでの利用に求められるサーバ・セキュリティが全く働かない問題があります。

WebOTX では、デフォルトでセキュリティ・マネージャが動作します。これにより、Web アプリケーションが不用意に資源（OS 上のファイルや他システムへの接続など）にアクセスすることを制限することができます。ただし、意図した資源へのアクセスを許可する場合には、Java セキュリティポリシーに追加設定する必要があります。

ここでは、Web アプリケーションが必要とする資源へのアクセスを許可させる方法について説明します。

2.7.1. セキュリティポリシーの設定方法

Web アプリケーションが、任意のディレクトリ/ファイルにアクセスする場合の設定を例に説明します。

Web アプリケーションが、「/Temp」ディレクトリとその配下のファイルの読み込みを行う場合、次のファイルに定義を追加します。

【${INSTANCE_ROOT }/config/server.policy】

```
...
// Basic set of required permissions granted to all remaining code
grant {
      permission java.io.FilePermission "${/}Temp", "read";
      permission java.io.FilePermission "${/}Temp${/}-", "read";
      ...
}
...
```

上記では、「/Temp」ディレクトリへの read の許可と、「/Temp」ディレクトリ配下のファイルの read の許可を追加しています。

設定後は、ドメインを再起動することで反映されます。

JDK のアクセス権とポリシーの構文については以下のサイトを参照してください。(J2SE 1.5 の説明)

http://java.sun.com/j2se/1.5.0/docs/guide/security/permissions.html
http://java.sun.com/j2se/1.5.0/docs/guide/security/PolicyFiles.html

2.7.2. セキュリティ例外の調査方法

事前にアクセスするディレクトリ/ファイルなどが分かっている場合には、動作させる前に server.policy に定義を追加できます。しかし、どのようなアクセスがあるか分からない場合には、Web アプリケーションを動作させてみて、セキュリティ例外メッセージから追跡します。

例えば、許可されない資源へのアクセスがあると、${INSTANCE_ROOT}/logs/server.log ログファイルに次のような例外メッセージが記録されます。

```
access denied (java.io.FilePermission /Temp read)
```

この場合には、先に説明した「/Temp」ディレクトリへの read を許可するポリシーの追加が必要になります。

さらに、次のように Java VM のオプションを追加することによって、詳細な情報を確認することができます。

以下は domain1 ドメインにセキュリティ違反の詳細を確認する Java VM オプションを追加するコマンド例です。

```
otxadmin> create-jvm-options --user admin --password adminadmin
        --port 6212 --host localhost
        -Djava.security.debug=access¥¥:failure:
```

※ 実際には 1 行で実行してください。

設定後、ドメインを再起動してください。詳細な情報が logs/server.log に出力されますので、「denied」と記録されているものを確認し、セキュリティポリシーの追加を行ってください。

2.7.3. セキュリティポリシー設定の移行

Tomcat でのセキュリティポリシーの設定は conf/catalina.policy で行います。 WebOTX でのセキュリティポリシーの設定は ${INSTANCE_ROOT }/config/server.policy で行います。 現行のシステムでセキュリティポリシーを設定している場合は、この設定内容を移行します。

【conf/catalina.policy 抜粋 （Tomcat の例）】

```
grant {
    permission java.util.PropertyPermission "java.home", "read";
    :省略
};
```

【${INSTANCE_ROOT }/config/server.policy 抜粋 （WebOTX の例）】

```
grant principal javax.management.remote.JMXPrincipal "admin" {
    :省略
    permission java.io.FilePermission "${java.home}${/}..${/}bin${/}-",
         "read,execute";
    :省略
};
```

2.8. Tomcat の差異による対応

Tomcat の各バージョンの差異により、WebOTX への移行の際に対応が必要となる場合があります。

ここでは、各バージョンの差異とその対応方法について説明します。

2.8.1. Tomcat 5.0 と Tomcat 5.5 の差異

・server.web-container.property.forwardRequestURL の値が変更されている

Tomcat 5.0 以前から WebOTX V8 へ移行する際にこの対応が必要になる場合があります。

Tomcat 5.0 と Tomcat 5.5 の差異として、この値のデフォルト値が “original” → “forward” に変更されています。

WebOTX V7 → V8 の変更は Tomcat 5.0 → 5.5 の変更に合わせているため、対応が必要な場合は設定を行ってください。

設定方法の詳細は 4.1 Tomcat 設定項目との対応 をご覧ください。

2.8.2. Tomcat 5.5 と Tomcat 6.0 の差異

・invokerServlet が機能しない

Tomcat 5.5 以前から WebOTX V8 に移行する際にこの対応が必要になる場合があります。

これは Tomcat 5.5 で機能していた invokerServlet の機能が Tomcat 6.0 ではデフォルトで動作しないようになっているため、Tomcat6.0 をベースにしている WebOTX V8 ではこの問題が発生します。

この機能を使用する場合は下記の設定が必要になります。

【設定手順】

1) Web アプリケーションの web.xml の invokeServlet に load-on-startup が指定されている場合は、これを削除します。
   ※配備時には Web アプリケーションに特権が与えられていないため、この指定があると起動に失敗し、配備ができません。
2) Web アプリケーションの設定で特権(privileged)を与えます。
   設定方法の詳細は 4.1 Tomcat 設定項目との対応 をご覧ください。
3) ドメインを再起動します。

※privileged を true に設定すると、Web アプリケーションが内部クラスにアクセスできるようになります。セキュリティの観点から使用には注意してください。

3.運用管理方法

Tomcat での運用管理では、Web ブラウザから Administration Tool や Manager アプリケーションを利用し、設定の変更は server.xml を直接編集するのが一般的です。

WebOTX では、JMX(Java Management Extensions)による運用管理機能を提供しています。これにより、Tomcat の server.xml に相当する設定項目をコマンドや GUI により容易に運用管理できるようになっています。

運用管理において、Tomcat と WebOTX は次の違いがあります。

項目	Tomcat	WebOTX	本書での説明
Web サーバとの連携	設定ファイルを直接編集	環境設定ツールを提供	3.1章
配備方法	Manager アプリケーションを使うか、webapps ディレクトリへ WAR ファイルをコピー	運用管理コマンド、運用管理コンソールを提供し、autodeploy ディレクトリへのファイルコピーでの配備にも対応	3.4 章
複数 Web コンテナの構成	Tomcat を複数インストール	複数のドメインを作成	-
運用管理コマンド	なし	運用管理コマンドを提供	3.2 章
運用管理コンソール	Administration Tool、Manager アプリケーションを提供	運用管理コンソールを提供	3.3 章

3.1. Web サーバとの連携

WebOTX では、各 Web サーバとの連携に必要なプラグイン(mod_jk)と、連携のための設定作業を支援する環境設定ツールも提供しています。

Web サーバとの連携を設定する場合には、環境設定ツールを利用して行います。複数のドメインが存在する場合、この設定はドメイン毎に行います。

【環境設定ツール】 Windows 版

「スタート」-「プログラム」-「WebOTX」の「環境設定ツール」を起動します。

【環境設定ツール】(シェル) UNIX 版

ログイン名 root でログインします。

```
login: root
```

WebOTX のインストールディレクトリ/bin ディレクトリへ移動します。

```
root> cd /opt/WebOTX/bin
```

./setconf.sh と入力し環境設定ツールを起動します。

```
root> ./setconf.sh
```

上記の環境設定を行った後、ドメインを起動して、次のように運用管理コマンド(otxadmin)で設定します。

```
otxadmin> set --user admin --password adminadmin
            --port 6212 --host localhost
            server.http-service.virtual-server.server.http-listeners="
            ajp-listener-1"
```

環境設定ツール、運用管理コマンドでの設定手順の詳細については、「WebOTX マニュアル セットアップガイド」をご覧ください。

3.2. 運用管理コマンド

WebOTX では、運用のための様々なコマンドを用意しています。

domain の起動/停止、構成情報の参照や変更、アプリケーションの配備/配備解除などが運用管理コマンドで可能です。

詳細は「WebOTX マニュアル 運用管理コマンドリファレンスマニュアル」ー「1.運用管理エージェント運用管理コマンド」をご覧ください。

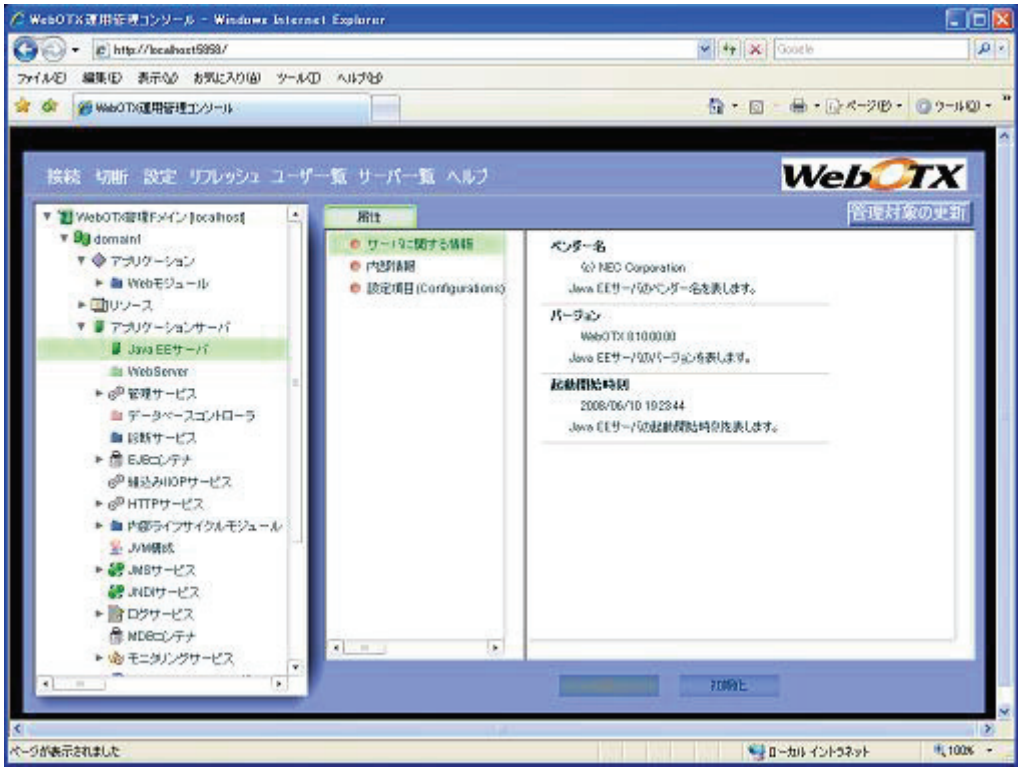
3.3. Web 版運用管理コンソール

Web ブラウザから利用できる Web 版統合運用管理コンソールを用意しています。

詳細は「WebOTX マニュアル WebOTX 運用管理ツールガイド Web 版統合運用管理コンソール」、もしくは Web 版統合運用管理コンソールのヘルプをご覧ください。

Web 版統合運用管理コンソールを利用するには、クライアントに Web ブラウザと Macromedia Flash Player 7 以上(Macromedia Flash Player 8 推奨)が必要です。

【Web 版統合運用管理コンソール】



3.4. 配備方法

WAR ファイルを配備するには、運用管理コマンドを利用する方法と統合運用管理ツール/Web 版統合運用管理コンソールを利用する方法、autodeploy ディレクトリにコピーする方法があります。

統合運用管理ツール/Web 版統合運用管理コンソールによる配備の方法は以下のマニュアルを参照ください。

WebOTX V8 マニュアル「運用編」－「統合運用管理ツール」－「3.運用と操作」－「アプリケーションの配備」

本節では、運用管理コマンドで配備する方法について、説明します。

WebOTX に Web アプリケーションを配備するには、WAR ファイルで配備する方法と、WebOTX が動作するサーバ上の Web アプリケーションのディレクトリを指定して配備する方法があります

3.4.1. WAR ファイルでの配備

WebOTX に Web アプリケーションを配備するには、WAR ファイルで配備する方法と、WebOTX が動作するサーバ上の Web アプリケーションのディレクトリを指定して配備する方法があります。ここでは、WAR ファイルで配備する場合について説明します。

WAR ファイルの作成

Web アプリケーションのディレクトリを”D:¥temp¥sampleAP”とします。JDK のインストールディレクトリは” C:¥jdk1.5.0_16”とします。

Windows では、次のようにコマンドを実行して sampleAP.war ファイルを作成します。

```
cd D:¥temp¥sampleAP
C:¥jdk1.5.0_16¥bin¥jar -cfv sampleAP.war *
```

※WAR ファイルの構成については以下のマニュアルを参照ください。

「アプリケーション開発ガイド」－「プログラミング・開発」－「2 章 Java EE」－「2.2 Web アプリケーション」

運用管理コマンドでの配備

運用管理コマンドで WAR ファイルを配備するには次のようにコマンドを実行します。

自ホストでデフォルト作成した domain1 に sampleAP.war を配備する場合の例です。

```
otxadmin> deploy --user admin --password adminadmin
              --host localhost --port 6212  sampleAP.war
```

※ 実際には1行で実行してください。

※ 各引数の詳細は「WebOTX マニュアル 運用管理コマンドリファレンスマニュアル 1.運用管理エージェント運用管理コマンド」をご覧ください。

3.4.2. ディレクトリ指定での配備

運用管理コマンドでディレクトリ指定によって Web アプリケーションを配備するには次のようにコマンドを実行します。

自ホストでデフォルト作成した domain1 に/home/temp/sampleAP ディレクトリ配下に作成したアプリケーションを配備する場合の例です。

```
otxadmin> deploydir --user admin --password adminadmin
                    --host localhost --port 6212
                    /home/temp/sampleAP
```

※ 実際には1行で実行してください。

※ 各引数の詳細は「WebOTX マニュアル 運用管理コマンドリファレンスマニュアル 1.運用管理エージェント運用管理コマンド」をご覧ください。

4. 設定項目

本章では、WebOTX に設定することのできる項目について説明しています。

4.1. Tomcat 設定項目との対応

Tomcat 6.x の設定項目を WebOTX V8 に設定する方法をまとめました。

設定を反映させる方法は以下のようになります。

- domain.xml への設定を運用管理コンソールにより設定する場合（推奨）

  運用管理コンソールにて設定を行った後、ドメインを再起動してください。

- domain.xml への設定をコマンドにより設定する場合

  ドメインを起動した状態で、運用管理コマンドにて設定コマンドを入力した後、ドメインを再起動してください。

- domain.xml を直接編集する場合

  ドメイン停止後、記述箇所に設定内容を記述し、ドメインを起動してください。

- nec-web.xml を編集する場合

  Web アプリケーションを更新して、更新したアプリケーションを配備し直してください。

また、domain.xml と nec-web.xml の両方に設定箇所がある項目については、同時に設定した場合 nec-web.xml への設定が優先されます。

コネクタ（HTTP、AJP 共通）

項目	説明	設定箇所	設定方法
allowTrace	TRACE HTTP メソッドを有効にする設定です。 デフォルト値は false です。	domain.xml	■運用管理コンソールへの設定例： ＜ドメイン名＞ －「アプリケーションサーバ」 －「Web コンテナ」 の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「allowTrace=false」 ■運用管理コマンド例： otxadmin> set server.web-container.property.allowTrace=false ■domain.xml 設定例： web-container 要素に property を設定しま

			す。 <web-container> <property name="allowTrace" value="false" /> </web-container>
emptySessionPath	セッション、クッキーのパスについての設定を指定します。WebOTX では、Web アプリケーションごとにパスに設定する値を指定します。 cookiePath に"/"を指定することで、emptySessionPath=true と同等の動作をします。 ポートレットなど特殊な実装の場合に利用します。特に Web アプリケーションで必要がなければ設定不要です。 デフォルト値はコンテキストパスです。	nec-web.xml または domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「emptySessionPath=true」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.emptySessionPath=true ■nec-web.xml への設定例: <nec-web-app> <session-config> <cookie-properties> <property name="cookiePath" value="/" /> ■domain.xml への設定例: <http-listener > <property name="emptySessionPath" value="true" /> </http-listener>
enableLookups	true を指定すると、リモートクライアントのホスト名を IP アドレスから逆引きして、request.getRemoteHost()に返却します。false の場合は DNS の逆引きをスキップして IP アドレスを返却します。 デフォルト値は false です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名>

			の順にクリックしていき、「属性」タブの「一般」の「DNS ルックアップ有無」にチェックを行います。 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.enable-lookups=false ■domain.xml への設定例: http-lister 要素に設定します。 <http-service> <http-listener enable-lookups="false" /> </http-service>
maxPostSize	コンテナの FORM URL パラメタ構文解析で扱う POST の最大バイト数を指定します。この属性を 0 以下の値にセットすると無制限になります。 デフォルト値は 2097152 (2MByte)です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「maxPostSize=2097152」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.maxPostSize=2097152 ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener> <property name="maxPostSize" value="2097152" /> </http-listener>
maxSavePostSize	FORM または CLIENT-CERT 認証の間、コンテナが保存、バッファリングする POST の最大バイト数を指定します。どちらのタイプの認証に対しても, ユー	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」

	ザが認証される前にPOSTが保存、バッファリングされます。 CLIENT-CERT 認証に対しては、POSTは SSL ハンドシェークの間バッファリングされ、リクエストが処理されるとバッファが空にされます。 FORM 認証に対しては、POST はユーザが login フォームにリダイレクトされる間保存され、ユーザが認証に成功するか、認証リクエストに関するセッションが期限切れになるまで維持されます。この属性を -1 にすると無制限になります。属性を 0 にすると認証の間の POST データの保存が禁止されます。 デフォルト値は 4096 (4KByte)です。		－「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「maxSavePostSize=4096」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.maxSavePostSize=4096 ■domain.xml への設定例: web-container 要素または http-listener 要素に property を設定します。 <web-container> <property name="maxSavePostSize" value="4096" /> </web-container> または <http-listener> <property name="maxSavePostSize" value="4096" /> </http-listener>
protocol	通信プロトコルを指定します。HTTP/1.0、HTTP/1.1 または AJP/1.3 を指定します。 デフォルト値は HTTP/1.1 です。	domain.xml	■運用管理コンソールへの設定例: ＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「属性」タブの「一般」の「扱うプロトコル」に記述を行います。 ※この項目は設定変更できません ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.protocol=HTTP/1.1 ■domain.xml への設定例:

			http-listener 要素に設定します。 <http-service> <http-listener protocol="HTTP/1.1" /> </http-service>
proxyName	このコネクタがプロキシを利用する場合、そのプロキシのサーバ名を記述します。 デフォルト値はありません。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「proxyName=www.my company.com」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.proxyName=www.my company.com ■domain.xml への設定例: http-listener 要素に property を設定します <http-listener > <property name="proxyName" value="www.mycompany.com" /> </http-listener>
proxyPort	このコネクタがプロキシを利用する場合、そのプロキシのポート番号を記述します。 デフォルト値はありません。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「proxyPort=80」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.proxyPort=80

			■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener > <property name="proxyPort" value="80" /> </http-listener>
redirectPort	リダイレクトに使用するポート番号を指定します。もしもこのコネクタが非 SSL リクエストをサポートしており, 受け取ったリクエストの宛先に該当する <security-constraint> が SSL トランスポートを要求しているならば、Catalina は自動的にそのリクエストをここで指定されたポート番号へリダイレクトします。 デフォルト値はありません。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「redirectPort=443」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.redirectPort=443 ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener> <property name="redirectPort" value="443" /> </http-listener>
scheme	request.isSecure() 呼び出しで返されるセキュア情報を指定します。SSL を使用する場合には true を指定して下さい。 デフォルト値は false です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「sheme=https」 ■運用管理コマンド例:

			otxadmin> set server.http-service.http-listener.http-listener-1.property.sheme=https ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener > <property name="scheme" value="https" /> </http-listener>4
secure	request.getScheme() 呼び出しで返されるスキーマ文字列を定義します。 この値が true ならば https 通信が可能となります。 デフォルト値は false です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「属性」タブの「一般」の「SSL の使用の有無」にチェックを行います。 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.security-enabled=true ■domain.xml への設定例: http-listener 要素に設定します。 <http-service> <http-listener security-enabled="true"/> </http-service>
URIEncoding	「http://contents?a=%12%34」形式で記述されたURLをデコードする際の文字エンコーディングを指定します。 デフォルトは「ISO-8859-1」です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「uriEncoding=Shift_JIS」

			■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.uriEncoding=Shift_JIS ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener <property name="uriEncoding" value="Shift_JIS" </http-listener>
useBodyEncodingForURI	Tomcat4.1.x 系との互換性の為に使用されます。 デフォルトは false ですが、互換を保つには true を指定して下さい。 true にすると、ContentType 又は Request.setCharacterEncoding メソッドで指定されたエンコーディングをデコードに使用します。URIEncoding は使われません。 デフォルト値は false です。	domain.xml	■運用管理コンソールへの設定例: ＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「useBodyEncodingForURI=true」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.useBodyEncodingForURI=true ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener <property name="useBodyEncodingForURI" value="true"/> </http-listener>
xpoweredBy	Servlet 仕様で推奨されたヘッダを使って、その仕様をサポートしていることを広告します。 デフォルト値は false です。	domain.xml	■運用管理コンソールへの設定例: ＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」

			－ ＜リスナ名＞ の順にクリックしていき、「属性」タブの「一般」の「X-Powered-By ヘッダ付加の有無」にチェックを行います。 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.xpowered-by=true ■domain.xml への設定例: http-listener 要素に設定します。 <http-service> <http-listener xpowered-by="true" /> </http-service>
address	このリクエストを受け付けるIP アドレスを指定します。 デフォルト値は"0.0.0.0"です。このサーバが持つ全ての IP アドレスでリクエストを受け付けます。	domain.xml	■運用管理コンソールへの設定例: ＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「属性」タブの「一般」の「バインドするサーバのアドレス」に記述を行います。 「hostname」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.address=hostname ■domain.xml への設定例: http-listener 要素に設定します。 <http-service> <http-listener address="hostname"/> </http-service>
bufferSize	このコネクタが入力ストリームとして生成するバッファサイズを byte 単位で指定します。 デフォルトは 2048bytes です。	domain.xml	■運用管理コンソールへの設定例: ＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」

			－「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、、「属性」タブの「チューニング」の「リクエスト処理用入力ストリームのバッファサイズ」に記述を行います。 「4096」 ■運用管理コマンド例： otxadmin> set server.http-service.http-listener.http-listener-1.buffer-size=4096 ■domain.xml への設定例： http-listener 要素に設定します。 <http-service> <http-listener buffer-size="4096" /> </http-service>
connectionTimeout	接続を受け付けた後、リクエストの URI 行が現れるのを、このコネクタが待つミリ秒数を指定します。 デフォルト値は 60000 です。	domain.xml	■運用管理コンソールへの設定例： ＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「属性」タブの「チューニング」の「リクエスト受け付け用ソケットの接続タイムアウト値」に記述を行います。 「60000」 ■運用管理コマンド例： otxadmin> set server.http-service.http-listener.http-listener-1.connection-timeout=60000 ■domain.xml への設定例： http-listener 要素に設定します。 <http-service> <http-listener connection-timeout="60000" /> </http-service>
executor	Executor 要素の名前への参照を指定	domain.xml	■運用管理コンソールへの設定例：

	します。 もしもこの属性が有効であって、指名された executor が存在するならば、コネクタは 他のすべてのスレッド属性を無視して、この executor を使用します。 デフォルト値はありません。		＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「executor=xxx」 ■運用管理コマンド例： otxadmin> set server.http-service.http-listener.http-listener-1.property.executor=xxx ■domain.xml への設定例： http-listener 要素に property を設定します。 <http-listener <property name="executor" value="xxx"/> </http-listener>
keepAliveTimeout	このコネクタが接続を閉じる前に別の HTTP リクエストを待つミリ秒数を指定します。 デフォルト値は connectionTimeout 属性にセットされた値が使われます。	domain.xml	■運用管理コンソールへの設定例： ＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「keepalive-timeout=30」 ■運用管理コマンド例： otxadmin> set server.http-service.http-listener.http-listener-1.property.keepalive-timeout=30 ■domain.xml への設定例： http-listener 要素に property を設定します。 <http-listener <property name="keepalive-timeout"

			value="30"/> </http-listener>
maxThreads	リクエスト処理スレッドの最大数を指定します。 デフォルト値は 20 です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「属性」タブの「チューニング」の「最大プロセッサ数」に記述を行います。 「20」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.max-processors=20 ■domain.xml への設定例: http-listener 要素に設定します。 <http-service> <http-listener max-processors="20" /> </http-service>
port	このコネクタがサーバ・ソケットを作成して接続を受け付ける TCP ポート番号を指定します。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「属性」タブの「一般」の「ポート番号」に記述を行います。 「8080」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.port=8080 ■domain.xml への設定例: http-listener 要素に設定します。

			<http-service> <http-listener port="8080" /> </http-service>
tcpNoDelay	true にセットすると、TCP_NO_DELAY オプションがサーバ・ソケットにセットされます。 デフォルト値は true です。	domain.xml	■運用管理コンソールへの設定例: ＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「tcp-no-delay=false」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.tcp-no-delay=false ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener > <property name="tcp-no-delay" value="false"/> </http-listener>
accesslog-pattern-web	アクセスログの出力形式をカスタマイズすることができます。 デフォルト値は"%h %l %u %t "%r" %s %b" です。 フォーマットは以下となります。 %a - 接続元 IP アドレス %A - 自ホストの IP アドレス %b - 送信バイト数 なしの場合は"-" %B - 送信バイト数 %h - 接続元ホスト名 %H - リクエストプロトコル名 %l - 接続元論理ユーザ名 %m - リクエストメソッド	domain.xml	■運用管理コンソールへの設定例: ＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「 accesslog-pattern-web="%h %l %S %u %t ¥"%r¥" %s %b %D"」 ■運用管理コマンド例: otxadmin>set server.http-service.virtual-server.server.property.accesslog-patt

	%p – 接続ポート番号 %q – クエリ文字列 %r – リクエストの最初の一行 %s – HTTP レスポンスステータスコード %S – セッション ID %t – 日付と時間 %u – 認証済みのユーザ名 %U – リクエストされた URL %v – ローカルホスト名 %D – リクエスト処理時間(ミリ秒) %T – リクエスト処理時間(秒)		ern-web="%h %l %S %u %t ¥"%r¥" %s %b %D" ■domain.xml への設定例: http-service の virtual-server 要素に設定します。 <http-service> <virtual-server> <property name="accesslog-pattern-web" value="%h %l %S %u %t "%r" %s %b %D"/> </virtual-server> </http-service>
timeoutSeconds	アクセスのない状態のセッションのタイムアウト時間を秒単位で指定します。0 以下を指定するとこのWeb アプリケーションのセッションは期限切れになりません。 デフォルト値は 1800(秒)です。 ※web.xml と nec-web.xml への設定を同時に行った場合、web.xml への設定が優先されます。	web.xml または nec-web.xml	■web.xml への設定例: session-config 要素に設定します。 ※単位は分です。 <session-config> <session-timeout>60</session-timeout> </session-config> ■nec-web.xml への設定例: session-properties 要素に property を設定します。 <session-properties> <property name="timeoutSeconds" value="1800"/> </session-properties>
forwardRequestURL	Webアプリケーションにおいて、RequestDispatcher#forward()にて転送したServlet上でHttpServletRequest#getRquestURL()の戻り値として何が返るかを選択します。 デフォルト値は"forward"(WebOTX V8)、"original"(WebOTX V7以前)です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「Web コンテナ」 の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「forwardRequestURL=forward」 ■運用管理コマンド例: otxadmin> set server.web-container.property

	forward：forward後のURLを返却 original：forward前のURLを返却		.forwardRequestURL=forward ■domain.xml 設定例： web-container 要素に property を設定します。 <web-container> <property name=" forwardRequestURL " value="forward" /> </web-container>

コネクタ（HTTP）

項目	説明	設定箇所	設定方法
SSLEnabled	コネクタ上で SSL トラフィックを有効にする場合はこの属性を使用します。コネクタ上で SSL ハンドシェーク/暗号化/復号をオンにするには、この値を true に指定します。この値を true にする時は、同時に正しい request.getScheme()値と request.isSecure() 値をサーブレットに渡すため、scheme 属性と secure 属性を true に設定してください。 デフォルト値は false です。	domain.xml	■運用管理コンソールへの設定例： <ドメイン名> －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「SSLEnabled=true」 ■運用管理コマンド例： otxadmin> set server.http-service.http-listener.http-listener-1.property.SSLEnabled=true ■domain.xml への設定例： http-listener 要素に property を設定します。 <http-listener > <property name="SSLEnabled" value="true" /> </http-listener >
acceptCount	このサーバへの接続バックログ数（接続要求キューのサイズ）を指定します。この値を超えて要求された全てのリクエストは拒否されます。 デフォルト値は 10 です。	domain.xml	■運用管理コンソールへの設定例： <ドメイン名> －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － <リスナ名>

			の順にクリックしていき、「属性」タブの「チューニング」の「リクエスト受け付け用ソケットのバックログ」に記述を行います。 「10」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.accept-count=10 ■domain.xml への設定例: http-listener 要素に設定します。 <http-service> <http-listener accept-count="10" /> </http-service>
compressableMimeType	後述の compression で圧縮する MIME タイプをカンマ区切りで指定します。 デフォルト値は、 text/html,text/xml,text/plain です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「compressable-mime-type=text/html」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.compressable-mime-type=text/html ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener> <property name="compressable-mime-type" value="text/html"/> </http-listener>
compression	GZIP compression を使ったデータ圧縮の可否を指定します。これはサーバ帯域の節	domain.xml	■運用管理コンソールへの設定例: <ドメイン名>

	約を目的とします。パラメタとして受理可能な値は "off"（圧縮禁止)、"on"（圧縮許可，テキスト・データを圧縮)、 "force"（すべての場合で圧縮を強制)、または整数値("on" と等価ですが、出力が圧縮される前の最小のデータ量を指定します）です。 content-length が未知で compression が "on" 以上だった場合も，出力は圧縮されます。 デフォルト値は "off" です。		－「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「compression=on」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.compression=on ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener > <property name="compression" value="on"/> </http-listener>
connectionLinger	このコネクタが使うソケットを閉じるとき、ソケットが滞留するミリ秒数を指定します。 デフォルト値は -1 (ソケット滞留は禁止)です。	domain.xml	■運用管理コンソールへの設定例: ＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「connectionLinger=15」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.connection-linge=15 ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener > <property name="connectionLinger" value="15"/>

			</http-listener>
disableUploadTimeout	サーブレットが実行されている間、設定されているものより長い接続タイムアウトをサーブレット・コンテナに許すかどうかを指定します。 デフォルト値は true です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「disableUploadTimeout=true」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.disableUploadTimeout=true ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener > <property name="disableUploadTimeout" value="true"/> </http-listener >
maxHttpHeaderSize	リクエストとレスポンスの HTTP ヘッダの最大サイズ（バイト単位)を指定します。 デフォルト値は 4096（4KB）です。 ※AJP リスナ利用時の HTTP ヘッダ最大サイズは AJP プロトコルの制限により、8KB 固定です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「maxHttpHeaderSize=8000」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.maxHttpHeaderSize=8000 ■domain.xml への設定例:

			http-listener 要素に property を設定します。 <http-listener > <property name="maxHttpHeaderSize" value="8000" /> </http-listener>
maxKeepAliveRequests	サーバが接続を閉じるまでにパイプラインされる HTTP リクエストの最大個数を指定します。この属性を 1 にセットすると HTTP/1.1 の keep-alive とパイプラインが禁止されます。これを -1 にセットすると，無制限の個数のパイプライン化または keep-alive の HTTP リクエストが許されます。 デフォルト値は 100 です	domain.xml	■運用管理コンソールへの設定例： <ドメイン名> －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「maxKeepAliveRequests=100」 ■運用管理コマンド例： otxadmin> set server.http-service.http-listener.http-listener-1.property.maxKeepAliveRequests=100 ■domain.xml への設定例： http-listener 要素に property を設定します。 <http-listener > <property name="maxKeepAliveRequests" value="100" /> </http-listener >
noCompressionUserAgents	compression の機能を用いない HTTP クライアントの user-agent をマッチする正規表現のカンマ区切りの並びで指定します。 デフォルト値は空文字列（正規表現のマッチングをしません）です。	domain.xml	■運用管理コンソールへの設定例： <ドメイン名> －「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「noCompressioUserAgents=opera」

			■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property. noCompressioUserAgents=opera ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener > <property name="noCompressionUserAgents" value="opera" /> </http-listener >
restrictedUserAgents	keep alive の機能を用いない HTTP クライアントの user-agent をマッチする正規表現のカンマ区切りの並びで指定します。 デフォルト値は空文字列（正規表現のマッチングをしません）です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「restrictedUserAgents~opera」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property. restrictedUserAgents~opera ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener > <property name="restrictedUserAgents" value="opera" /> </http-listener >
server	HTTP レスポンスの「Server」ヘッダに返す値を指定します。 デフォルト値は”WebOTX_Web_Container/<バージョン番号>”です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」

			- ＜リスナ名＞ の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「server= WebOTX_Web_Container/8.1」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.server=WebOTX_Web_Container/8.1 ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener > <property name="server" value="WebOTX_Web_Container/8.1" /> </http-listener >
socketBuffer	ソケットの出力バッファサイズ(byte 単位)を指定します。-1 にするとバッファを使用しません。 デフォルト値は 9000 です。	domain.xml	■運用管理コンソールへの設定例: ＜ドメイン名＞ -「アプリケーションサーバ」 -「HTTP サービス」 -「HTTP リスナ」 - ＜リスナ名＞ の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「socket-buffer=48」 ■運用管理コマンド例: otxadmin> set server.http-service.http-listener.http-listener-1.property.socket-buffer=48 ■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener > <property name="socket-buffer" value="48"/> </http-listener >
threadPriority	リクエスト処理を行うプロセスのスレッド優先順位を指定します。	domain.xml	■運用管理コンソールへの設定例: ＜ドメイン名＞

	デフォルト値は 5 です。		－「アプリケーションサーバ」 －「HTTP サービス」 －「HTTP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「thread-priority=10」 ■運用管理コマンド例： otxadmin> set server.http-service.http-listener.http-listener-1.property.thread-priority=10 ■domain.xml への設定例： http-listener 要素に property を設定します。 <http-listener > <property name="thread-priority" value="10"/> </http-listener >

コネクタ（AJP）

項目	説明	設定箇所	設定方法
backlog	このサーバへの接続バックログ数（接続要求キューのサイズ）を指定します。この値を超えて要求された全てのリクエストは拒否されます。 デフォルト値は 10 です。	domain.xml	■運用管理コンソールへの設定例： ＜ドメイン名＞ －「アプリケーションサーバ」 －「HTTP サービス」 －「AJP リスナ」 － ＜リスナ名＞ の順にクリックしていき、「属性」タブの「チューニング」の「リクエスト受け付け用ソケットのバックログ」に記述を行います。 「10」 ■運用管理コマンド例： otxadmin> set server.http-service.http-listener.ajp-listener-1.acceptcount=10 ■domain.xml への設定例：

			http-listener 要素に設定します。 <http-service> <http-listener accept-count="10"/> </http-service
packetSize	AJP パケットの最大値（Byte）を指定します。 デフォルト値は 8192 (8KByte)で、指定可能な最大値は 65535 です。	domain.xml	■運用管理コンソールへの設定例： <ドメイン名> －「アプリケーションサーバ」 －「HTTP サービス」 －「AJP リスナ」 － <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「packetSize=10000」 ■運用管理コマンド例： otxadmin> set server.http-service.http-listener.ajp-listener-1.property.packetSize=10000 ■domain.xml への設定例： http-listener 要素に property を設定します。 <http-listener > <property name="packetSize" value="10000"/> </http-listener >
tomcatAuthentication	認証処理を WebOTX 側で行うかどうかを指定します。 デフォルト値は true です。	domain.xml	■運用管理コンソールへの設定例： <ドメイン名> －「アプリケーションサーバ」 －「HTTP サービス」 －「AJP リスナ」 － <リスナ名> の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「otxAuthentication=false」 ■運用管理コマンド例： otxadmin> set server.http-service.http-listener.ajp-listener-1.property. otxAuthentication=false

			■domain.xml への設定例: http-listener 要素に property を設定します。 <http-listener > <property name="otxAuthentication" value="false" /> </http-listener >

コンテキスト(共通)

項目	説明	設定箇所	設定方法
backgroundProcessorDelay	このコンテキストとその子コンテナでの backgroundProcess メソッドの呼出しと呼出しの間の遅延時間(秒) を指定します。 子コンテナの遅延値が正ならば子コンテナが自分自身の処理スレッドを使うことを意味するため、コンテキストの処理スレッドがわざわざ子コンテナのメソッドを呼び出すことはしません。この値を正にすると、そのコンテキスト用の処理スレッドが1本生成され、指定された秒数だけ待った後、このスレッドはこのコンテキストおよびその子コンテナすべての backgroundProcess メソッドをどれか1個呼び出します。コンテキストは background 処理を使ってセッションの破棄と再ロードのためのクラス監視を行います。 デフォルト値は -1 であり、コンテキストが自分の親であるホストの background 処理スレッドに依存します。	nec-web.xml	■nec-web.xml への設定例: manager-properties 要素に property を設定します。 <nec-web-app> <session-config> <session-manager > <manager-properties> <property name="reapIntervalSeconds" value="-1" />
cookies	クライアントがクッキーをサポートしているとき、セッション管理にクッキーを使いたい場合は、true にセットしてください。 セッション管理にクッキーを使うことを禁止し、アプリケーションによる URL Rewriting だけに頼りたい場合はこの値を false にしてください。 デフォルト値は true です。	nec-web.xml または domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーション」 -「Web モジュール」 - <Web モジュール名> の順にクリックしていき、「属性」タブの「セッション ID の管理」にチェックを行います。 ■nec-web.xml への設定例: session-properties 要素にpropertyを設定しま

			す。 <nec-web-app> <session-config> <session-properties> <property name="enableCookies" value="false" /> ■domain.xml への設定例: web-module 要素に設定します。 <applications> <web-module cookies="true"/> </applications>
crossContext	このアプリケーション内での ServletContext.getContext() の呼出しが、この仮想ホストで走っている他の web アプリケーションへのリクエストディスパッチャの結果を返却するようにしたい場合は、これを true にしてください。セキュリティを意識した環境で、getContext() が常に null を返すようにしたい場合は、これを false にしてください。 デフォルト値は false です。	nec-web.xml または domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーション」 -「Web モジュール」 - <Web モジュール名> の順にクリックしていき、「属性」タブの「他モジュールへのアクセス」にチェックを行います。 ■運用管理コマンド例: otxadmin> set server.applications.web-module.<Web アプリケーション名>.cross-context=true ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。 <nec-web-app> <property name="crossContextAllowed" value="true" /> </nec-web-app> ■domain.xml への設定例: web-module 要素に設定します。 <applications> <web-module cross-context="true"/> </applications>
docBase	この Web アプリケーションの Document	domain.xml	■運用管理コンソールへの設定例:

	Base（別名 Context Root ）ディレクトリ、または WAR ファイルへのパスです。WebOTX では、Web アプリケーションを配備した時にこの値が設定されます。 デフォルト値は ${com.nec.webotx.instanceRoot}/applications/j2ee-modules/<Web アプリケーション名> です。		<ドメイン名> －「アプリケーション」 －「Web モジュール」 － <Web モジュール名> の順にクリックしていき、「属性」タブの「パス」で確認ができます。設定変更はできません。 ■運用管理コマンド例： otxadmin> set server.applications.web-module.<Web アプリケーション名>.location="${com.nec.webotx.instanceRoot}/applications/j2ee-modules/<Web アプリケーション名>" ■domain.xml への設定例： web-module 要素に設定します。 <applications> <web-module location="${com.nec.webotx.instanceRoot}/applications/j2ee-modules/<Web アプリケーション名>" </applications>
override	この値を true に指定すると、自分のホストの DefaultContext 要素での対応する設定を上書きします。 デフォルト値は false であり、DefaultContext 要素での設定が使用されます。	domain.xml	■運用管理コンソールへの設定例： <ドメイン名> －「アプリケーション」 －「Web モジュール」 － <Web モジュール名> の順にクリックしていき、「属性」タブの「他モジュールへのアクセス」にチェックを行います。 ■運用管理コマンド例： otxadmin> set server.applications.web-module.<Web アプリケーション名>.override=true ■domain.xml への設定例： web-module 要素に設定します。 <applications> <web-module override="false"/>

			</applications>
privileged	このコンテキストが manager サーブレットのようなコンテナサーブレットを使えるようにしたい場合は、これを true にしてください。 デフォルト値は false です。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> －「アプリケーション」 －「Web モジュール」 － <Web モジュール名> の順にクリックしていき、「属性」タブの「内部クラスへのアクセス」にチェックを行います。 nec-web.xml へ設定する場合は記述方法を記載してください。 ■運用管理コマンド例: otxadmin> set server.applications.web-module.<Web アプリケーション名>..privileged=false ■domain.xml への設定例: web-module 要素に設定します。 <applications> <web-module privileged="false"/> </applications>
path	この Web アプリケーションのコンテキストパスを指定します。WebOTX では、Web アプリケーションを配備するときに"コンテキストルート"として指定します。	domain.xml	■運用管理コンソールへの設定例: <ドメイン名> －「アプリケーション」 －「Web モジュール」 － <Web モジュール名> の順にクリックしていき、「属性」タブの「コンテキストルートで確認ができます。設定変更はできません。 ■運用管理コマンド例: otxadmin> set server.applications.web-module.<Web アプリケーション名>.context-root=xxxx ■domain.xml への設定例: web-module 要素に設定します。 <applications>

			<web-module context-root="xxxx" /> </applications>
reloadable	/WEB-INF/classes/と/WEB-INF/lib にあるクラスが変更されていないかどうかを監視します。変更の検出時に自動的にその web アプリケーションを再ロードさせたい場合はこの値を true にしてください。 この機能はアプリケーション開発時には非常に有用ですが、顕著な実行時オーバヘッドを要するため、製品アプリケーションを配備して使用するときには勧められません。 デフォルト値は false です。	nec-web.xml または domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーション」 -「Web モジュール」 - <Web モジュール名> の順にクリックしていき、「属性」タブの「自動リロード」にチェックを行います。 ■運用管理コマンド例: otxadmin> set server.applications.web-module.<Web アプリケーション名>.reloadable=true ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。 <nec-web-app> <property name="relodable" value="true" /> </nec-web-app> ■domain.xml への設定例: web-module 要素に設定します。 <applications> <web-module relodable="true" /> </applications>

コンテキスト(Standard)

項目	説明	設定箇所	設定方法
allowLinking	web アプリケーション内でシンボリックリンク(web アプリケーションの base path の外側のリソースを参照できる)を有効にするかを指定します。 デフォルト値は false です。 注意: Windows プラットフォームまたは他の英字の大小を区別するファイルシステムを	nec-web.xml または domain.xml	※運用管理コンソールへの設定方法はありません。(V8.1 未サポート) Web アプリケーション配備解除や再配備すると設定が消えます。nec-web.xml に記述するようにしてください。 ■運用管理コマンド例: otxadmin> set server.applications.web-m

	持たない OS において、この値を true にすべきではありません。case sensitivity check を無効にして JSP ソースコードの暴露を許すことになるセキュリティ問題があります。		odule.<Web アプリケーション名>.property.allowLinking=true ■nec-web.xml への設定例： nec-web-app 要素に property を設定します。 <nec-web-app> <property name="allowLinking" value="true" /> </nec-web-app> ■domain.xml への設定例： web-module 要素に設定します。 web-module 要素の property に設定します。 <applications> <web-module> <property name="allowLinking" value="true" /> </web-module> </applications>
antiJARLocking	この値を true にすると、クラスローダは，、リソースを JAR 内部から URL 経由でアクセスするとき、JAR ファイルロッキングを回避するように特別な手段をとります。これはアプリケーションの起動時間に多少の遅延が生じますが、ファイルロッキングが起こり得るプラットフォームや設定状況によっては有用です。 デフォルト値は false です。	nec-web.xml または domain.xml	※運用管理コンソールへの設定方法はありません。（V8.1 未サポート） Web アプリケーションを配備解除や再配備すると設定が消えます。nec-web.xml に記述するようにしてください。 ■運用管理コマンド例： otxadmin> set server.applications.web-module.<Web アプリケーション名>.property.antiJARLocking=true ■nec-web.xml への設定例： nec-web-app 要素に property を設定します。 <nec-web-app> <property name="antiJARLocking" value="true" /> </nec-web-app> ■domain.xml への設定例：

			web-module 要素の property に設定します。 <web-module> <property name="antiJARLocking" value="true" /> </web-module>
antiResourceLocking	この値を true にすると、すべての場合のファイルロッキングを防ぎます。これはアプリケーションの起動時間に大幅な遅延が生じますが、ファイル ロッキングが起こり得るプラットフォームや設定状況での web アプリの完全なホットデプロイやデプロイ解除を可能にします。 デフォルト値は true です。	nec-web.xml または domain.xml	※運用管理コンソールへの設定方法はありません。(V8.1 未サポート) Web アプリケーションを配備解除や再配備すると設定が消えます。nec-web.xmlに記述するようにしてください。 ■運用管理コマンド例: otxadmin> set server.applications.web-module.<Web アプリケーション名>.property.antiResourceLocking=true ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。 <nec-web-app> <property name="antiResourceLocking" value="true" /> </nec-web-app> ■domain.xml への設定例: web-module 要素の property に設定します。 <web-module> <property name="antiResourceLocking" value="true" /> </web-module>
cacheMaxSize	キロバイト単位での静的リソースキャッシュの最大サイズを指定します。 デフォルト値は 10240 (10MB)です。	nec-web.xml または domain.xm	※運用管理コンソールへの設定方法はありません。(V8.1 未サポート) Web アプリケーションを配備解除や再配備すると設定が消えます。nec-web.xmlに記述するようにしてください。 ■運用管理コマンド例: otxadmin> set server.applications.web-module.<Web アプリケーション名>.property.c

			acheMaxSize=10240 ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。 <nec-web-app> <property name="cacheMaxSize" value="10240" /> </nec-web-app> ■domain.xml への設定例: web-module 要素の property に設定します。 <web-module> <property name="cacheMaxSize" value="10240" > </web-module>
cacheTTL	キャッシュ エントリの再バリデーション (revalidation) 間隔をミリ秒単位で指定します。 デフォルト値は 5000 (5 秒)です。	nec-web.xml または domain.xm	※運用管理コンソールへの設定方法はありません。(V8.1 未サポート) Web アプリケーションを配備解除や再配備すると設定が消えます。nec-web.xml に記述するようにしてください。 ■運用管理コマンド例: otxadmin> set server.applications.web-m odule.<Web アプリケーション名>.property.c acheTTL=5000 ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。 <nec-web-app> <property name="cacheTTL" value="5000" /> </nec-web-app> ■domain.xml への設定例: web-module 要素の property に設定します。 <web-module> <property name="cacheTTL" value="5000" />

			</web-module>
cachingAllowed	静的リソースに対してキャッシュを行うかどうかを指定します。 デフォルト値は true です。	nec-web.xml または domain.xm	※運用管理コンソールへの設定方法はありません。(V8.1 未サポート) Web アプリケーションを配備解除や再配備すると設定が消えます。nec-web.xmlに記述するようにしてください。 ■運用管理コマンド例: otxadmin> set server.applications.web-module.<Web アプリケーション名>.property.cachingAllowed=false ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。 <nec-web-app> <property name="cachingAllowed" value="false" /> </nec-web-app> ■domain.xml への設定例: web-module 要素の property に設定します。 <web-module> <property name="cachingAllowed" value="false"/ > </web-module>
caseSensitive	この値を true に指定すると、すべての case sensitivity check が無効にされます。 デフォルト値は true です。 注意: Windows プラットフォームまたは他の英字の大小を区別するファイルシステムを持たない OS において、この値を true にすべきではありません。case sensitivity check を無効にして JSP ソースコードの暴露を許すことになるセキュリティ問題があります。	nec-web.xml または domain.xm	※運用管理コンソールへの設定方法はありません。(V8.1 未サポート) Web アプリケーションを配備解除や再配備すると設定が消えます。nec-web.xmlに記述するようにしてください。 ■運用管理コマンド例: otxadmin> set server.applications.web-module.<Web アプリケーション名>.property.caseSensitive=false ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。

			<nec-web-app> <property name="caseSensitive" value="false" /> </nec-web-app> ■domain.xml への設定例: web-module 要素の property に設定します。 <web-module> <property name= "caseSensitive" value="false"/ > </web-module>
processTlds	コンテキストが起動時に TLD [Tag Library Descriptor] を処理すべきかどうかを指定します。 デフォルト値は true です。	domain.xml	※運用管理コンソールへの設定方法はありません。(V8.1 未サポート) Web アプリケーションを配備解除や再配備すると設定が消えます。nec-web.xmlに記述するようにしてください。 ■運用管理コマンド例: otxadmin> set server.applications.web-module.<Web アプリケーション名>.property.processTlds=false ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。 <nec-web-app> <property name="processTlds" value="false" /> </nec-web-app> ■domain.xml への設定例: web-module 要素の property に設定します。 <web-module> <property name= "processTlds" value="false"/ > </web-module>
swallowOutput	この値を true に指定すると、web アプリケーションによる System.out と System.err へのバイト出力が web アプリケーションの logger	nec-web.xml または domain.xm	※運用管理コンソールへの設定方法はありません。(V8.1 未サポート) Web アプリケーションを配備解除や再配備す

	へリダイレクトされます。 デフォルト値は false です。		ると設定が消えます。nec-web.xmlに記述するようにしてください。 ■運用管理コマンド例: otxadmin> set server.applications.web-module.<Web アプリケーション名>.property.swallow-output=true ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。 <nec-web-app> <property name="swallowOutput" value="true" /> </nec-web-app> ■domain.xml への設定例: web-module 要素の property に設定します。 <web-module> <property name="swallow-output"="true" /> </web-module>
tldNamespaceAware	この値をtrueに指定すると、TLDファイルのXML バリデーションが名前空間対応(namespace-aware) になります。 デフォルト値は false です。	nec-web.xml または domain.xm	※運用管理コンソールへの設定方法はありません。(V8.1 未サポート) Web アプリケーションを配備解除や再配備すると設定が消えます。nec-web.xmlに記述するようにしてください。 ■運用管理コマンド例: otxadmin> set server.applications.web-module.<Web アプリケーション名>.property.cacheMaxSize=10240 ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。 <nec-web-app> <property name="cacheMaxSize" value="10240" /> </nec-web-app>

			■domain.xml への設定例： web-module 要素の property に設定します。 <web-module> <property name="cacheMaxSize" value="10240" > </web-module>
tldValidation	この値を true に指定すると、TLD ファイルが起動時に XML バリデーションを受けとります。 デフォルト値は false です。	nec-web.xml または domain.xm	※運用管理コンソールへの設定方法はありません。（V8.1 未サポート） Web アプリケーションを配備解除や再配備すると設定が消えます。nec-web.xml に記述するようにしてください。 ■運用管理コマンド例： otxadmin> set server.applications.web-module.<Web アプリケーション名>.property.tldValidation=true ■nec-web.xml への設定例： nec-web-app 要素に property を設定します。 <nec-web-app> <property name="tldValidation" value="true" /> </nec-web-app> ■domain.xml への設定例： web-module 要素の property に設定します。 <web-module> <property name="tldValidation" value="true"/ > </web-module>
unloadDelay	サーブレットがアンロードされるまでコンテナが待つ時間（ミリ秒単位)を指定します。 デフォルト値は 2000（ミリ秒）です。	nec-web.xml または domain.xm	※運用管理コンソールへの設定方法はありません。（V8.1 未サポート） Web アプリケーションを配備解除や再配備すると設定が消えます。nec-web.xml に記述するようにしてください。 ■運用管理コマンド例：

			otxadmin> set server.applications.web-m odule.<Web アプリケーション名>.property.u nloadDelay=5000 ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。 <nec-web-app> <property name="unloadDelay" value="5000" /> </nec-web-app> ■domain.xml への設定例: web-module 要素の property に設定します。 <web-module> <property name="unloadDelay" value="5000"/ > </web-module>
unpackWAR	この値を true に指定すると、圧縮された web アプリケーションを走らせる前にそれをすべて展開します。 デフォルト値は true です。	nec-web.xml または domain.xm	※運用管理コンソールへの設定方法はありません。(V8.1 未サポート) Web アプリケーションを配備解除や再配備すると設定が消えます。nec-web.xmlに記述するようにしてください。 ■運用管理コマンド例: otxadmin> set server.applications.web-m odule.<Web アプリケーション名>.property.u npackWAR=false ■nec-web.xml への設定例: nec-web-app 要素に property を設定します。 <nec-web-app> <property name="unpackWAR" value="false" /> </nec-web-app> ■domain.xml への設定例: web-module 要素の property に設定します。 <web-module>

			<property name="unpackWAR" value="false"/> </web-module>
workDir	関連する web アプリケーション内のサーブレットが一時的な読み書きに使うために、コンテキストが用意するスクラッチディレクトリ（scratch directory)への絶対パス名を指定します。このディレクトリは，Servlet Specification に記述されるように javax.servlet.context.tempdir という名前の(java.io.File 型の）サーブレット context 属性によって web アプリケーション内のサーブレットから可視になります。 デフォルト値は <domain_dir>/generated/jsp /j2ee-modules/<appname> です、	nec-web.xml	■nec-web.xml への設定例： nec-web-app 要素に property を設定します。 <nec-web-app> <property name="tempdir" value="true" /> </nec-web-app>

Manager(共通)

項目	説明	設定箇所	設定方法
distributable	このアプリケーションでセッションレプリケーションを行いたい場合に、distributable 属性を記述します。デフォルトでは記述されていません。 distributable 属性が指定されているとき、直列化不可能(Serializable)なオブジェクトをセッションに格納しようとすると IllegalArgumentException が発生します。 ※セッションレプリケーション（JNDI サーバへのセッション情報の登録）は次のメソッド処理の延長で実行されます。 HttpServletRequest#getSession() HttpSession#getAttribute() HttpSession#getAttributeNames() HttpSession#removeAttribute() HttpSession#setAttribute() HttpSession#invalidate() HttpSession#getMaxInactiveInterval()	web.xml	■web.xml への設定例： web-app 要素に記述します。 <web-app> <distributable /> </web-app>

	HttpSession#setMaxInactiveInterval()		

Manager (Standard)

項目	説明	設定箇所	設定方法
maxActiveSessions	マネージャによって生成されるアクティブ・セッションの最大数を制限します。 -1（デフォルト値）にすると、セッションは無制限に生成されます。	nec-web.xml	■nec-web.xml への設定例： manager-properties 要素の property に設定します。 <nec-web-app> <session-config> <session-manager > <manager-properties> <property name="maxSessions" value="3000" />
maxInactiveInterval	セッションタイムアウトの監視を行う間隔を指定します。 デフォルト値は 60（秒）です。	nec-web.xml	■nec-web.xml への設定例： manager-properties 要素の property に設定します。 <nec-web-app> <session-config> <session-manager > <manager-properties> <property name="reapIntervalSeconds" value="60" />
pathname	アプリケーション再起動時に保存されるセッション状態を記述したファイルの格納場所を指定します。 デフォルト値は workDir/SESSIONS.ser です。	nec-web.xml	■nec-web.xml への設定例： manager-properties 要素の property に設定します。 <nec-web-app> <session-config> <session-manager > <manager-properties> <property name="sessionFilename" value="<ファイルパス>" />

Realm

JDBC レルムについては WebOTX マニュアルの以下を参照してください。

運用編⇒ユーザ管理⇒4.3JDBC レルムを使用する場合

項目	説明	設定箇所	設定方法

classname	JDBC レルムに使用するクラス名を記述します	Domain.xml	JDBC レルムの設定は下記のプロパティをまとめて設定します。 ■運用管理コンソールへの設定例 〈ドメイン名〉 –「アプリケーションサーバ」 –「セキュリティサービス」 の順にクリックしていき、「操作」タブの「認証レルムの作成」にて設定します。 ■運用管理コマンドでの設定例 otxadmin> create-auth-realm --classname com.nec.webotx.enterprise.security.auth.realm.jdbc.JDBCRealm --property ″driverName=oracle.jdbc.driver.OracleDriver .jaas-context=JDBCRealm:connectionURL=jdbc¥:oracle¥:thin¥:@OraclServer¥:1521¥:OracleDB:connectionName=scott:connectionPassword=tiger:userTable=jdbc_user:userNameCol=userid:userCredCol=passwd:userRoleTable=jdbc_role:roleNameCol=role:digest=MD5:digestEncoding=Shift_JIS″ JDBCRealm1
connectionName	データベースにJDBC接続するためのユーザ名を記述します。		
connectionPassword	データベースにJDBC接続するためのパスワードを記述します。		

digest	データベースに格納されているパスワードの暗号方式を指定します。 デフォルトではクリアテキストで保存されます。	domain.xml	■domein.xml への設定例: 〈security-service〉 〈auth-realm

digestEncoding	“digest”によって指定された暗号方式によって暗号化を行う時のエンコード形式を指定します。 デフォルト値は OS の設定に依存します。		name="JDBCrealm"/> <property name="connectionName" value="scott"/> <property name="connectionPassword" value="tiger"/> <property name="connectionURL" value="jdbc:oracle:thin:@OraclServer:1521:OracleDB"/> <property name="digest" value="MD5"/> <property name="digestEncoding" value="Shift_JIS"/> <property name="driverName" value="oracle.jdbc.driver.OracleDriver"/> <property name="roleNameCol" value="role"/> <property name="userCredCol" value="passwd"/> <property name="userNameCol" value="userid"/> <property name="userRoleTable" value="jdbc_role"/> <property name="userTable" value="jdbc_user"/> </auth-realm> </security-service>
driverName	使用する JDBC ドライバのクラス名を記述します。		
roleNameCol	対応するユーザを含む "user roles"テーブルの列名を記述します。		
userCredCol	対応するパスワードを含む"users" テーブルの列名を記述します。		
userNameCol	ユーザ名を含む、"users"と"user roles"テーブルの列名を記述します。		
userRoleTable	記述した userNameCol と roleNameCol を含む "user roles"テーブルの名前を記述します。		
userTable	記述した userNameCol と userCredCol を含む "users"テーブルの名前を記述します		
connectionURL	データベースに JDBC 接続するための URL を記述します。		
Pathname	アプリケーション再起動時に保存されるセッション状態を記述したファイルの格納場所を指定します。	nec-web.xml	■nec-web.xml への設定例: manager-properties 要素の property に設定します。 <nec-web-app> <session-config> <session-manager > <manager-properties> <property name="sessionFilename" value="<ファイルパス>" />

Loader

項目	説明	設定箇所	設定方法
delegate	Web アプリケーションを読み込む際にクラスをロードするかどうかを指定します。 デフォルト値は true です。	nec-web.xml	■nec-web.xml への設定例: class-loader 要素に設定します。 <nec-web-app> <class-loader delegate="true"/> </nec-web-app>
loaderClass	クラスローダに独自のクラスを指定する場合に指定します。 デフォルト値はありません。 loaderName には Tomcat の "org.apache.catalina.loader.WebappLoader" クラスを継承した独自のクラス名を指定してください。 classloaderName には Tomcat の "org.apache.catalina.loader.WebappClassLoader"クラスを継承した独自のクラス名を指定してください。	nec-web.xml または domain.xml	■運用管理コンソールへの設定例: <ドメイン名> -「アプリケーションサーバ」 -「Web コンテナ」 の順にクリックしていき、「操作」タブの「プロパティの設定」に記述を行います。 「loaderClass-/servlets-examples=jp.co.nec.util.V8WebappLoader,jp.co.nec.util.V8WebappClassLoader」 ■運用管理コマンド例: otxadmin> set server.web-container.property.loaderClass-/AP 名=WebappLoader 名,WebappClassLoader 名 ■nec-web.xml への設定例: class-loader 要素の property に overrideablejavaxpackages を設定します。 <nec-web-app> <class-loader> <property name="loaderName" value="<クラス名>,…." /> <property name="classloaderName" value="<クラス名>,…." /> </class-loader> </nec-web-app> ■domain.xml への設定例: web-container 要素に設定します。 <web-container> <propertiy name="loderClass-<AP 名>" value="WebappLoader 名,WebappClassLoader

			名″/> </web-container>

Single Sign On

項目	説明	設定箇所	設定方法
cookieDomain	シングルサインオンのためのクッキーを使用する場合に指定します。 デフォルト値はありません。	nec-web.xml	■nec-web.xml への設定例: cookie-properties 要素に property を設定します。 <nec-web-app> <session-config> <session-manager > <cookie-properties> <property name=″cookieDomain″ value=″<クッキードメイン名>″ />

4.2. Java VM オプションのサポート

WebOTX は、Servlet/JSP コンテナにさまざまな機能があり、それらの機能の制御を行える Java VM オプションを説明します。

下記にプロセスモード別の設定方法を記載します。

■シングルプロセスモードの場合

・domain.xml 直接編集での設定方法

ドメインを停止し、下記の例のように対象ファイルの<java-config>要素に<system-jvm-options>タグを記述した後”-D”の後に続いて設定を記述してください。その後、ドメインを起動することで設定が反映されます。

対象ファイル:(WebOTX インストールディレクトリ)/domains/domain1/config/domain.xml

設定例:

```
<java-config>
  <system-jvm-options>-Dcom.nec.webotx.enterprise.web.session.jndidelete=480</system-jvm-options>
</java-config>
```

・運用管理コンソールでの設定方法

運用管理コンソールにログインし、下記の箇所に設定を記述した後、ドメインを再起動してください。

<ドメイン名>

- 「アプリケーションサーバ」
  - 「JVM 構成」

の順にクリックしていき、「属性」タブの「JVM オプション」の「システム JVM オプション」の欄に”-D”の後に続いて設定を記述してく

ださい。その後、ドメインを再起動することで反映されます。

設定例:

-Dcom.nec.webotx.enterprise.web.session.jndidelete=480

■マルチプロセスモードの場合

・運用管理コンソールでの設定方法

運用管理コンソールにログインし、下記の箇所に設定を記述した後、TP システムを再起動してください。

＜ドメイン名＞

- 「TP システム」
  - 「アプリケーショングループ」
    - ＜アプリケーショングループ名＞
      - 「プロセスグループ」
        - ＜プロセスグループ名＞

の順にクリックしていき、「属性」タブの「Java システムプロパティの欄に設定を行ってください。

「名前」に項目名を記述し、「値」に指定する値を記述してください。

設定例:

名前	値
com.nec.webotx.enterprise.web.session.jndidelete	480

設定項目一覧

項目	説明
com.nec.webotx.enterprise.web.session.jndidelete	JNDI に格納したセッション情報をタイムアウト後、何秒で削除するかを指定します。 デフォルト値は 480 秒です。
com.nec.webotx.enterprise.web.session.jndidelete	JNDI に格納したセッション情報をタイムアウト後、何秒で削除するかを指定します。 デフォルト値は 480 秒です。
com.nec.webotx.enterprise.web.serverconfig.Append	JK プラグインが利用する連携のための設定ファイルを出力するか否かを指定します。 デフォルト値は true です。
com.nec.webotx.enterprise.web.serverconfig.ForwardAll	JK プラグインに対するコンテキストのリクエストを全て転送するか否かを指定します。 デフォルト値は true です。
com.nec.webotx.enterprise.web.serverconfig.ForwardKeyInfo	Web サーバの持つ認証情報を Webコンテナに転送するか否かを指定します。 デフォルト値は true です。
com.nec.webotx.enterprise.web.serverconfig.NoRoot	JKプラグインに対するリクエストのうち、ルートコンテキスト(/)を転送するか否かを指定します。 デフォルト値は true です。
com.nec.webotx.enterprise.web.serverconfig.OMlistener.ConfAuto	IIOP プラグインが利用する連携のための設定ファイル

	が既に存在した場合に出力しないかどうかを指定します。 デフォルト値は true です。
com.nec.webotx.enterprise.web.serverconfig.OMlistener.Context	IIOP プラグインが利用する連携のための設定ファイルに、コンテキストの情報を出力するか否かを指定します。 デフォルト値は false です。
com.nec.webotx.enterprise.web.connector.checkLastProcessTimeInterval	IIOP プラグインと Web コンテナとの通信状況のチェック間隔を指定します。 デフォルト値は 1800 です。
com.nec.webotx.enterprise.web.connector.fileSwitchSize	IIOP プラグインが扱うメッセージのサイズが大きい場合にテンポラリファイルを利用しますが、そのしきい値を指定します。 デフォルト値は 128(Kbyte)です。
com.nec.webotx.enterprise.web.connector.iiopDriveMode	fileSwitchSize で指定したサイズを超えた場合に利用する一時媒体にメモリを利用するかファイルを利用するかどうかを指定します。 デフォルト値はファイルです。
com.nec.webotx.enterprise.web.connector.iiopSendMaxSize	IIOP プラグインと Web コンテナ間でやりとりするメッセージを分割送信する場合のサイズを指定します。 デフォルト値は 128(Kbyte)です。
com.nec.webotx.enterprise.web.connector.poolBufferSize	IIOP プラグインからのリクエストを処理する時に使用するバッファ領域の初期個数を指定します。 デフォルト値は 5 です。
com.nec.webotx.enterprise.web.connector.securityEnabled	マルチプロセスモード時、Cookie に登録したセッション情報にセキュアオプションを付加するか否かを指定します。 デフォルト値は false です。
com.nec.webotx.enterprise.web.connector.useBodyEncodingForURI	ContentType 又は Request.setCharacterEncoding メソッドで指定されたエンコーディングを URL のデコードに使用するか否かを指定します。 デフォルト値は false です。
com.nec.webotx.enterprise.web.connector.URIEncoding	「http://contents?a=%12%34」形式で記述された URL をデコードする際の文字エンコーディングを指定します。 デフォルト値は「ISO-8859-1」です。
com.nec.webotx.webcontainer.performance	Web Edition 利用時に、不要な処理をスキップして性能改善を行うか否かを指定します。

	デフォルト値は true です。
com.nec.webotx.enterprise.overrideablejavaxpackages	Web アプリケーション内に配置したクラスのうちパッケージ名が javax、sun、org 等で始まるクラスは参照することができません。参照を許可するパッケージをここで指定します。複数設定する場合はカンマで区切って指定します。("javax.sub,org.sub"等) デフォルト値はありません。
com.nec.webotx.enterprise.web.InvalidOverridableJavaxPackages	Webapp クラスローダが自身でロードするクラスから除外するパッケージを指定します。 デフォルト値はありません。
com.nec.webotx.enterprise.registerRequest	リクエスト毎に発生する MO 登録処理を行うかどうかを指定します。 デフォルト値は false です。
com.nec.webotx.enterprise.web.connector.useCoyoteConnector	Coyote コネクタを使用するか否かを指定します。false を指定すると Grizzly コネクタを使用します。 デフォルト値は true です。
jvmRoute	ロードバランサを使用して、サーバ固定で振り分けたい場合に jsessionid に付加させる任意の文字列を指定します。プラグインによる Sticky セッションの負荷分散を使用する場合には値にワーカ名(ajp13 や otxiiop1 など)を指定してください。 デフォルト値はありません。

5. Q&A

Tomcat からの移行時によく起きる問題や Tomcat からの移行以外にもよくある質問に対する Q&A をまとめました。

5.1. Web アプリケーションの実行エラーに関する Q&A

Q1 Web アプリケーションを実行すると ClassCastException が発生するのですが、どのような原因が考えられますか？

A1 対象となるクラスのロードが正しく行われていない可能性があります。nec-web.xml の設定を変更して、クラスのロード処理を変えて、アプリケーションの動作を確認してください。(以下の例を参考に"delegate=false"を指定した nec-web.xml を追加してください)

```
<?xml version="1.0" encoding="UTF-8"?>
<nec-web-app xmlns="http://java.sun.com/xml/ns/j2ee">
  <context-root>context_name</context-root>
  <class-loader delegate="false"/>
</nec-web-app>
```

nec-web.xml の詳細は 2.6.2. クラスロード優先順位の設定を参照してください。

Q2 Web アプリケーションを実行するとセキュリティ例外が発生するのですが、どのような原因が考えられますか？

A2 セキュリティポリシーの追加が必要です。本資料の「2.7 セキュリティポリシーの設定」をご覧ください。

Q3 Web アプリケーションからのログが出力されないのですが、どのような原因が考えられますか？

A3 log4jの設定を変更してください。本資料の「2.5 log4jの設定」をご覧ください。

Q4 フィルタがうまく動作しないのですが、どのような原因が考えられますか？

A4 フィルタは Servlet2.3 仕様から追加された仕組みです。WebOTX V8 では Servlet2.5 に準拠しているため、web.xml 先頭の version を 2.3、2.4 または 2.5 に変更してください。変更後のイメージは次のようになります。

```
<?xml version="1.0" encoding="Shift_JIS"?>
<web-app xmlns="http://java.sun.com/xml/ns/j2ee" version="2.4" xmlns:xsi
="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd
">
```

Q5 getParameter で取得したデータが文字化けするのですが、どのような原因が考えられますか？

A5 Tomcat 6.x では、URI のエンコードの指定のために、server.xml で URIEncoding と useBodyEncodingForURI が指定できますが、WebOTX では、server.http-service.http-listener.http-listener-name.property で指定します。
それぞれ、運用管理コマンド（otxadmin コマンド）を利用して、次のように指定します。

```
otxadmin> set server.http-service.http-listener.http-listener-1.property.uri-
          encoding=Windows-31J

otxadmin> set server.http-service.http-listener.http-listener-1.property.use-
          body-encoding-for-uri=true
```

Q6 Web ブラウザで JSP の出力を参照したとき、機種依存文字（㈱など）が文字化けするのですが、どのような原因が考えられますか？

A6 エンコードの指定を Windows-31J にする必要があります。
JSP の場合には、web.xml の<jsp-config>の指定で次のように定義することができます。

【web.xml】

```
<web-app  ....
  <jsp-config>
    <jsp-property-group>
      ....
      <page-encoding>Windows-31J</page-encoding>
    </jsp-property-group>
  </jsp-config>
  ....
  <servlet>
  ....
  </servlet>
</web-app>
```

そのほかに、Servlet での Content-Type による設定、JSP での page ディレクティブの contentType による設定などに起因する文字化けがあります。 詳細については「WebOTX Web コンテナ チューニングとトラブルシューティング」ー「3.1.5. Web アプリケーションで文字化けが発生する」をご覧ください。

また、WebOTX V8 では文字コードを強制的に指定することができるようになりました。

以下に、Servlet/JSP への入出力、エンコーディングを強制設定する機能について説明します。

1) HTTP リクエスト

エンコーデイングを指定してリクエストのデータをどの文字コードとして

Java で扱う Unicode にデコードするか、 の設定があります。

詳細は 01. HTTPリクエスト 参照

2) HTTP レスポンス

レスポンス出力時、Java で扱う Uniode をどのエンコーデイングを指定して、

意図する文字コードにエンコード(レスポンス出力)するか の設定があります。

詳細は 02. HTTPレスポンス 参照

3) jsp ファイルのコンパイル

jsp ファイルはどの文字コードで記述されているか(どのエンコーディングを

指定して Java で扱う Unicode にデコードするか) の設定があります。

詳細は 03. jspファイルのコンパイル 参照

4) jsp ファイルのレスポンス

JSP を実行した際、Java で扱う Uniode をどのエンコーデイングを指定して、

意図する文字コードにエンコード(レスポンス出力)するか の設定があります。

詳細は 04. jspファイルのレスポンス 参照

5) ファイル入力/出力

ファイルにアクセスする InputStream、OutputStream を作成する際ユーザが

文字コードを指定します。WebOTX 独自の設定機能は存在しませんが Java VM オプション

(-Dfile.encoding=<MS932 等>)指定によりデフォルトキャラセットを変更可能

6) データベースアクセス

JDBC ドライバに依存します。

※各設定で指定した文字コードは Java のコンバータにより処理されます。

01. HTTP リクエスト

リクエストのデータをどの文字コードとして扱うかは通常、以下の処理順で決定されます。

(ここで決定された文字コードはリクエストからデータを取得する際の InputStream に適用されます。)

1) Filter 等でリクエストデータを参照する前に、ServletRequest.setCharcterEncoding() によりエンコーデイングを設定した場合、そのエンコーディングでリクエストデータをデコードします。

2) HTTP リクエストのヘッダ情報に charset 指定がある場合、そのエンコーディングでリクエストデータをデコードします。

"Content-Type: application/x-www-form-urlencoded; charset=windows-31j" 等

3) HTTP リクエストのヘッダ情報に charset 指定が無い場合、nec-web.xml ファイルに <locale-charset-info>要素を設定し、war ファイルに予めアーカイブすることでリクエストデータのデコードに用いるデフォルトのエンコーディングを指定することができます。

※ nec-web.xml の例

```
<locale-charset-info default-locale="ja">
  <locale-charset-map locale="ja" charset="windows-31j"/>
  <parameter-encoding form-hint-field="penc"/>
</locale-charset-info>
```

3-1) リクエストのパラメータ中に該当データ(この場合パラメータ名が penc のデータ penc=x-IBM943C 等)があればそれをエンコーディングとして使用する。

3-2) 3-1)が無い場合、リクエスト中の Accept-Language: ja、無ければ nec-web.xml の <locale-charset-info default-locale="ja">を取得 <locale-charset-map>より locale が一致する map を検索、その charset でリクエストパラメータを Unicode に変換

4) 上記のいずれも無い場合、リクエストデータはデフォルトのエンコーディング"ISO-8859-1"でデコードされます。

02. HTTP レスポンス

レスポンスにデータをどの文字コードとして出力するかは通常、以下の処理順で決定されます。

(ここで決定された文字コードは HTTP レスポンスのヘッダ情報(Content-Type の charset)とデータを出力する際の OutputStream に適用されます)

1) Servlet2.4 の仕様でロケールおよび文字エンコードのマッピングを web.xml に指定できます。javax.servlet.ServletResponse の setLocale()メソッドが呼ばれた時に参照され、指定したロケールに対応する文字コードが設定されます。

※ web.xml 指定例

```
<local-encoding-mapping-list>
  <local-encoding-mapping>
    <locale>ja</locale>
    <encoding>windows-31j</encoding>
  </locale-encoding-mapping>
  <locale-encoding-mapping>
    <locale>ko_KR</locale>
    <encoding>EUC-KR</encoding>
  </locale-encoding-mapping>
</locale-encoding-mapping-list>
```

2) ServletResponse.setContentType()にて charset が指定された場合、そのエンコーディングでエンコード(レスポンス出力)します。

3) 上記の指定が無い場合、デフォルトのエンコーディング"ISO-8859-1"でエンコード(レスポンス出力)します。

03. jsp ファイルのコンパイル

jsp ファイルどの文字コードで記述されているか(読み込み時どのエンコーディングでデコードするか)は通常、以下の処理で決定されます。

(ここで決定されたエンコーディングは jsp ファイルを読み込む際の InputStrem に適用されます)

1) jsp に pageEncoing 指定が記述されている場合、そのエンコーディングで jsp ファイルをデコード(読み込み)します。

2) jsp に pageEncoing 指定が無く contenType 指定(charset 指定)が記述されている場合、そのエンコーディングで jsp ファイルをデコード(読み込み)します。

3) JSP2.0 の仕様で jsp のエンコーディングを web.xml に指定できます。
contenType 指定(charset 指定)が無い場合、web.xml で指定したエンコーディングが適用されます。

※ web.xml 例

```
<jsp-config>
  <jsp-property-group>
    <url-pattern>/jsp/*</url-pattern>
    <page-encoding>Shift_JIS</page-encoding>
  </jsp-property-group>
</jsp-config>
```

4) pageEncoing 指定、contenType 指定が共に無い、web.xml の指定も無い場合、otxadmin コマンドにてデフォルトのエンコーディングを指定することができます。(要ドメイン再起動)

```
otxadmin> set server.web-container.property.default-encoding=Shift_JIS
```

5) 上記のいずれの指定も無い場合、jsp ファイルはデフォルトのエンコーディング"ISO-8859-1"でデコード(読み込み)されます。

※default-web.xml の javaEncoding は jsp->servlet で変換した servlet の java ファイルを

エンコード(出力)する際のエンコーディング指定

※実際のファイルの文字コードが、複数あり、個別のページでエンコーディングを指定して
いる場合、priorityJspInEncoding を指定して特定の文字コードでエンコードするように
設定すると、一部のページで文字化けが発生します。この場合は、該当のページの
ファイルの文字コード修正してください。

04. jsp ファイルのレスポンス

出力するレスポンスデータはどのエンコーディングエンコード(出力)するかは通常、以下の処理で
決定されます。
(ここで決定されたエンコーディングはレスポンスのヘッダ情報(Content-Type の charset)と出力
する際 OutputStream に適用されます)

1) jsp に contenType 指定(charset 指定)が記述されている場合、そのエンコーディングでレスポンス
データをエンコード(出力)します。

2) JSP2.0 の仕様で jsp のエンコーディングを web.xml に指定できます。
contenType 指定(charset 指定)が無い場合、web.xml で指定したエンコーディングが適用され
ます。
※ web.xml 例

```
<jsp-config>
  <jsp-property-group>
    <url-pattern>/jsp/*</url-pattern>
    <page-encoding>Shift_JIS</page-encoding>
  </jsp-property-group>
</jsp-config>
```

3) contenType 指定、web.xml の指定共に無い場合、otxadmin コマンドにてデフォルトの
エンコーディングを指定することができます。(要ドメイン再起動)
otxadmin> set server.web-container.property.default-encoding=Shift_JIS

4) 上記のいずれの指定も無い場合、jsp のレスポンスはデフォルトのエンコーディング"ISO-8859-1"
でエンコード(出力されます。

※1 この機能の指定方法は2パターンあります、全 Web アプリケーション共通の指定と、Web アプリ
ケーション毎に指定するものです。それぞれ domain.xml の<web-container>要素の property、と
war ファイルにアーカイブする nec-web.xml に定義されます。

全アプリケーション共通の設定、Web アプリケーション毎の設定の両方が指定された場合、Web アプリケ
ーション毎の設定が優先されます。
nec-web.xml に指定する場合は、エンコーディングを1つ設定する指定か、locale とエンコーディングを
マッピングさせてエンコーディングを決定する指定の2パターンを可能とします。
両方が指定された場合、locale とエンコーディングのマッピング指定が優先されます。
以下、各設定例を記述します。

===== 設定例 =====

1. レスポンスエンコーディングの優先設定

・Web アプリケーション毎に有効な Locale に対応したエンコーディング指定の nec-web.xml への記述(a1)

例） <nec-web-app>

:

<property name="priorityResponseEncoding-map:ja" value="EUC_JP"/>

</nec-web-app>

※ "priorityResponseEncoding"に"-map"を付加し、":"で区切って Locale を記述します

Locale は HttpResponse の getLocale()で取得できる文字列と同じである必要があります。

・Web アプリケーション毎に有効な nec-web.xml への記述 (a2)

例） <nec-web-app>

:

<property name="priorityResponseEncoding" value="EUC_JP"/>

</nec-web-app>

・全 Web アプリケーションに有効な domain.xml への指定 (b)

例） otxadmin> set server.web-container.property.priority-response-encoding=EUC_JP

※要ドメイン再起動

2. JSP ファイル読み込みエンコーディングの優先指定

・Web アプリケーション毎に有効な nec-web.xml への記述(c)

例） <nec-web-app>

:

<jsp-config>

<property name="priorityJspInEncoding" value="EUC_JP"/>

</jsp-config>

</nec-web-app>

・全 Web アプリケーションに有効な domain.xml への指定(d)

例） otxadmin> set server.web-container.property.priority-jsp-in-encoding=EUC_JP

※要ドメイン再起動

3. JSP ファイル出力時エンコーディングの優先指定

JSP 出力時とはコンパイルした java ソースで HttpServletResponse に setContentType()する際に指定する "charset=XXX"部分に相当します。

・Web アプリケーション毎に有効な Locale に対応したエンコーディング指定の nec-web.xml への記述(e1)

例） <nec-web-app>

:

<jsp-config>

<property name="priorityJspOutEncoding-map:ja" value="EUC_JP"/>

```
    </jsp-config>
  </nec-web-app>
```

※ "priorityJspOutEncoding"に"-map"を付加し、":"で区切って Locale を記述します

Locale は HttpResponse の getLocale()で取得できる文字列と同じである必要があります。

・Web アプリケーション毎に有効な nec-web.xml への記述 (e2)

例)

```
<nec-web-app>
    :
    <jsp-config>
        <property name="priorityJspOutEncoding" value="EUC_JP"/>
    </jsp-config>
</nec-web-app>
```

・全 Web アプリケーションに有効な domain.xml への指定(f)

例) otxadmin> set server.web-container.property.priority-jsp-out-encoding=EUC_JP

※要ドメイン再起動

Q7 Tomcat からの移行時に JSP のコンパイルエラーになるのですが、原因は何が考えられるでしょうか？

A7 Java 1.4 で記述された JSP を Java 1.5 のソースとしてコンパイルした際に、コンパイルエラーが発生します。WebOTX V8 ではデフォルトでは、Java 5 (Java1.5)としてコンパイルするため、この問題が発生します。次の例のように、コンパイルする際に使用する Java の バージョンを指定してください。

<servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>の定義に compilerTargetVM と compilerSourceVM のパラメータで 1.4 を指定する定義を追加し、ドメインを再起動してください。

対象ファイル：(WebOTX インストールディレクトリ)/domains/domain1/config/default-web.xml

```
<servlet>
  <servlet-name>jsp</servlet-name>
  <servlet-class>org.apache.jasper.servlet.JspServlet</servlet-class>
  <init-param>
    <param-name>compilerTargetVM</param-name>
    <param-value>1.4</param-value>
  </init-param>
  <init-param>
    <param-name>compilerSourceVM</param-name>
    <param-value>1.4</param-value>
  </init-param>
</servlet>
```

エラー原因が上記でない場合は、AP でデータ型を明示的に指定することによりエラーを回避できる可能性があります。

AP 修正内容 例)

```
xxxxxxxx a = (xxxxxxxx)form.getViewList().get(idx);
                                  ↓
xxxxxxxx a = (xxxxxxxx)form.getViewList().get(idx.intValue());
```

Q8 Servlet を実行する際に URL を"/<コンテキスト名>/servlet/サーブレットのクラス名"というように実行すると HTTP404 エラーとなるのですが、設定により直接 Servlet を実行する方法はありますか？

A8 Tomcat4.1.12 以降では、次のようにセキュリティの観点からデフォルトでは無効になっています。

WebOTX V8 では、同様にこの機能をデフォルトで無効としています。

有効にするには次の部分2箇所をコメントアウトしているので、利用するように変更し、ドメインを再起動してください。

対象ファイル：(WebOTX インストールディレクトリ)/domains/domain1/config/default-web.xml

```
<servlet>
<servlet-name>invoker</servlet-name>
<servlet-class>org.apache.catalina.servlets.InvokerServlet</servlet-class>
<init-param>
<param-name>debug</param-name>
<param-value>0</param-value>
</init-param>
<load-on-startup>2</load-on-startup>
</servlet>



<servlet-mapping>
<servlet-name>invoker</servlet-name>
<url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

Q9 Servlet の Java プログラムから、ネットワークドライブにあるファイルを開きたいのですが、どのようにしたらよろしいでしょうか？

A9 ネットワークドライブは以下の条件で利用可能となります。

WebOTX がサービス起動の場合は、サービスのログオンユーザの設定と、共有フォルダへのログオンユーザのアクセス権が必要です。また、AP での共有フォルダの指定は、UNC 表記（¥¥hostname¥共有フォルダ名）で指定してください。

＃ D:¥xxxxx¥xxxx の表記では、アクセスできません。

Q10 シンボリックリンクを使用したいのですが、どのようにしたらよろしいでしょうか？

A10 例） WebAPP1 という Web アプリケーション内からリンクする例です

・Web モジュールのディレクトリ

/opt/WebOTX/domains/domain1/applications/j2ee-modules/WebAPP1

・リンク先の対象のファイル

/home/hoge/test.html

1） シンボリックリンクを有効にします。

運用管理コマンドで以下のコマンドを入力します。

./otxadmin set --user admin --password adminadmin server.http-service.virtual-server.server.property.allowLinking=true

./otxadmin set --user admin --password adminadmin server.http-service.virtual-server.server.property.caseSensitive=false

2） ドメインを再起動します。

3） 次のコマンドでハードリンクします。

cd /opt/WebOTX/domains/domain1/applications/j2ee-modules/WebAPP1

ln /home/hoge/test.html t1.html

4） 確認します。

cat ./t1.html

5） /WebAPP1/t1.html にアクセスすると、test.html の内容が表示され、シンボリックリンクを行えていることが確認できます。

※シンボリックリンクは Windows OS では使用できません。Unix 系 OS で使用してください。

Q11 https の通信で Cookie のセキュアオプションがついていないのですが、設定によりつけることはできますか？

A11 内蔵 Web サーバを使用している場合は https での送受信時には自動的に secure 属性が付加されます。
外部 Web サーバを使用している場合は運用管理コマンドで下記を入力した後、ドメインを再起動することで secure 属性を付加することができます。

```
otxadmin> set server.http-service.http-listener.ajp-listener-1.security-enabled=True
```

また、以下の設定により、http の通信でも secure 属性を付加させることができます。
設定内容：
nec-web.xml に下記の記述を行い、Web アプリケーションを更新した後、アプリケーションを配備し直してください

<session-properties>タグに property を記述します。

```
<session-config>
    <session-properties>
        <property name="alwayCookieSecure" value="true"/>
    </session-properties>
</session-config>
```

Q12 Web サーバプラグインのログローテート機能を利用したいのですが、どのようにしたらよろしいでしょうか？

A12 以下の例を参考にファイルを編集し、Web サーバを再起動してください。
対象ファイル：（WebOTX インストールディレクトリ）/domains/domain1/config/WebCont/*.conf-auto

JkLogFile "D:/WebOTX/domains/domain1/logs/webcontainer/mod_jk-20.log"

↓

JkLogFile "|D:/WebOTX/WebServer2/bin/rotatelogs.exe D:/WebOTX/domains/domain1/logs/webcontainer/mod_jk-20.log 86400"

一日（86400 秒）毎にログローテートされます。

※ *.conf-auto ファイルはデフォルト名のままである場合、ドメイン再起動で上書きされます。必要に応じてファイル名をリネームしてください。

5.2. Web アプリケーションの開発に関する Q&A

Q1 servlet をコンパイルするときに、WebOTX ではどの jar を利用するのでしょうか？

A1 ${AS_INSTALL}/lib/javaee.jar を利用してください。

Q2 Web アプリケーションを自動的に配備するには、どうすれば良いのでしょうか？

A2 Tomcat では、webapps ディレクトリに WAR ファイルを置くと自動的に配備されますが、WebOTX では ${ INSTANCE_ROOT}/autodeploy ディレクトリに WAR ファイルを格納することで、自動的に配備できます。

5.3. 環境設定、チューニングに関する Q&A

Q1 WebOTX で JavaVM のメモリ量を指定するには、どうすれば良いのでしょうか？

A1 Tomcat では、起動用のファイルで JavaVM のメモリ量を指定しますが、WebOTX では運用管理コマンド（otxadmin） で指定します。

【割り当てメモリの最大値を 640 MB にする例】

```
otxadmin> create-jvm-options --user <ユーザ名> --password <パスワード> --host <ホスト名> --port <管理ポート> -Xmx640m:
```

詳細については、WebOTX V8 マニュアル「運用編」-「6.チューニング」-「2.3.1.ドメインエージェントプロセスのヒープサイズの見直し」をご覧ください。

Q2 Apache と連携するときの同時接続（スレッド）数の設定は、どうすれば良いのでしょうか？

A2 Tomcat では server.xml を編集して設定しますが、WebOTX では運用管理コマンド（otxadmin） で指定します。

【同時に処理できるリクエストの数を拡大する例】

```
otxadmin> set --user <ユーザ名> --password <パスワード> --host <ホスト名> --port <管理ポート> server.http-service.http-listener.<リスナ ID>.max-processors=<最大数>
```

詳細については、WebOTX V8 マニュアル 「運用編」-「6.チューニング」-「4.1.1.最大プロセッサ数」をご覧ください。

Q3 JDBC データソースの設定は、どうすれば良いのでしょうか？

A3 Tomcatでは server.xml を編集して設定しますが、WebOTXでの設定方法は本資料の「2.4 JDBCデータソースの設定」をご覧ください。

Q4 1台のサーバで複数の Web コンテナを起動するには、どうすれば良いのでしょうか？

A4 Tomcat では、Tomcat のディレクトリをコピーして、同じマシンで複数の Tomcat を起動しますが、WebOTX では複数のドメインを作成することで複数の Web コンテナを起動することができます。

Q5 Tomcat からの移行時に対応する環境設定を教えてください。

A5 ・ JAVA_HOME、

JAVA_HOME は、インストール時に「JDK がインストールされているディレクトリ」を指定していると思います。

・ CATALINA_OPTS、

Java VM オプションは、次の箇所に設定してください。

■シングルプロセスモードの場合

運用管理コンソールでドメイン⇒アプリケーションサーバ⇒JVM 構成⇒属性

”JVM オプション” に設定してください

■マルチプロセスモードの場合

運用管理コンソールでドメイン⇒TP システム⇒アプリケーショングループ⇒＜アプリケーショングループ名＞

⇒プロセスグループ⇒＜プロセスグループ名＞

「属性」タブの「「Java システムプロパティ」に設定してください。

既に設定されている Java VM のメモリ指定等は編集してください。

新規の項目は、追加してください。

・ SHILIB_PATH、LD_LIBRARY_PATH、CONFIG_PATH

既存の環境のとおり OS の環境変数に設定してください。

・ CLASSPATH

クラスパスは、次の箇所に設定してください。

■シングルプロセスモードの場合

運用管理コンソールでドメイン⇒アプリケーションサーバ⇒JVM 構成⇒属性

”サーバのクラスパス” に設定してください

■マルチプロセスモードの場合

運用管理コンソールでドメイン⇒TP システム⇒アプリケーショングループ⇒＜アプリケーショングループ名＞

⇒プロセスグループ⇒＜プロセスグループ名＞

「属性」タブの「「環境変数」に設定してください。

※環境変数の詳細については、WebOTX V8 マニュアル「運用編」-「3.運用と操作」-「3.システムに関する設定」-「3.1 システム環境変数」をご覧ください。

6. 注意事項

6.1. J2SE SDK に関する注意事項

WebOTX が動作保証する J2SE SDK(JDK)に変更する場合、Web アプリケーションや Web アプリケーションが依存するライブラリが、特定の J2SE SDK に依存しないか動作確認を行ってください。

6.2. Tomcat との差異

- WebOTX では WAR ファイル名から拡張子を除いた名称と、配備時に指定したコンテキスト名が異なった場合、WAR ファイル名から拡張子を除いた名称をコンテキスト名としては利用できません。
- WebOTX では、web.xml ファイルから Web アプリケーションの配備情報を取得しているため、web.xml ファイルは必須です。web.xml ファイルがない、もしくは、正しく記述されていない場合、WAR ファイルを配備することができません。

機能比較		
■YES、□NO	**Tomcat 6.0.x**	**WebOTX 8.1**
仕様		
Java Servlet	2.5	2.5
JavaServer Pages	2.1	2.1
JSF	-	1.2
JSTL	-	1.2
Java EE 5 対応	□	■
パフォーマンス/チューニング		
リクエスト処理スレッドの動的制御	■	■
NewI/O を利用した高速なリクエスト処理	■（独自実装）	■（Grizzly を利用）
統計情報の提供（詳細は別途）	■	■
プロファイラの提供	□	■
スケーラビリティ		
ハードウェアを使用した負荷分散	■	■
ソフトウェアを使用した負荷分散	■	■
セッションのレプリケーション	■(file,database, TCP)	■（JNDI）
仮想ホストのサポート	■	■
マルチプロセスモードのサポート	□	■
セキュリティ		
SSL (通信)	■	■
SSL (証明書の管理と運用)	□	■
認証		
- BASIC 認証	■	■

- フォームベース	■	■
- クライアント証明書	■	■
- DIGEST 認証	■	■
レルム		
- JDBC レルム	■	■（JDBCRealm）
- DataSource レルム	■	■（JDBCRealm で同じ事を実現可能）
- JNDI レルム	■	□
- Memory レルム	■	■（fileRealm）
- JAAS レルム	■	□
- LDAP レルム	□	■
ログ		
ローテーション	■（サイズ、時間、個数を指定可能）	■（log4j を採用しているので柔軟なカスタマイズが可能）
HTTP のログ採取	■	■（アクセスログにて細かく指定可能）
Web アプリケーションの実行		
クラスローダの優先順位変更	■	■
ユーザ独自のクラスローダ利用	■	■
外部 Web サーバ連携時のコンテキスト動的反映	□	■（ON/OFF/一回のみ実行を指定可能）
国際化		
リクエストデータの文字エンコーディング		
- ServletRequest.setCharacterEncoding()	■	■
- 設定ファイルでの指定	■	■
レスポンスデータの文字エンコーディング		
- ServletResponse.setContentType()	■	■
- JSP での page ディレクティブ（contentType）	■	■
- 優先指定（上記の指摘より優先する）（JSP の場合だけ有効）	□	■
- 優先指定（全ての指摘より優先する）（Servlet と JSP で有効）	□	■
JSP の文字エンコーディング		
- page ディレクティブ（pageEncoding）	■	■
- 設定ファイルでのデフォルト値の指定	■	■
- 優先指定（上記の指摘より優先する）	□	■

運用管理（コンテナ）		
初期設定ツール	□	■
外部 Web サーバとの連携設定	□（手動で設定）	■
リモートからの起動/停止	□	■
運用管理（Web アプリケーション）		
配備/配備解除/起動/停止などの運用		
- ツールによる配備/配備解除	■	■
- コマンドによる配備/配備解除	□	■
- autodeploy	□	■
アクセス中クライアントの情報表示	□	■
任意のコンテキストで配備	■（独自の配備記述子で指定）	■
コンフィグレーション		
コネクタ（共通）		
- allowTrace	■	■
- emptySessionPath	■	■
- enableLookups	■	■
- maxPostSize	■	■
- maxSavePostSize	■	■
- protocol	■	■
- proxyName	■	■
- proxyPort	■	■
- redirectPort	■	■
- SSLEnabled	■	■
- scheme	■	■
- secure	■	■
- URIEncoding	■	■
- useBodyEncodingForURI	■	■
- useIPVHosts	■	■
- xpoweredBy	■	■
コネクタ（HTTP）		
- acceptCount	■	■
- address	■	■
- bufferSize	■	■
- compressableMimeType	■	■
- compression	■	■
- connectionLinger	■	■
- connectionTimeout	■	■
- executor	■	■

- keepAliveTimeout	■	■
- disableUploadTimeout	■	■
- maxHttpHeaderSize	■	■
- maxKeepAliveRequests	■	■
- maxThreads	■	■
- noCompressionUserAgents	■	■
- port	■	■
- restrictedUserAgents	■	■
- server	■	■
- socketBuffer	■	■
- tcpNoDelay	■	■
- threadPriority	■	■
コネクタ（SSL）		
- algorithm	■	□
- clientAuth	■	■
- keystoreFile	■	■
- keystorePass	■	■
- keystoreType	■	□
- keystoreProvider	■	□
- sslProtocol	■	■
- ciphers	■	■
- keyAlias	■	□
- truststoreFile	■	■
- truststorePass	■	■
- truststoreType	■	□
- truststoreProvider	■	□
コネクタ（AJP-共通）		
- allowTrace	■	■
- emptySessionPath	■	■
- enableLookups	■	■
- maxPostSize	■	■
- maxSavePostSize	■	■
- protocol	■	■
- proxyName	■	■
- proxyPort	■	■
- redirectPort	■	■
- request.registerRequests	■	□
- scheme	■	■

- secure	■	■
- URIEncoding	■	■
- useBodyEncodingForURI	■	■
- useIPVHosts	■	■
- xpoweredBy	■	■
コネクタ（AJP-Standard）		
- address	■	■
- backlog	■	■
- bufferSize	■	■
- connectionTimeout	■	■
- executor	■	■
- keepAliveTimeout	■	■
- maxThreads	■	■
- packetSize	■	■
- port	■	■
- request.secret	■	□
- request.shutdownEnabled	■	□
- request.useSecret	■	□
- tcpNoDelay	■	■
- tomcatAuthentication	■	■
コンテキスト（共通）		
- backgroundProcessorDelay	■	■
- className	■	□
- cookies	■	■
- crossContext	■	■
- docBase	■	■
- override	■	■
- privileged	■	■
- path	■	■
- reloadable	■	■
- wrapperClass	■	□
コンテキスト（Standard）		
- allowLinking	■	■
- antiJARLocking	■	■
- antiResourceLocking	■	■
- cacheMaxSize	■	■
- cacheTTL	■	■
- cachingAllowed	■	■

- caseSensitive	■	■
- processTlds	■	■
- swallowOutput	■	■
- tldNamespaceAware	■	■
- tldValidation	■	■
- unloadDelay	■	■
- unpackWAR	■	■
- useNaming	■	□
- workDir	■	■
Manager（共通）		
- className	■	□
- distributable	■	■
Manager（Standard）		
- algorithm	■	□
- entropy	■	□
- maxActiveSessions	■	■
- maxInactiveInterval	■	■
- pathname	■	■
- processExpiresFrequency	■	□
- randomClass	■	□
- sessionIdLength	■	□
Realm（共通）		
- classname	■	■
Realm（JDBC）		
- connectionName	■	■
- connectionPassword	■	■
- connectionURL	■	■
- digest	■	■
- digestEncoding	■	■
- driverName	■	■
- roleNameCol	■	■
- userCredCol	■	■
- userNameCol	■	■
- userRoleTable	■	■
- userTable	■	■
Realm（DataSource）		
- dataSourceName	■	□
- digest	■	□

- localDataSource	■	□
- roleNameCol	■	□
- userCredCol	■	□
- userNameCol	■	□
- userRoleTable	■	□
- userTable	■	□
Realm（JNDI）		
- alternateURL	■	□
- authentication	■	□
- connectionName	■	□
- connectionPassword	■	□
- connectionURL	■	□
- contextFactory	■	□
- derefAliases	■	□
- digest	■	□
- protocol	■	□
- roleBase	■	□
- roleName	■	□
- roleSearch	■	□
- roleSubtree	■	□
- userBase	■	□
- userPassword	■	□
- userPattern	■	□
- userRoleName	■	□
- userSearch	■	□
- userSubtree	■	□
Realm（Memory）		
- digest	■	□
- pathname	■	■
システムプロパティ（EL）		
- org.apache.el.parser.COERCE_TO_ZERO	□	□
システムプロパティ（Jasper）		

- org.apache.jasper.compiler.Generator.VAR_EXPRESSIONFACTORY	■	■
- org.apache.jasper.compiler.Generator.VAR_INSTANCEMANAGER	□	□
- org.apache.jasper.compiler.Parser.STRICT_QUOTE_ESCAPING	□	□
- org.apache.jasper.runtime.BodyContentImpl.LIMIT_BUFFER	■	■
- org.apache.jasper.runtime.JspFactoryImpl.USE_POOL	■	■
- org.apache.jasper.runtime.JspFactoryImpl.POOL_SIZE	■	■
- org.apache.jasper.Constants.JSP_SERVLET_BASE	■	■
- org.apache.jasper.Constants.SERVICE_METHOD_NAME	■	■
- org.apache.jasper.Constants.SERVLET_CLASSPATH	■	■
- org.apache.jasper.Constants. JSP_FILE	■	■
- org.apache.jasper.Constants.PRECOMPILE	■	■
- org.apache.jasper.Constants.JSP_PACKAGE_NAME	■	■
- org.apache.jasper.Constants.TAG_FILE_PACKAGE_NAME	■	■
- org.apache.jasper.Constants.ALT_DD_ATTR	■	■
- org.apache.jasper.Constants.TEMP_VARIABLE_NAME_PREFIX	■	■
- org.apache.jasper.Constants.USE_INSTANCE_MANAGER_FOR_TAGS	□	□
システムプロパティ（Security）		
- org.apache.catalina.connector.RECYCLE_FACADES	■	■
- org.apache.catalina.connector.CoyoteAdapter.ALLOW_BACKSLASH	■	■
- org.apache.tomcat.util.buf.UDecoder.ALLOW_ENCODED_SLASH	■	■

- org.apache.coyote.USE_CUSTOM_STATUS_MSG_IN_HEADER	□	□
統計情報の詳細		
HTTP リスナ		
- 受信バイト数	□	■
- 送信バイト数	□	■
- オープンコネクション数	□	■
- オープンコネクションの最大数	□	■
- 現在のスレッド数	□	■
- 現在のビジースレッド数	□	■
- 最大スレッド数	□	■
- 最大スペアスレッド数	□	■
- 最小スレッド数	□	■
- リクエスト回数	□	■
- エラーリクエスト回数	□	■
- 200 番台のレスポンス数	□	■
- 300 番台のレスポンス数	□	■
- 400 番台のレスポンス数	□	■
- 500 番台のレスポンス数	□	■
- 200-500 番台のレスポンス数	□	■
- 200 のレスポンス数	□	■
- 302 のレスポンス数	□	■
- 304 のレスポンス数	□	■
- 400 のレスポンス数	□	■
- 401 のレスポンス数	□	■
- 403 のレスポンス数	□	■
- 404 のレスポンス数	□	■
- 503 のレスポンス数	□	■
- 最大レスポンス時間	□	■
- 累積処理時間	□	■
Web モジュール		
- アクティブなセッション数	□	■
- リクエスト回数	□	■
- エラーリクエスト回数	□	■
- アクティブなセッションの最大数	■	■
- 有効切れセッションの累計数	□	■
- JSP のロード数	□	■

- JSP のエラー回数	□	■
- JSP のリロード回数	□	■
- 拒否されたセッションの累計数	□	■
- 生成されたセッションの累計数	□	■
- トータルリクエスト処理時間	□	■
- 最大処理時間	□	■
- 最小処理時間	□	■
- リクエスト処理時間	□	■
- サーブレット累計処理時間	□	■
サーブレット		
- リクエスト回数	□	■
- エラーリクエスト回数	□	■
- 最大処理時間	□	■
- 最小処理時間	□	■
- トータルリクエスト処理時間	□	■
- サービス実行最大時間	□	■

6.3. Tomcat ポート番号との対応表

Tomcat 6.x、WebOTX V6、V7、V8 で使用するポート番号（デフォルト）の一覧を下記に示します。

	Tomcat 6.x	WebOTX V6.x	WebOTX V7.x	WebOTX V8.x
HTTP/1.1	8080	80	80	80
AJP/1.3	8009	8009	8009	8099
運用管理コンソール	8080	4848	4848	5858
HTTPS	8443	443	443	443
終了待ち受けポート	8005	–	–	–
IIOP リスナ（マルチプロセスモード使用時）	–	5151	5151	5151
運用管理ポート	–	6202	6202	6202

信頼性、柔軟性、サポート
3つの安心でお客様のシステムを支えます。

をどうぞよろしくお願いします。

■ お問合わせ先

NEC 第二システムソフトウェア事業部

http://www.nec.co.jp/WebOTX/feedback.html

■ 製品ホームページURL

http://www.nec.co.jp/WebOTX/