



WDS*c*によるRPGプログラミング

© 2003 IBM Japan Systems Engineering Co.,Ltd.



特記事項

当資料で解説される項目の更に詳細な説明は、製品から提供されるマニュアル、オンラインヘルプ、Web上の情報を参照してください。

当資料は、2002年8月現在のIBMその他の製品情報に基づいて作成されております。この資料に含まれる情報は可能な限り正確を期しておりますが、日本アイ・ビー・エム株式会社による正式なレビューは受けておらず、当資料に記載された内容に関して日本アイ・ビー・エム株式会社および日本アイ・ビー・エム システムズ・エンジニアリング株式会社が何ら保証をするものではありません。したがって、この情報の利用またはこれらの技法の実施はひとえに使用者の責任においてなされるものであり、当資料の内容によって受けたいかなる被害に関して一切の保証をするものではありませんのでご了承ください。

日本IBMシステムズ・エンジニアリング株式会社サーバー・システム部
日本アイ・ビー・エム株式会社
SWSC (システム&ウェブ・ソリューション・センター)


The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

商標

ibm.com/eserver/iseries

以下の用語は、アメリカ合衆国、あるいは他国、あるいは両国でのIBM Corporationの商標です:

- AS/400
- AS/400e
- DB2
- IBM
- MQSeries
- Operating System/400
- OS/400
- SanFrancisco
- stylized 
- WebSphere
- 400
- iSeries
- eServer

以下の用語は、アメリカ合衆国、あるいは他国、あるいは両国でのLotus Development社の商標です:

- Domino
- Domino.Doc
- LearningSpace
- Lotus
- QuickPlace
- Sametime

JavaとすべてのJavaをベースとする商標およびロゴは、アメリカ合衆国、他国、あるいは両国のサン・マイクロシステムズ社の商標または登録商標です。

Microsoft Windows, Windows NT, およびWindowsのロゴは、アメリカ合衆国、他国、あるいは両国のマイクロソフト社の商標です。

他の会社、製品、およびサービス名は、その会社の商標あるいはサービスマークかもしれません。

このプレゼンテーションに含まれるサードパーティーに関連する題材は、これらのサードパーティーから得られた情報に基づいています。これらの情報の正確さの確認のための、いかなる努力もなされていません。このプレゼンテーションは、いかなるサードパーティー製品またはサービスの、IBMによる推薦あるいは指示を表したり、ほのめかすものではありません。



目次

1. iSeriesにおけるWebアプリケーション開発環境
2. Webアプリケーション開発概説
3. WDSc V5による開発

1. iSeriesにおけるWebアプリケーション開発環境(開発ツール)について理解する。
2. iSeriesにおけるWebアプリケーションの開発手順について理解する。
3. 開発ツールWDScでの開発手順について理解する。

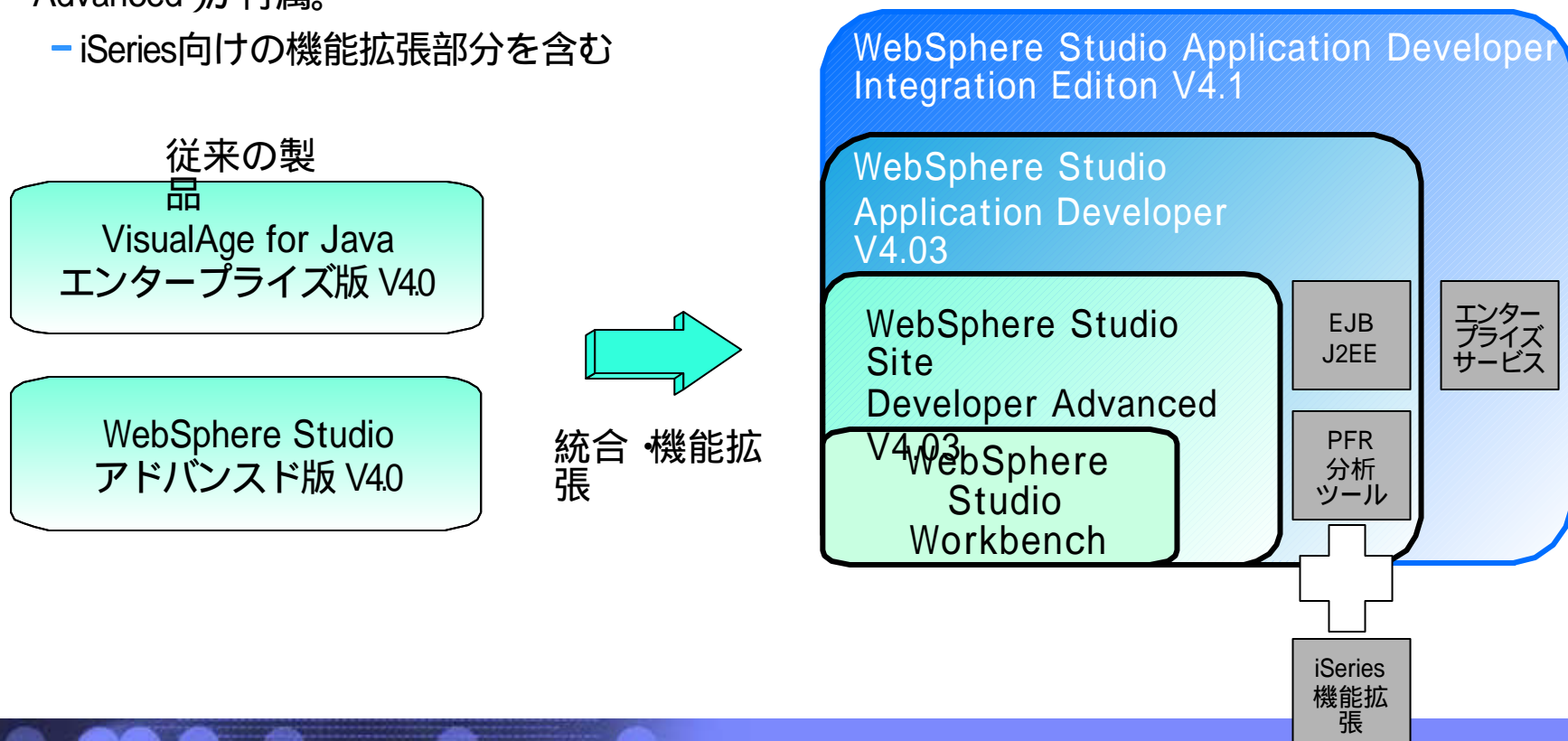
第 1章.iSeriesにおけるWebアプリケーション開発環境



WebSphere開発環境

WebSphere Studio製品概要

- ▶ 3つの製品
- ▶ iSeriesの開発ライセンス (5722-WDS)にはWSSDa(WebSphere Studio Site Developer Advanced)が付属。
 - iSeries向けの機能拡張部分を含む



The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

- WebSphere Studioの最新版は従来のVisualAge for Java V4.0とWebSphere Studio V4.0を統合した製品となりました。
 - 従来の個別のツール、個別の操作方法 統一した操作手順、開発手順
 - それぞれの環境で個別にソース管理 ソースを一元管理
 - WAS V3.5レベルのテスト機能 WAS V4.0レベルのテスト機能
 - WebSphere Studio Workbenchと呼称するオープン・ツール統合プラットフォームの採用
 - WebSphere Studio Workbenchのコア・テクノロジーはeclipse Open Source Project (開発ツールのオープンソース プロジェクト)と同じ
 - 現時点ではeclipse V1.0ベース。 (次期WebSphere Studio Workbenchはeclipse V2.0ベースを予定)
 - Windows 版とLinux版があり共通の開発環境を提供
 - eclipse用に開発されたIBM製以外のプラグインをWebSphere Studio Workbenchに組み込んで使用する事が可能
- iSeriesの開発ライセンス (5722-WDS : WebSphere Development Studio)にはWSSDa (WebSphere Studio Site Developer Advanced)が同梱
 - 通常版のWSSDaにiSeries用の機能拡張部分を追加。
 - WSSDaではEJBの開発ができません。(EJB,J2EE開発には別途WSAD(WebSphere Studio Application Developer)等が必要。)

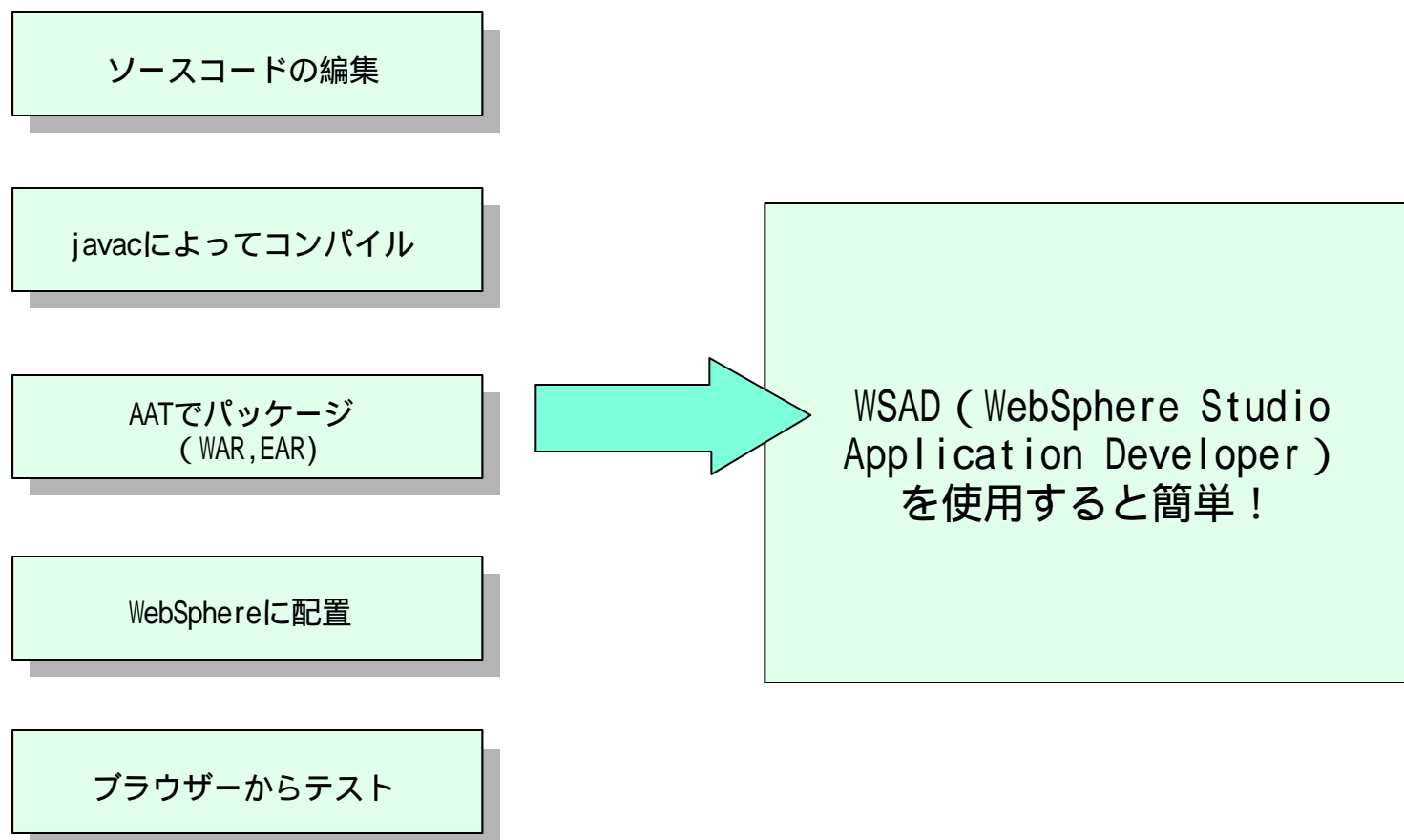
	WebSphere Studio Site Developer Advanced V4.03	WebSphere Studio Application Developer V4.03	WebSphere Studio Application Developer Integration Edition V4.1
位置づけ	エントリーレベルのe-ビジネス・アプリケーション開発用	本格的なe-ビジネス・アプリケーション開発用	WebSphere Studio Application Developerの機能に加え、エンタープライズ・サービスのサポート用
Web開発ツール			
Java開発ツール			
Webサービス開発ツール			
XMLツール			
WebSphereテスト環境			
チーム開発			
データベース・ツール			
Webページ分析ツール			
EJB開発/テスト			
パフォーマンス分析ツール			
エンタープライズ・サービス			

The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

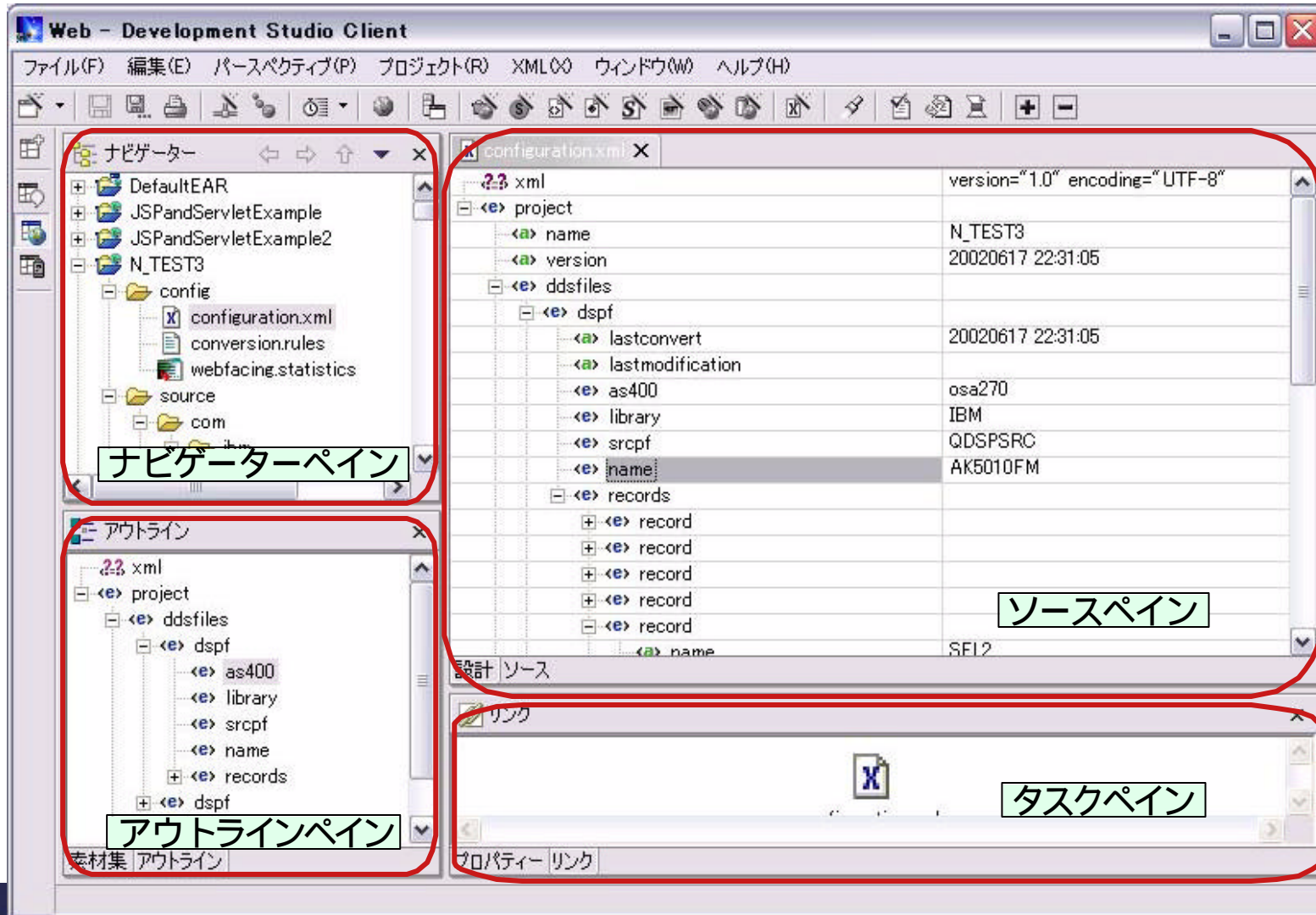
WebSphere Studioによる開発

▪ (例) サブレットをWASで実行させるまでの開発ステップ



- サブレットの開発は通常、下記ようになります。
 - 1. ソースコード編集。サブレット、HTML、JSPなど必要なモジュールのソースコードをエディターで作成
 - 2. コンパイル。javacでコンパイル
 - 3. AATでパッケージング。AATというツールを使用してWASにインストールするためのWARファイル、EARファイルにパッケージ。
 - 4. WASに配置。アプリケーションをWASコンソールからインストールします。
 - 5. ブラウザーからのテスト。テストを実行します。
 - 上記で、45に關しても、WSADはWASローカルテスト環境でテストを実行できます。
- WebSphere Studioの特徴
 - アプリケーション開発環境をシームレスにサポート
 - 設計、プログラム開発、テストを単一インターフェースから実行可能
 - オープンソース(eclipse)をベースにした強力な Java 開発環境
 - プラグインによる機能拡張性
 - パースペクティブと呼ばれるカスタマイズ可能な開発環境を提供
 - WebアプリケーションとJ2EE開発環境 (WSAD以上)
 - ページデザイナー(JSP/HTML)
 - ウィザード(サブレット/EJB)
 - WebSphereテスト環境、Tomcatテスト環境
 - Webサービス(UDDI, SOAP, WSDL)アプリケーション開発
 - DDL作成、SQL照会ビルダー、プロファイラー、XML開発環境
 - 柔軟なチーム環境 CVS、ClearCaseLT (同梱)、他を利用可能

WebSphere Studio : ワークベンチ



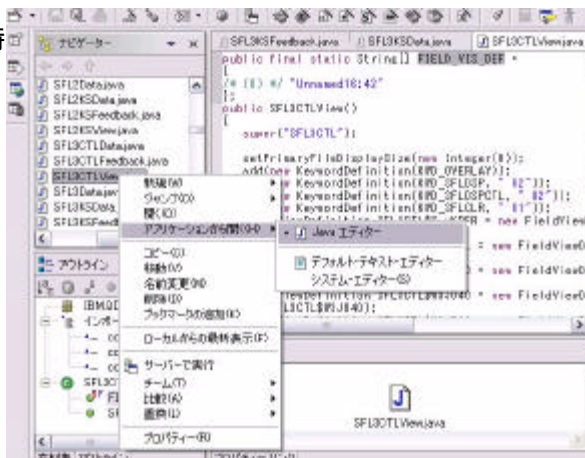
The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

■ ワークベンチ

- WebSphere Studioにおける初期画面であり、この画面からあらゆる開発機能にアクセスする事ができます。
- ワークベンチは大きく4つのウィンドウから構成されています。(後述するパースペクティブによって多少変化します。)おおよかに以下のような情報を表示します。
 - ナビゲーターペイン = 開発中のアプリケーションに含まれるリソースが一覧で表示されます。
 - ソースペイン = 編集中のファイルの内容・ソース等が表示されます。前ページの例ではxmlファイルの編集画面が表示されています。(xmlの場合、テキストイメージとxmlタグを解釈して表示するモードと2つのモードを切り替え可能)
 - アウトラインペイン = で編集中のファイルのアウトライン構造が表示されます。ソースペインで修正してる行が変わると連動して アウトラインペインも移動します。逆に である行をクリックすると ソースペインに選択行の情報が表示されます。
 - タスクペイン = ソースペインの編集に必要な各種の補助機能を表示させることが出来ます。(サブレットコンパイル時のコンパイルエラー、jsp編集時のイメージファイル表示等)
- WebSphere Studioで採用されているワークベンチ (WebSphere Studio Workbench)はeclipseオープンソース・プロジェクトと共通の機能で開発されています。
 - eclipse用に開発されたサードベンダー製プラグインを組み込み可能。(Pluggable Platform :プラグابلプラットフォーム)
 - 開発者の趣向に合わせて柔軟なカスタマイズが可能 (パースペクティブのカスタマイズ)
- 作成しているアプリケーションコンポーネントのタイプ毎に自動的に各ペインの表示内容、メニュー等が切り替わります。
- ユーザーは開発中のコンポーネントのタイプに関係なく同一の操作で開発を進める事ができます。
 - 右クリック エディター を選択すると、コンポーネントに合ったデフォルトエディター (または指定のエディター) が起動する、等

サブレット編集時



JSP編集時



The next generation iSeries...simplicity in an on demand world

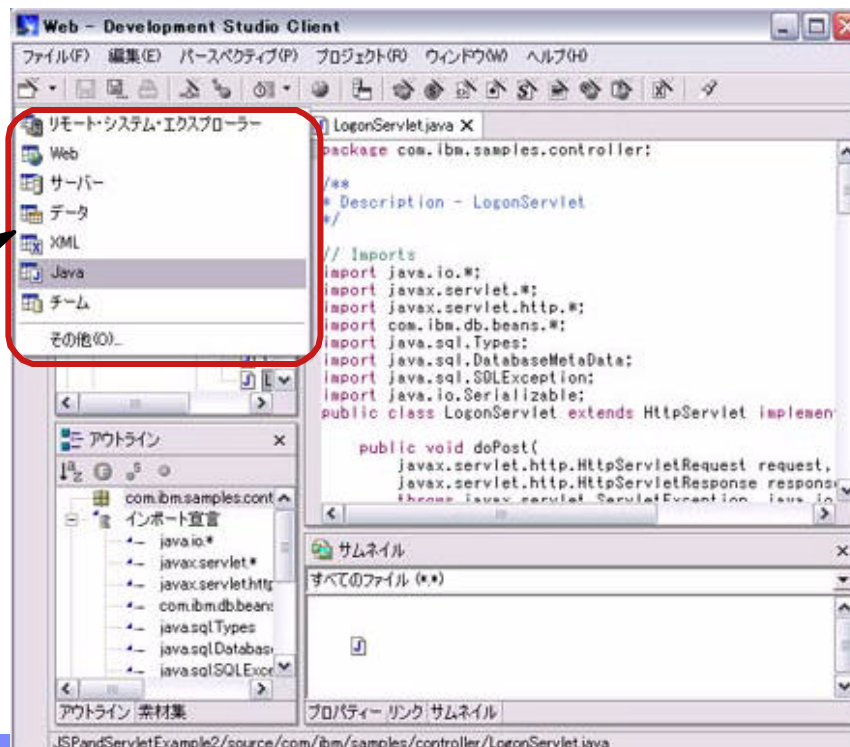
© 2003 IBM Japan Systems Engineering Co.,Ltd.

WebSphere Studio : パースペクティブ

■ パースペクティブ (Perspective)

- ▶ 開発者毎/アプリケーション開発作業毎に適した開発環境 (ビュー) を提供する機能
 - 単一の開発用データリポジトリを利用する人に応じて最適な形式で表示する事が可能
 - ビューとワークベンチ・ウィンドウのレイアウトの初期セットとして提供される
 - ◆ Javaアプリケーション・プログラマーのビュー
 - ◆ ページデザイナー(JSP,HTML)のビュー
 - ◆ J2EEアプリケーションアセンブラーのビュー
 - ◆ 使用者毎にカスタマイズしたビュー
 - ◆ :

パースペクティブ選択メニュー



The next generation iSeries...simplicity in an on demand world

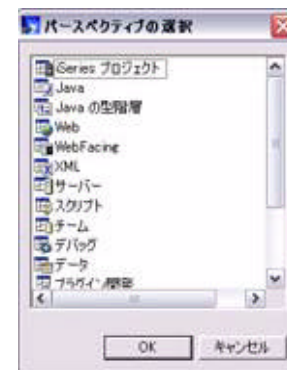
© 2003 IBM Japan Systems Engineering Co.,Ltd.

■ パースペクティブの種類

- 現在以下のパースペクティブが提供されています。パースペクティブ毎に異なった (それぞれの開発に最適化された) 画面が表示されます。パースペクティブは単一のワークステーション上で複数開始する事ができ、画面を切り替えて使用できます。

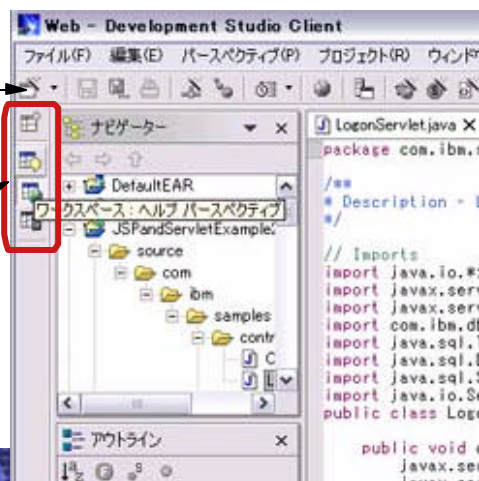
- J2EE J2EEアプリケーション開発に特化 (WSAD以上)
- Java Javaアプリケーションに特化
- Javaの型階層型 Javaアプリケーション/コンポーネントの型階層表示に特化
- Web Weアプリケーション開発に特化
- XML XMLアプリケーション開発に特化
- サーバー ビルトインWAS環境稼働設定に特化
- スクリプト スクリプト開発に特化
- チーム チーム開発環境に特化
- デバッグ アプリケーション・デバッグに特化
- データ データベース開発関連に特化
- プラグイン開発 プラグイン開発に特化
- プロファイル アプリケーションプロファイリングに特化
- ヘルプ ヘルプ表示に特化
- リソース リソースの関係表示に特化
- iSeriesプロジェクト iSeries開発用に特化 (WDSにのみ付属)

■ パースペクティブの切り替えは自動的にされる場合と手動で切り替える場合があります。

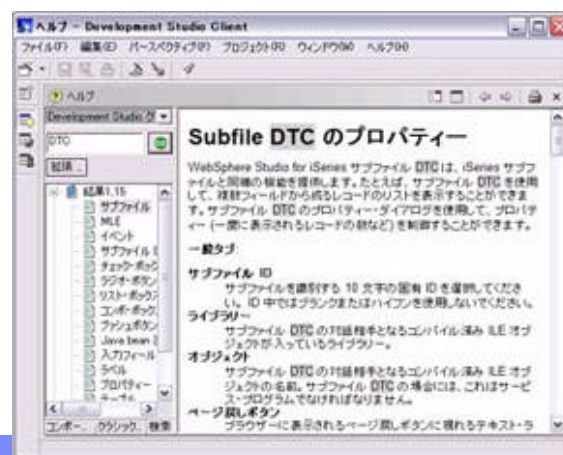


新規パースペクティブの開始

パースペクティブの切り替えアイコン
現在3つのパースペクティブを起動中



パースペクティブの切り替え



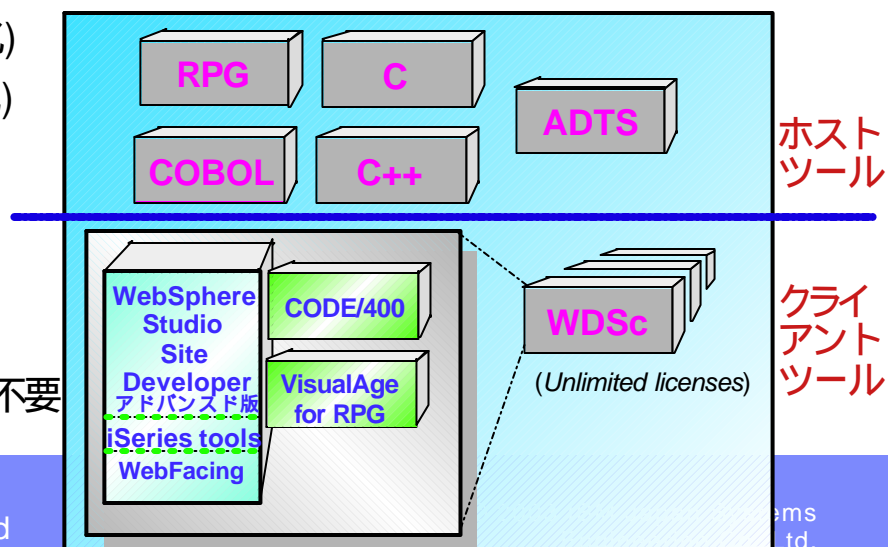
The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

iSeriesにおけるアプリケーション開発ツール

■ WebSphere Development Studio for iSeries (5722-WDS)

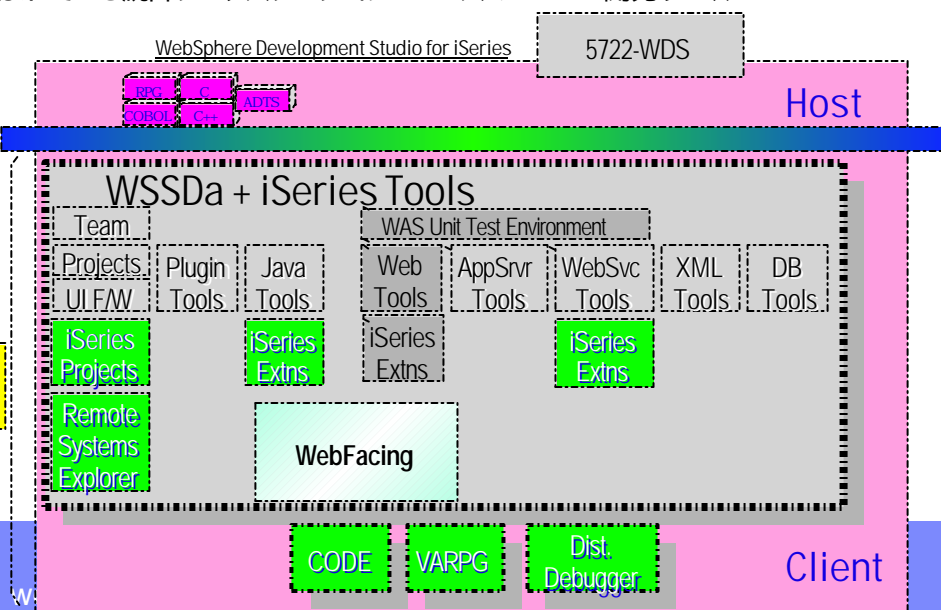
- ▶ 全ての開発用言語、ツールを単一のライセンスプログラムで提供
- ▶ ホスト(iSeries)側ツール
 - ILE RPG, ILE COBOL, ILE C, ILE C++, 適用業務開発ツール セット (ADTS)
- ▶ クライアント側ツール
 - WDSc (WebSphere Development Studio Client for iSeries) V4 (クライアントツールの総称)
 - WebSphere Studio Site Developer Advanced V4.0.3
 - ◆ WebFacing Tool (Series専用)
 - ◆ Java 開発ツール(iSeries 固有の機能強化)
 - ◆ Web 開発ツール(iSeries 固有の機能強化)
 - ◆ サーバー開発ツール等
 - CODE/400
 - VisualAge RPG
 - Distributed debugger (分散デバッガー)
- ▶ 開発したアプリケーションは実行時ライセンス料金不要



The next generation iSeries...simplicity in an on demand world

- WDSc (WebSphere Development Studio for iSeries) V4 は、以下のホストおよびワークステーション ツールで構成されています。
- ホスト (Series)用ツール
 - ILE RPG
 - ILE COBOL
 - ILE C
 - ILE C++
 - 適用業務開発ツール セット (ADTS)
- クライアント側ツール
 - ワークステーション用ツールはOS/400 V5R1リフレッシュ版 (2002年7月より出荷)から新しくなりました。名称も従来のWDT(WebSphere Development Tool)から WDSc (WebSphere Development Studio Client for iSeries) V4 と変更されました。WDScには以下のツールが含まれます。
 - WebSphere Studio Site Developer Advanced (WSSDa) V4.0.3
 - Java 開発ツール(iSeries 固有の機能強化を含む)
 - Web 開発ツール(iSeries 固有の機能強化を含む)
 - 5250 アプリケーションを Web 対応させる IBM WebFacing Tool (Series専用機能)
 - リモート・システム・エクスプローラ と LPEX エディターが組み込まれている統合ワークステーション ベースの iSeries 開発ツール
 - Web サービス開発ツール
 - XML 開発ツール
 - データベース開発ツール
 - WebSphere テスト環境が組み込まれている サーバー開発ツール
 - VisualAge RPG
 - 従来のバージョンと同一
 - Distributed debugger (分散デバッガー)
 - 従来のバージョンと同一
 - CODE
- 開発したアプリケーションは実行時ライセンス料金が不要です。

WDSc



The next generation iSeries...simplicity in an on demand world

WDS*c* iSeries専用拡張機能

- ビジネスロジックをRPG,COBOLで記述したWebアプリケーションの作成
 - ▶ Web**対話ウィザード**で以下の2つをリンクするWebアプリケーションを生成
 - DTC(デザイン・タイム・コントロール)を使用したJSP
 - パラメーター渡しのRPG,COBOLプログラム
 - PCML CALLにより実行(バッチ型ジョブとして実行)
- 5250アプリケーション to Webアプリケーション コンバーター
 - ▶ WebFacing Tool
 - 5250対話型アプリケーションをWeb対応に自動変換
 - 対話型RPG, COBOL, CLをブラウザ、5250エミュレーター双方からアクセス可能に
 - WebFacing Tool 1st Editionから機能拡張
 - WDS*c*上にWebFacing Toolを統合

- WDS*c*のiSeries専用拡張機能の説明
- ビジネスロジックをRPG,COBOLで記述したWebアプリケーションの作成
 - Web**対話ウィザード**で以下の2つをリンクするWebアプリケーションを生成
 - DTC(デザイン・タイム・コントロール)を使用したJSP
 - パラメーター渡しのRPG,COBOLプログラム
 - ・ビジネスロジックはRPG,COBOLで記述します。(Web対話ウィザードで生成したWebアプリケーション(Java)上には一切ビジネスロジックはありません。従ってRPG,COBOLの開発・保守スキルだけでWebアプリケーションを開発する事が可能となります。
 - DTCを使用したJSPとパラメーター渡しのRPG,COBOL間をリンクするWebアプリケーションは自動生成されます。
 - ・Web対話ウィザードで生成されたJavaのWebアプリケーションは画面制御、パラメーターをJSPとビジネスロジックプログラム(RPG,COBOL)間で渡すだけのもので、ビジネスロジックは存在しません。)通常、Web対話ウィザードで自動生成されたアプリケーションプログラムの修正等メンテナンスは不要です。
 - パラメーター渡しのRPG,COBOLはPCML CALLにより実行(バッチ型ジョブとして実行)されるため、iSeriesの対話型CPUカードを必要としません。
- 5250アプリケーション to Webアプリケーション コンバーター
 - WebFacing Tool
 - 5250対話型アプリケーションをWebアプリケーションに自動変換
 - 対話型RPG, COBOL, CLをブラウザ、5250エミュレーター双方からアクセス可能に拡張します。
 - WebFacing Tool 1st Editionから機能拡張
 - ・DDSキーワードサポートの拡張
 - ・WAS V4.0のサポート
 - ・完全な各国語対応
 - ・認証の改良
 - ・WASテスト環境の完全サポート
 - ・ファンクションキー (PFキー)のサポート
 - ・パフォーマンスの改善(伝送サイズの低減)
 - WDS*c*上にWebFacing Toolを統合
 - ・共通操作による開発が可能に
 - WebFacing Toolで生成したアプリケーションは対話型ジョブとして実行されます。(iSeriesの対話型CPUカードが必要)

WDS*c* iSeries専用拡張機能

■ その他のiSeries用拡張機能

▶ ワークベンチからのiSeries固有のアクセス

- OS/400システムコマンドシェル
- AS/400 Toolbox for Java拡張機能
- ワークベンチツールにiSeries専用の通信機能

▶ Javaツールに対する機能拡張

- IFSとのエクスポート/インポート
- Javaクラス リモートコンパイル機能
- Javaアプリケーション リモート実行機能
- Javaアプリケーション リモートデバッグ機能
- DFU(データファイル更新)、PDM(オブジェクトリスト)、Swingフォーマット設定

▶ Webツールに対する機能拡張

- ワークベンチからのCODE設計機能の起動
- VisualAge RPGの機能拡張

▶ リモート・システム・エクスプローラー

- 複数iSeriesシステムへの接続作成 管理
- ネイティブ・オブジェクトの管理
- 処理の高速化
- リモート・コマンドの実行機能
- 5250セッションのオープン
- iSeries上のジョブの管理
- iSeries上のソースファイルメンバーの編集

■ その他のiSeries用拡張機能

- ワークベンチからのiSeries固有のアクセス
 - OS/400システムコマンドシェル
 - AS/400 Toolbox for Java拡張機能
 - ワークベンチツールにiSeries専用の通信機能
- Javaツールに対する機能拡張
 - IFSとのエクスポート/インポート
 - Javaクラス リモートコンパイル機能
 - Javaアプリケーション リモート実行機能
 - Javaアプリケーション リモートデバッグ機能
 - DFU(データファイル更新)、PDM(オブジェクトリスト)、Swingフォーマット設定
- Webツールに対する機能拡張
 - ワークベンチからのCODE設計機能の起動
 - VisualAge RPGの機能拡張
- リモート・システム・エクスプローラー
 - 複数iSeriesシステムへの接続作成 管理
 - ネイティブ・オブジェクトの管理
フィルターを作成してライブラリー、オブジェクト、メンバーの処理が可能
 - 処理の高速化
ポップアップメニューによる処理を高速化
 - リモート・コマンドの実行機能
コマンドセットの作成、定義済みフィルターによるリモートコマンドの実行等をサポート
 - 5250セッションのオープン
コマンドサブシステムから、関連する接続から5250セッションをオープン
 - iSeries上のジョブの管理
ジョブサブシステムから、iSeries上の特定ジョブグループの編成、モニターが可能。ジョブをツリー構造で表示。
 - iSeries上のソースファイルメンバーの編集

WDS*c*の前提条件 開発環境

■ iSeries AS/400

▶ ハードウェア

- iSeriesおよびAS/400 RISCモデル
- 270 #2250以上のCPU搭載モデルを推奨
- メモリ512MB以上
- ディスク4GB以上を推奨

▶ ソフトウェア

- OS/400 V5R1以降
- WASアドバンスド版 V4.0、アドバンスド・シングル・サーバー版 V4.0、WASアドバンスド版 V3.5、WASスタンダード版 V3.5
- 最新の累積PTF、WASグループPTFの適用を推奨

WDS*c*の前提条件 開発環境

■ ワークステーション

▶ ハードウェア

- Intel Pentium 600MHz以上を推奨
- メモリ最小256MB、512MB以上を推奨
 - ◆ WDS*c* V5ではさらに+256MBを推奨
- ディスク最小1.4GB、1.6GB以上を推奨

▶ ソフトウェア

- Windows 98, ME, XP + IE 5.5 SP1以降
- Windows NT SP6a以降 + IE 5.5 SP1以降
- Windows 2000 SP1以降 + IE 5.5 SP1以降
- Netscape 4.6以降も使用可能 (WebFacing Tool除 ◁)

WDSiCの前提条件 実行環境

■ iSeries AS/400

▶ ハードウェア

- iSeriesおよびAS/400 RISCモデル
- 270 #2252以上のCPU搭載モデルを推奨
- メモリ1GB ~ 2GB以上を推奨
- ディスク8GB ~ 16GB以上を推奨

▶ ソフトウェア

- OS/400 V4R4以降(WebFacing ToolはOS/400 V4R5以降)
- WASアドバンスド版 V4.0、アドバンスド・シングル・サーバー版 V4.0、WASアドバンスド版 V3.5、WASスタンダード版 V3.5
- 最新の累積PTF、WASグループPTF適用を推奨

WDSiCの前提条件 実行環境

■ クライアント

- ▶ IE 5.5 SP1以降を推奨
- ▶ Netscape 4.6以降を推奨 (WebFacing Toolを除く)
 - WebFacing Toolで生成したWebアプリケーションはNetscapeに未対応

WASを使用するためのPTF

▶ 以下の2種類のPTFを適用

– 累積PTFパッケージ

– WASグループPTF

- ◆ WASグループPTFの他にJAVA, HTTP, DB用のグループPTFも含む

▶ WASグループPTF

WASバージョン	OS/400 V4R4	OS/400 V4R5	OS400 V5R1
WAS V4.0 アドバンスド版	N/A	5733WA4-SF99239	5733WA4-SF99241
WAS V4.0 アドバンスド・シングル・サーバー版	N/A	5733WS4-SF99240	5733WS4-SF99242
WAS アドバンスド版 V3.5	5733WA3-SF99137	5733WA3-SF99138	5733WA3-SF99147
WAS スタンダート版 V3.5	5733AS3-SF99141	5733AS3-SF99142	5733AS3-SF99146

▶ iSeries上のWASと同じPTFレベルをWAS管理コンソールにも適用

- DSPDTAARAコマンドでWAS PTFレベルを確認

- iSeries上でWASを使用するには以下のPTFを適用します。
 - 累積PTFパッケージ
 - 最新の累積PTFパッケージを適用します。
 - WASグループPTF
 - 最新のWASグループPTFを適用します。
 - 最新のiSeries累積PTF、グループPTF情報は下記のURLより参照できます。
 - <http://www.iseries.ibm.com/websphere>
 - <http://www-6.ibm.com/jp/servers/eserver/series/techinfo/index.html>
 - WAS管理コンソールにもWASと同じレベルのPTFを適用
 - PC等に導入したWAS管理コンソールにもiSeries上のWASと同じレベルのPTFを適用する必要があります。適用しない場合、管理コンソールからWASに接続できない等、不具合が発生する場合があります。(下記のDSPDTAARAコマンドでiSeries上のWASバージョンを確認して同じバージョンになるようWAS管理コンソールにもPTFを適用します。)
 - Windows等の他プラットフォームのWAS PTFは以下のサイトからダウンロードします。
 - <http://www.ibm.com/software/webservers/appserv/support.html>
 - iSeries上のWASバージョン (PTFレベル)の確認方法
 - DSPDTAARAコマンドでWASの詳細なバージョンを確認できます。
 - DSPDTAARA (WASを導入したライブラリー)/(WASグループPTF番号)
 - (例) WAS V4.0 , OS/400 V5R1の場合
DSPDTAARA QEJBADV4/SF99241
- 上記コマンドを実行すると、以下のような結果が表示されます。以下の例では4.0.3がWASの詳細バージョンになります。この場合、WAS管理コンソールもWAS V4.0.3になるようPTFを適用します。
- Latest version (4.0.3):
Group PTF# SF99241-03 5733-WA4 V5R1 05/20/02 4.0.3

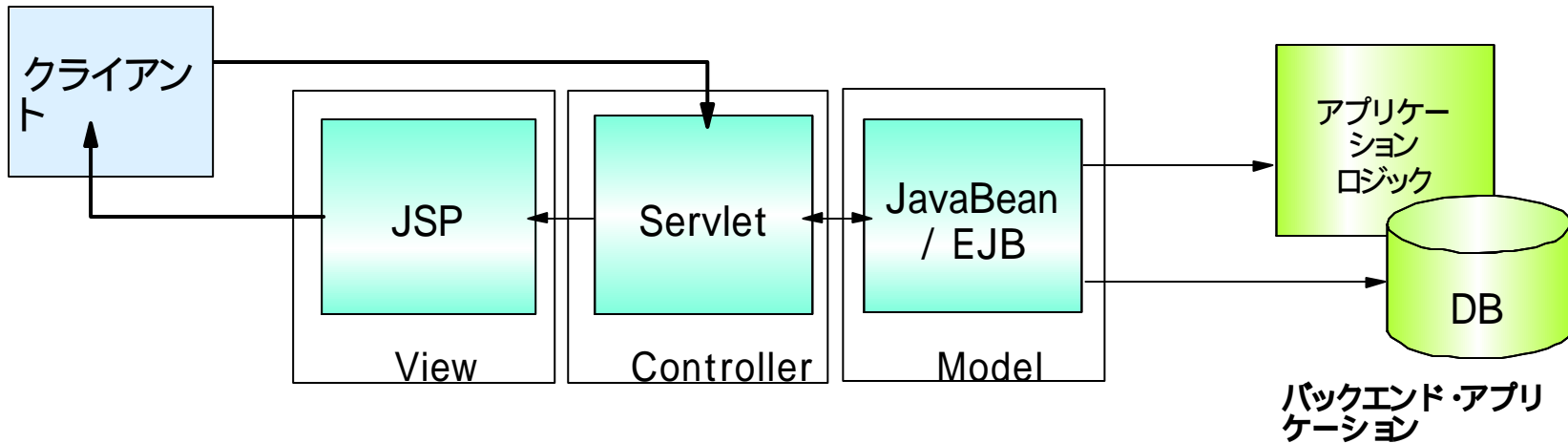
第2章.Webアプリケーション開発概説



プログラミングモデル

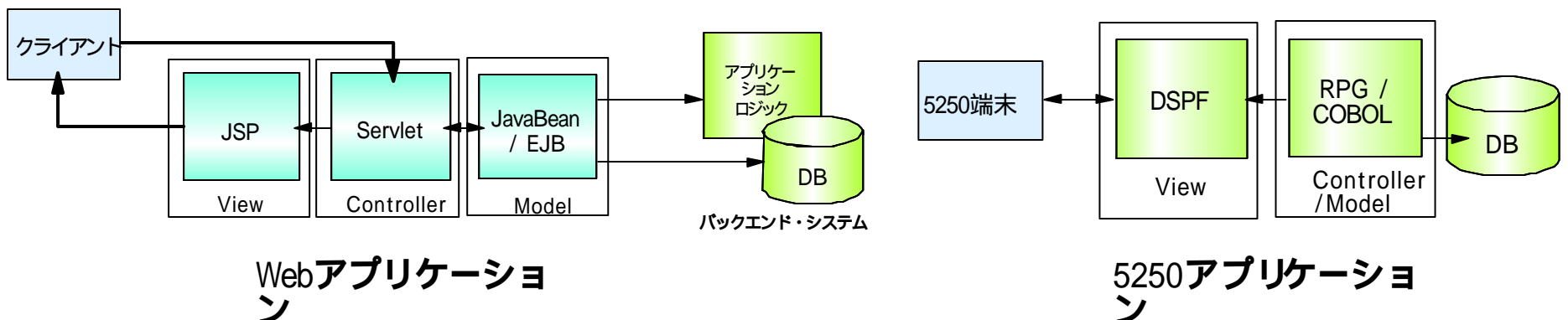
■ MVCモデル

- ▶ Webシステムを構築する際に使用するプログラミングモデル
- ▶ 処理を役割別に独立したコンポーネント(Model, View, Controller)に分割
 - Model (モデル) ビジネスロジックを記述。Java Bean, EJBで実装
 - View(ビュー) 画面生成。JSPで実装
 - Controller(コントローラー) Model, Viewの双方を制御、Servletで実装
- ▶ コンポーネントの独立性を高め、開発/運用保守性を向上させる事が目的



■ MVCモデル

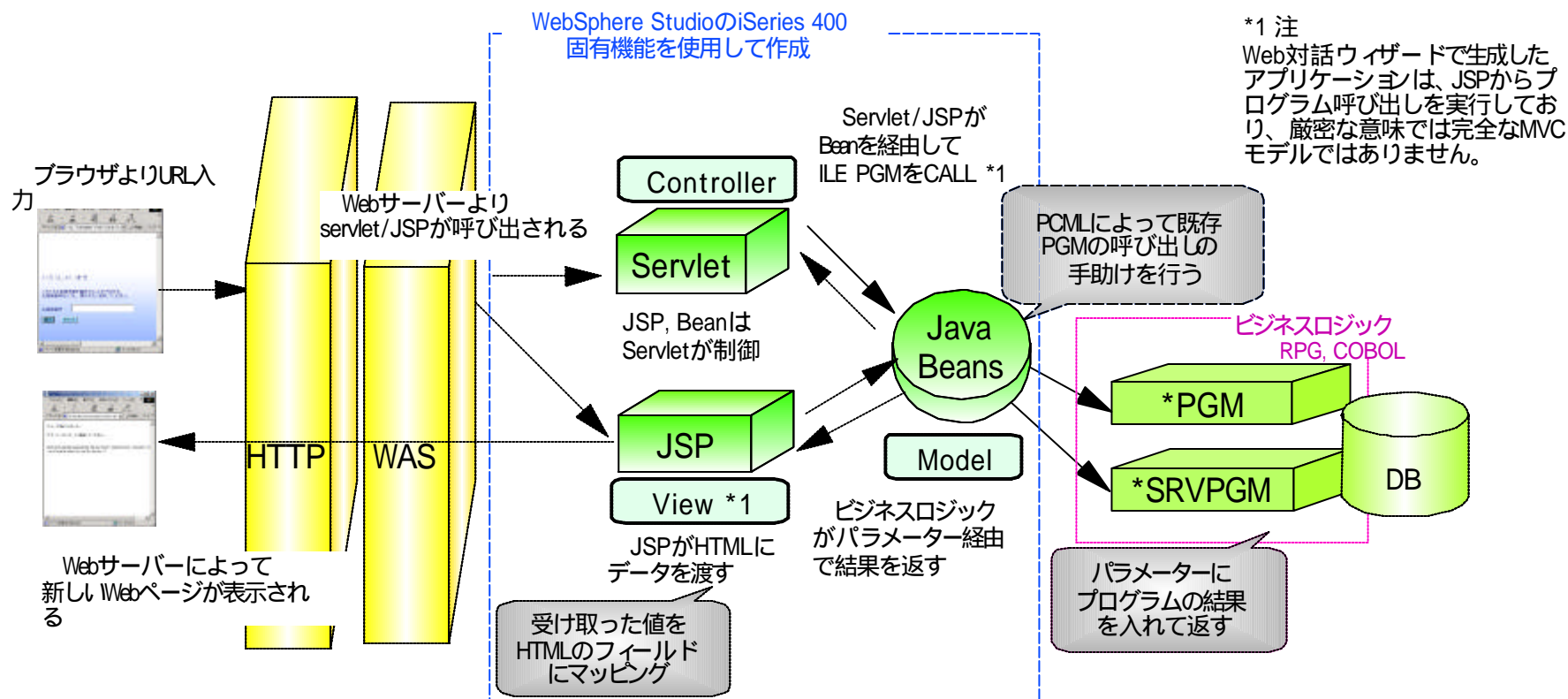
- MVCモデルとは、Webシステムを構築する際に使用される最も一般的なプログラミングモデルです。
- 役割別に独立したコンポーネント(Model, View, Controller)に分割して実装します。
 - Model (モデル) ビジネスロジックを記述。Java Bean, EJBで実装
 - View(ビュー) 画面生成。JSPで実装
 - Controller(コントローラー) Model, Viewの双方を制御、Servletで実装
- コンポーネントの独立性を高め、開発/運用保守性を向上させる事が目的です。
- Webクライアント (ブラウザ、アプレット等)からのHTTPリクエストはControllerであるサーブレットにより処理されます。サーブレットはModelを実装するJava Bean, EJB (Enterprise Java Beans)とViewを実装するJSPを制御します。
- Java BeanやEJBはバックエンドシステムのアプリケーションやデータベース等と通信を行いビジネスロジックを実行します。
- ビジネスロジックはJava Bean/EJBで実装する事もバックエンドのRPGやCOBOL等で実装する事も可能です。また、両方を混在させる事も可能です。
- JSP (Java Server Pages)はビジネスロジックにより得られた処理結果を元に動的にWebページを生成し、クライアントに送信します。
- MVCモデルに沿ってコンポーネントを分離すると、
 - ビジネスロジック部分はhttpなどWeb環境を意識せずに作成する事ができます。
 - JSPはビジネスロジックを知らない開発者でも画面制御のみを考えて作成できます。各コンポーネントの独立性が高まり、開発/運用保守の生産性向上が期待できます。
- 大まかな言い方ですが、JSP = 5250のDSPF, Servlet, JavaBean, EJB = RPG, COBOL と同じ役割と考えることも可能です。(サーブレットはRPG, COBOLのメインルーチン、Java Beanはサブ・ルーチン、あるいは汎用ロジックを実装した外部プログラムと見る事も可能です。)



(例) Web対話ウィザードで生成したWebアプリケーション

▶ 生成されるWebアプリケーションはMVCモデルとして生成(*1)

- Java Beansにはビジネスロジックなし。ロジックはバックエンドのRPG, COBOLに記述



The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

- Web対話ウィザードで生成したWebアプリケーションの例です。
 - WDSのWeb対話ウィザードで自動生成されたWebアプリケーションも基本的にはMVCモデルとなっています。
 - JSP, Java Beansをサーブレットが制御
 - JSPにはiSeries上のプログラムに渡すべき入出力フィールドが定義されています。
 - Java Beansは主にiSeries上のILE RPG, ILE COBOLをパラメータ呼び出しするための機能を提供しています。
 - 実際のビジネスロジックはILE RPG, ILE COBOL上に記述されます。
- *注 Web対話ウィザードで生成したJSP上からプログラム呼び出しを行っています。(JSP上にプログラム呼び出しのための記述がされています。) によって、厳密な意味では完全なMVCモデルではないと言えますが基本構造はMVCモデルに準拠しています。
- WAS V4iにはMVCモデルのサンプルが付属しています。
 - View : simple.jsp
 - Model : SimpleJSPBean.java
 - Controller : SimpleJSPServlet
 - 上記のWAS付属のサンプルプログラム、MVCモデルの解説については以下の技術情報URLをご参照ください。
WebSphere Developer Domain

<http://www-6.ibm.com/jp/software/websphere/developer/w40/iw/index.html>

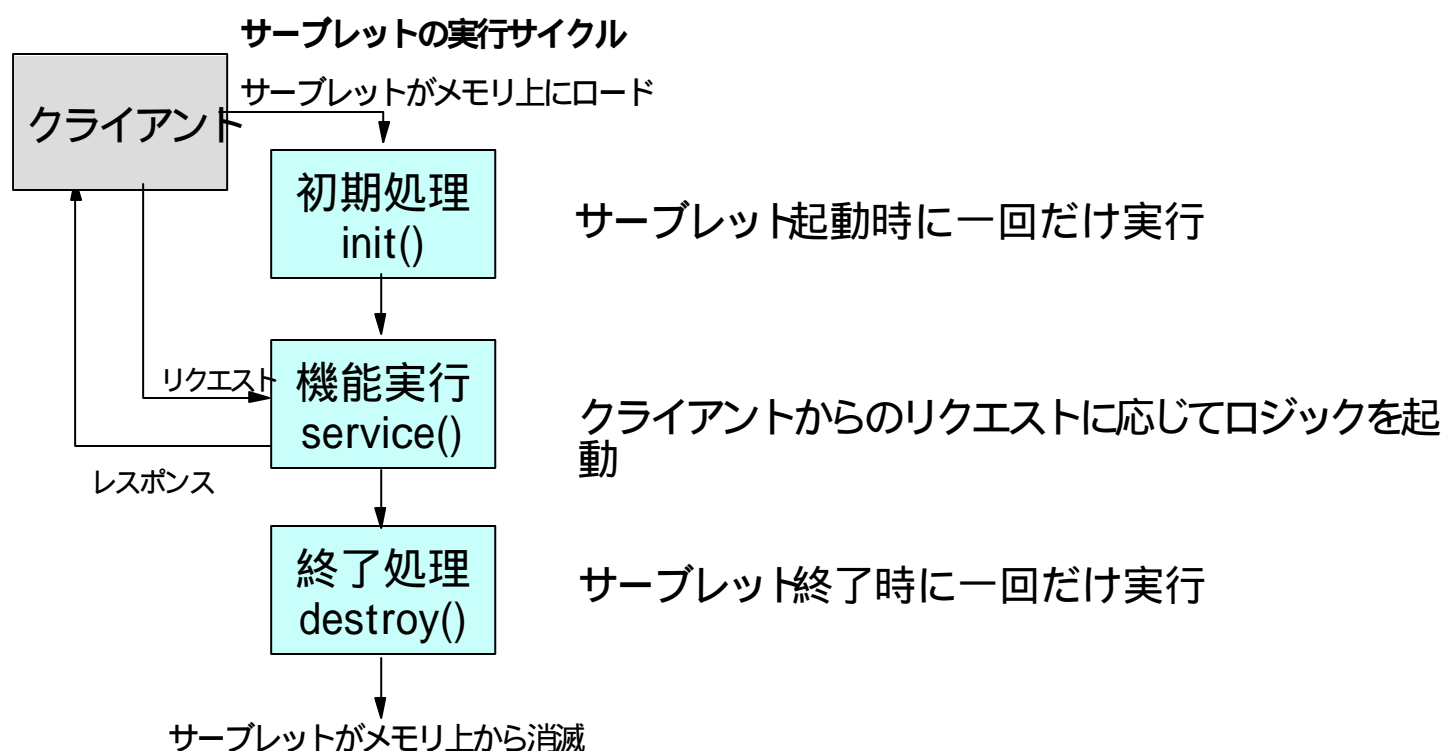
The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

Webアプリケーションのコンポーネント：サーブレット

▶ サーブレット

- クライアントと対話処理を行うサーバプログラム
- サーブレット・コンテナがリクエストに応じてservice()に記述された機能呼び出し



■ サーブレット

- サーブレットとはHTTPサーバーを経由してクライアントにサービスを行うJavaで記述されたサーバプログラムです。
- アプリケーションサーバー上 (JVM上)にロードされて実行します。
- サーブレットは最初にクライアントからのリクエストがあった時点でメモリ上にロードされます。その後はアプリケーションサーバーに常駐し、クライアントからのリクエストに応じて該当するサービスを実行します。
- init()メソッドはサーブレットがメモリにロードされた時に一度だけ実行されます。サーブレット実行に必要な初期化処理やサーブレットで使用するファイルを事前にメモリ上にロードしたりします。
- service()メソッドはクライアントからのリクエスト毎に毎回、呼び出され実行します。service()には、1.クライアントからのリクエスト内容を保持するオブジェクト、2.クライアントへのレスポンス内容を保持するオブジェクトの2つが渡されます。
- リクエスト・オブジェクトからパラメータを取得し、パラメータに応じたビジネスロジックを呼び出します。処理結果をレスポンス・オブジェクトに保管し結果をWebページ(JSP等)にセットして出力します。
- 出力された結果のWebページはHTTPサーバーを経由してクライアントに帰されます。
- destroy()メソッドはサーブレットがメモリ上から消滅する時に実行されます。データ保存やログ作成等を行います。

Webアプリケーションのコンポーネント : JSP

■ JSP(Java Server Pages)

- ▶ サーバサイドでのスクリプティング技術
- ▶ コンテンツを動的にインクルードしてWebページを生成
- ▶ HTML, XMLファイルにJSPタグを記述、Javaコードを実行

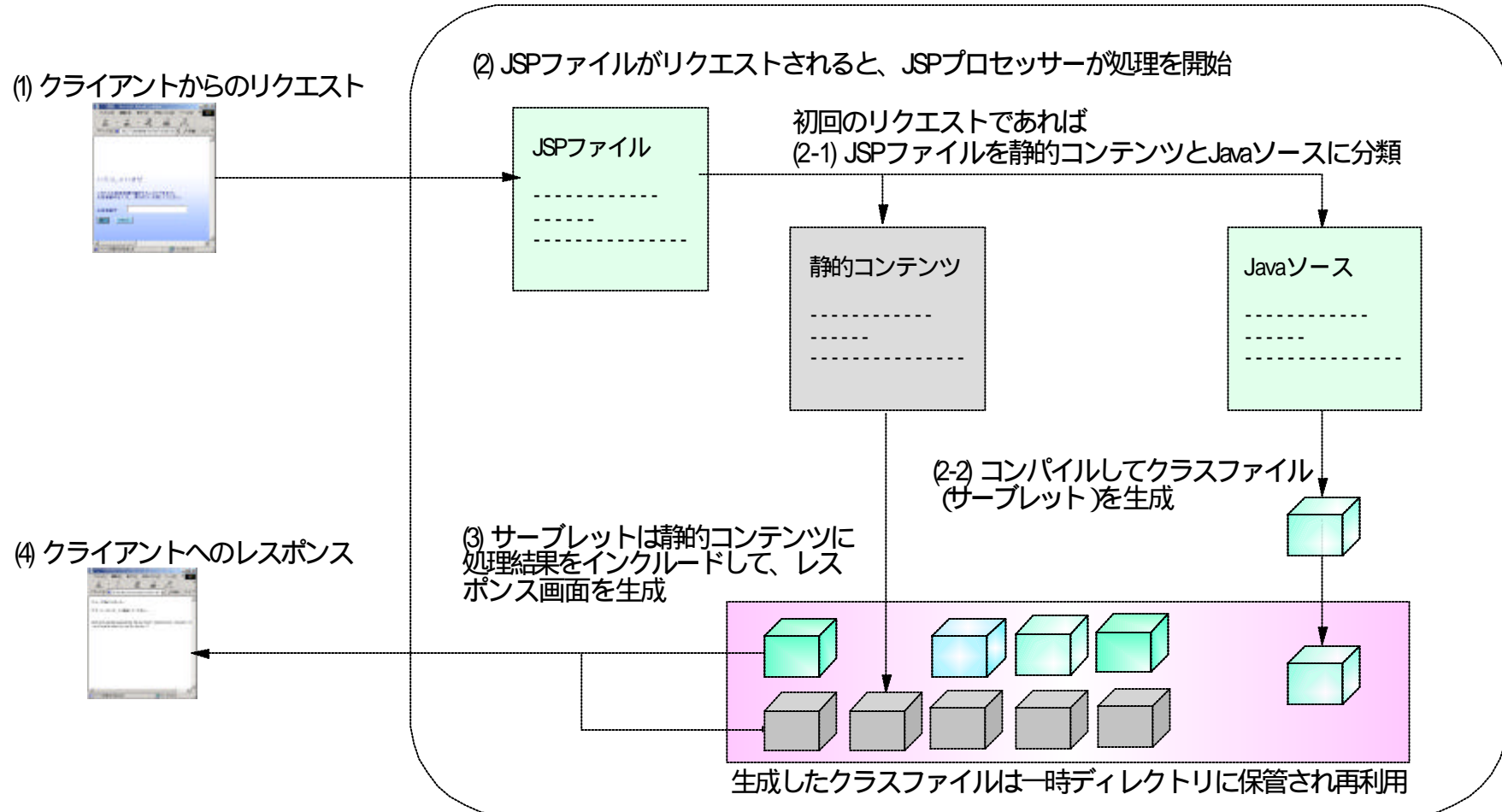
```

<HTML>
<HEAD><TITLE>Put Your Title Here</TITLE>
<%!
    String myName;           宣言
%>
<%
    myName = request.getParameter("myName");
%>
<body>                       スクリプレット
    <h1> Hello,
%>:
    <%=
        myName
    %>                         表示
</body>
</HTML>
    
```

JSPタグ	名称	用途
<%@ %>	ディレクティブ	生成されるサーブレットのプロフィール。パッケージのインポート等の指示
<%! %>	宣言	生成されるサーブレットのクラス変数、メソッドの宣言
<% %>	スクリプレット	処理の実行
<%= %>	表示	HTML, XMLに変数式の評価結果をインクルード
<JSP:xxx />	標準アクション	使用頻度の高い処理をタグ化。Java Beanの処理等

- JSP (Java Server Pages)はSUNが開発したサーバ・サイドスクリプティング技術です。
- 動的なコンテンツはJavaのコードによって生成されます。HTML, XMLファイルにJSPタグを使ってJavaのコードを埋め込み静的なコンテンツに動的コンテンツをインクルードします。
- JSPタグには以下の5つがあります。
 - <%@ %> ディレクティブ
JSPは実行時にサーブレットを生成します。ディレクティブでは生成されるサーブレットのプロフィールを指示します。例えばサーブレットにパッケージのインポートやクラス継承などを指示します。暗黙的にjavax.servletクラスとjavax.servlet.httpクラスをインポートし、HttpServletクラスを継承します。
 - <%! %> 宣言
ページで参照される編集、メソッドを宣言します。これらは生成されるサーブレットのクラス編集、メソッドとなります。
 - <% %> スクリプレット
クライアントのリクエストを処理するコードを記述します。クライアントとの対話処理で使用するHttpServletRequestクラス、HttpServletResponseクラスなどは宣言なしにそれぞれrequest, responseで参照可能です。
 - <%= %> 表示 (Expression)
Javaの変数式を記述します。編集式の評価結果が文字列に変換されてHTML, XMLファイルの該当箇所にインクルードされます。
 - <jsp:xxx /> 標準アクション
xxxで指定された処理を実行します。通常、処理はスクリプレットで記述しますが、使用頻度の高い処理は簡単に利用できるようにタグが提供されています。例えばJava Beanをアクセスするタグなどが提供されています。
- 前ページのJSPサンプルはJSPにクライアントのリクエスト処理を記述しています。MVCモデルで作成した場合、これらの処理は全てサーブレット上に記述し、JSP上にはリクエストの処理記述は記述しないようにする必要があります。

(参考) JSP動作フロー



The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

- JSPの動作フローについて
 - (1), (2) クライアントからJSPファイルが要求された場合、JSPプロセッサが処理を開始します。
 - (2-1) JSPプロセッサはJSPファイルを2つのファイルに分離します。
 - 一つは静的なコンテンツのみのファイル
 - もう一つはJavaソースのファイル (動的に変更される部分)
 - (2-2) JSPプロセッサはJavaソースファイルをコンパイルしてクラスファイルを作成します。このクラスファイルはアプリケーションサーバー上でサーブレットとして動作します。
 - 生成されたクラスファイルはJSP用のワークディレクトリに保管されます。(2度目以降のリクエストで再利用される)
 - (3) (2-2)で生成したサーブレットは静的コンテンツのファイルを読み込み処理結果をインクルードしながらレスポンス画面を生成します。
- JSPプロセッサは生成したクラスファイルを一時ディレクトリに保管します。2度目以降のリクエストの場合、このクラスファイルを再利用して処理が行われます。つまり、(2-1), (2-2)のステップがスキップされるため処理が高速化されます。
- 逆を言えばJSPの処理は初回のみサーブレット生成のためのコンパイルが発生し、遅くなるとも言えます。

The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

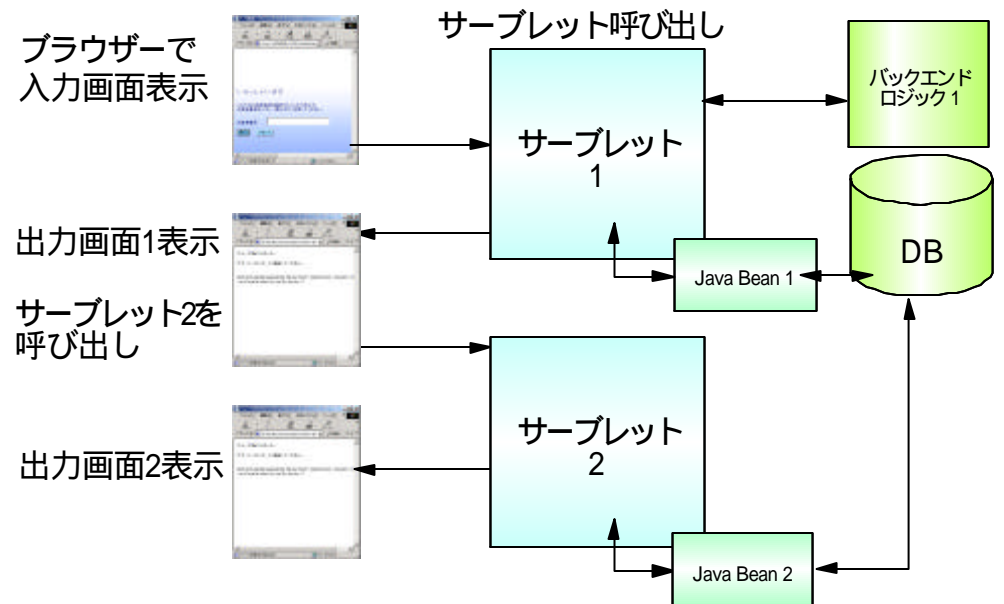
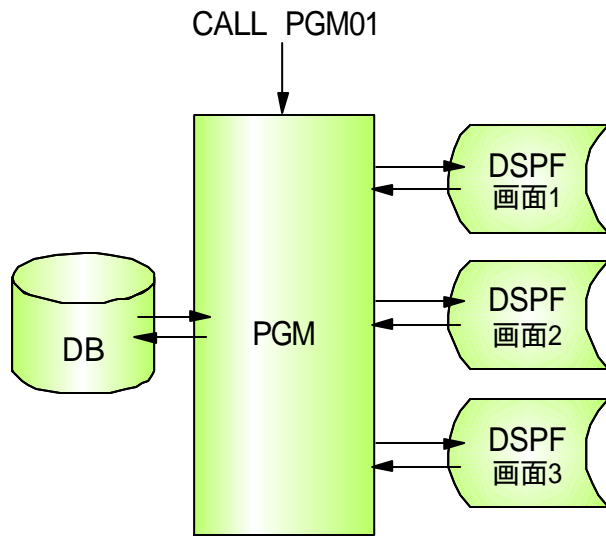
5250対話型プログラムとWebアプリケーションの違い

■ 5250対話型プログラム

- ▶ PGMを起動 画面の制御
- ▶ 3画面以上を一つのPGMで処理

■ Webアプリケーション

- ▶ 画面よりリクエスト PGMを起動
- ▶ 入力・出力の2画面を一つのPGMで処理
- ▶ 5250 パラメーター渡しPGM CALLと類似

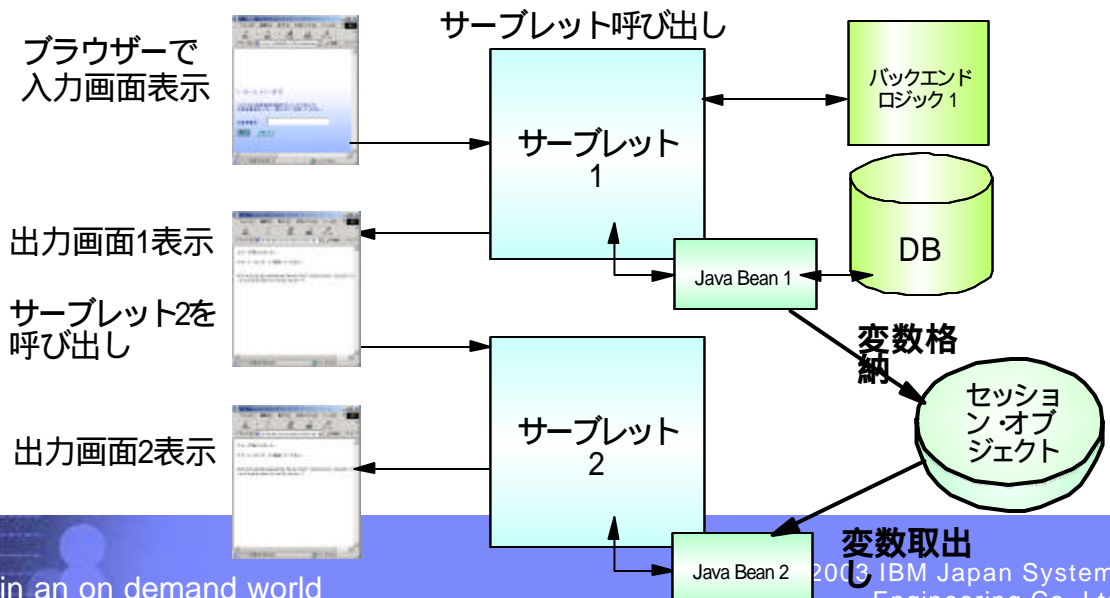


The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

- 5250対話型プログラムとWebアプリケーションの実行形態の違いについて説明します。
- 5250対話型プログラム
 - プログラムの起動方法 CALL PGM01 のようにプログラムの実行を指定する事によりプログラムが開始されます。プログラム内部に使用する5250画面ファイルの定義が保管されています。
 - 3画面以上を一つのプログラムで処理 (例: 受注入力プログラムなど) 3画面以上にわたる画面遷移が必要な場合でも一本のRPG、COBOLなどで制御を行う場合が普通です。
- Webアプリケーション
 - プログラムの起動方法 あるJSP画面上で送信ボタンなどを押したタイミングで、指定されたサーブレットが呼び出されます。呼び出されたサーブレットはJSP画面からの入力値等をパラメーターで受け取り、後の処理に使用します。
 - 入力・出力の2画面を一つのPGMで処理 5250対話型プログラムと異なり入力画面 プログラム処理 出力画面で処理が完了します。このため扱える画面も基本は2画面となります。
 - 3画面以上にわたる場合はプログラムを複数つなげて処理を続けます。その際、複数のプログラム間で変数の値を共有するための仕組みがWAS上に用意されています。(セッションオブジェクト)
 - (例) WdScのWeb対話ウィザードでは3画面以上にわたる画面遷移が必要な場合のためにセッションオブジェクトを使用した、プログラム間の変数値の共有を簡単に作成できます。
 - 形態として、Webアプリケーション用のプログラムは「5250アプリケーションのパラメーター渡しPGM CALLと類似している」と言えます。

複数サーブレット間での変数共有



The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

Webアプリケーション開発手順の概要

■ 開発環境

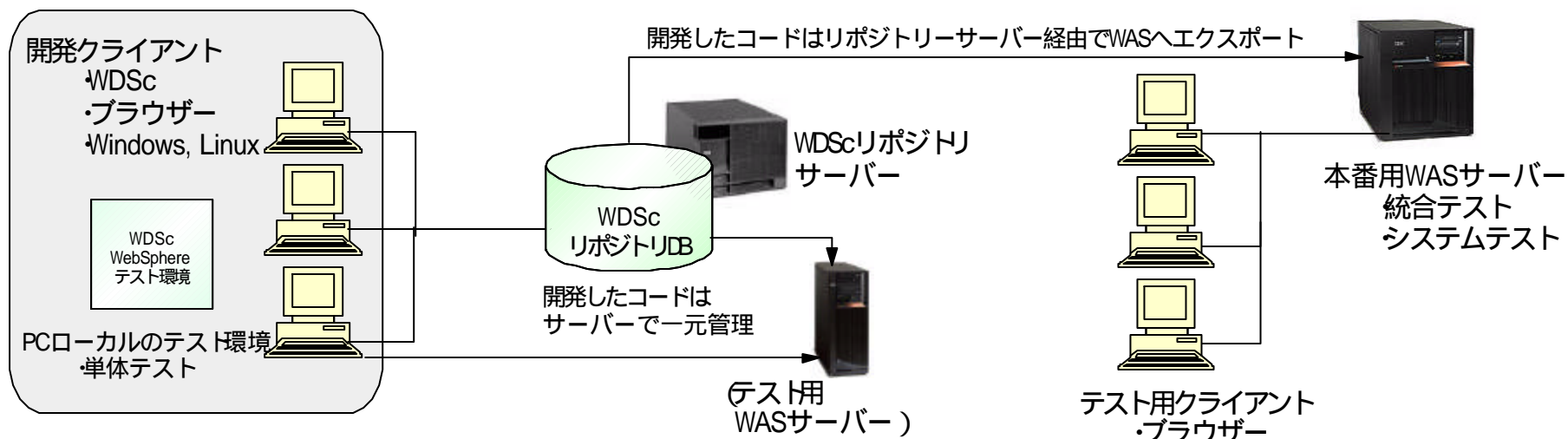
- ▶ 本番用サーバー(WAS)
- ▶ テスト用サーバー(WAS)
- ▶ WDSclリポジトリサーバー
- ▶ 開発用Windows, Linuxクライアント

■ コーディング・単体テスト

- ▶ PC + WDScl上のWASテスト環境

■ 統合テスト・システムテスト

- ▶ 本番サーバー上のWASでテスト
- ▶ テスト用WASサーバー設置も検討



The next generation iSeries...simplicity in an on demand world

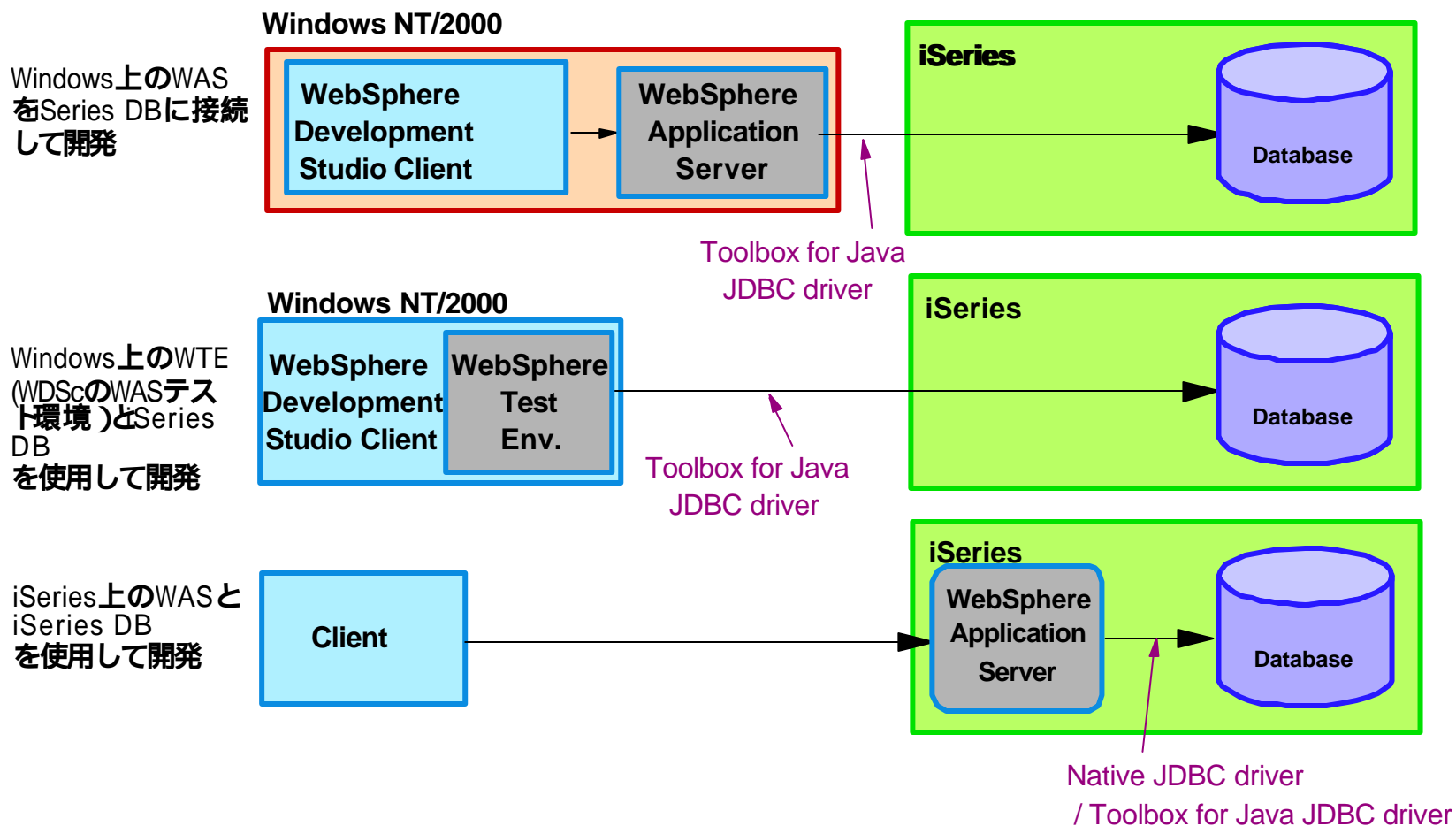
© 2003 IBM Japan Systems Engineering Co.,Ltd.

- Webアプリケーション開発手順の概要について
- 開発環境 : 以下のサーバー 開発用クライアントを準備します。
 - 本番用サーバー(WAS)
 - 本番時にWASの稼働するiSeries
 - WDSclリポジトリサーバー
 - 各クライアントで開発したコードを一元管理するためにリポジトリサーバーを設置します。リポジトリサーバー上にあるコードが必ず正となるような運用ルール決定も必要です。
 - 開発用クライアント
 - WDSclはWindowsクライアントの他にLinuxクライアントもサポートしています。
 - テスト用WASサーバー
 - 必須ではありませんが、プロジェクト状況から必要と判断した場合はテスト用サーバーを設置します。
- 開発方法 : コーディング・単体テスト
 - コーディングは各自のPC上でWDSclを使用して実施します。
 - 単体テストはWDScl上に用意されたWebSphereテスト環境を利用します。これはWASの動作環境をPC上でエミュレートする機能です。この機能を使用するとテストの度にWASサーバーへプログラムをエクスポートする手間を省くことができます。
- 開発方法 : 統合テスト・システムテスト
 - 本番用、あるいはテスト用のWASサーバー機にWebアプリケーション・プログラムをエクスポートしてテストを実施します。
 - コードの管理はここでも重要です。必ずリポジトリサーバーを経由するなどの運用ルールを決定する必要があります。

The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

アプリケーション開発形態とJDBCドライバー



The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

- iSeriesでは2種類のJDBCドライバーを提供
 - Toolbox for Java JDBCドライバー
 - OS/400の無料オプション、Toolbox for Javaに付属するJDBCドライバー (jt400.jar)
 - クラス名 : com.ibm.as4000.access.AS400JDBCdriver URL接頭部 : jdbc:// jdbc:as400
 - WASを導入しただけでは自動導入されないため、jt400.jarファイルを入手してWASにインストールする必要があります。以下のURLやiSeries IFS上からjt400.jarを入手できます。
 JTOpen (Toolbox for Java JDBCドライバーのオープンソース版) <http://www-124.ibm.com/developerworks/oss/jt400/>
 iSeries IFS (Toolbox for Java JDBCドライバー) /QIBM/ProdData/HTTP/Public/jt400/lib/jt400.jar
 - * 上記 2種類のドライバーは基本的な機能レベルは同等です。
 - 以下のネイティブJDBCドライバーと異なりWASとリモートのiSeries上のデータベース間でのJDBCアクセスが可能。
 - iSeries以外のWASとiSeries DB(DB2/400)間のJDBCアクセスでも使用可能。
 - ネイティブJDBCドライバー
 - WASの動作するiSeries上の(ローカル)DB2/400に対してJDBCアクセスする場合に使用可能です。
 - クラス名 : com.ibm.db2.jdbc.app.DB2Driver URL接頭部 : jdbc:// jdbc:db2
 - WASの動作するサーバー以外のリモートのデータベースにJDBC接続する場合には使用できません。
 - マイクロコードレベルの実装となるためToolbox for Java JDBCドライバーに比較して処理が高速になる場合があります。
 - iSeries上にWASを導入するとネイティブJDBCドライバーが自動的にインストールされます。
- 開発時のテスト環境について
 - Windows上のWASとSeriesDB使用して開発・テスト
 - Toolbox for Java JDBCドライバーをWindows上のWASにインストールして使用します。
 - Windows上の開発環境(WDSc, WSAD, VisualAge for Java等)とSeries DBを使用して開発・テスト
 - WDSc等の開発環境にToolbox for Java JDBCドライバーをインストールして使用します。
 - 単一のiSeries上のWASとDBを使用して開発・テスト
 - 一般にはiSeries上のWASにネイティブJDBCドライバーをインストールして使用します。通常、ネイティブJDBCドライバーの方がパフォーマンス上優位です。
 - 特定のケースによってはToolbox for Java JDBCドライバーをWASにインストールして使用します。

The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

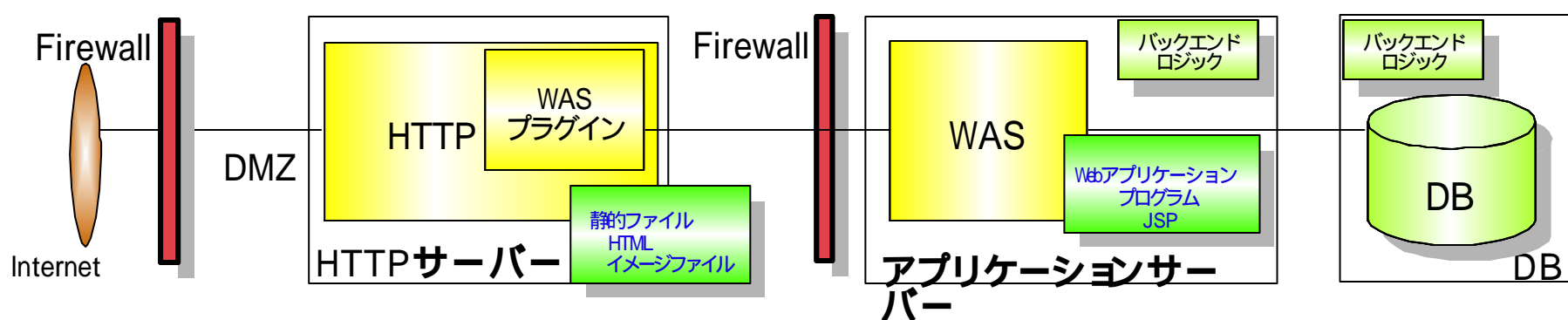
iSeries用のJDBCドライバー一覧

ドライバー Type	ドライバー名	インプリメンテーションクラスファイル名 /	データベース URL	接続形態 ローカルDB	モートDB	フェーズ コミット2
Type 2	DB2 UDB for iSeries ネイティブドライバー (V5R2以降)	com.ibm.db2.jdbc.app.UDBConnectionPoolDataSource /QIBM/UserData/Java400/ext/db2_class.jar	jdbc:db2://		(DRDA接続)	×
	DB2 UDB for iSeries ネイティブXAドライバー (V5R2以降)	com.ibm.db2.jdbc.app.UDBXDataSource /QIBM/UserData/Java400/ext/db2_class.jar	jdbc:db2://		(DRDA接続)	
	DB2 UDB for iSeries ネイティブドライバー (V5R1以前)	com.ibm.db2.jdbc.app.DB2StdConnectionPoolDataSource /QIBM/UserData/Java400/ext/db2_class.jar	jdbc:db2://		(DRDA接続)	×
	DB2 UDB for iSeries ネイティブXAドライバー (V5R1以前)	com.ibm.db2.jdbc.app.DB2StdXADataSource /QIBM/UserData/Java400/ext/db2_class.jar	jdbc:db2://		(DRDA接続)	
Type 4	Toolbox for Java (Toolbox)	com.ibm.as400.access.AS400JDBCConnectionPoolDataSource /QIBM/ProdData/http/Public/jt400jar/lib/jt400.jar	jdbc:as400://			×
	Toolbox for Java (Toolbox XA)	com.ibm.as400.access.AS400JDBCXADataSource /QIBM/ProdData/http/Public/jt400jar/lib/jt400.jar	jdbc:as400://			
	JTOpen	com.ibm.as400.access.AS400JDBCConnectionPoolDataSource (JTOpenのサイトよりダウンロード)	jdbc:as400://			×
	JTOpen XA	com.ibm.as400.access.AS400JDBCConnectionPoolDataSource (JTOpenのサイトよりダウンロード)	jdbc:as400://			



サーバーの3層構造

- Webアプリケーション実行に必要なサーバー機能
 - HTTPサーバー
 - アプリケーションサーバー (WAS)
 - DBサーバー
- サーバー機能の配置
 - 単一サーバー(H/W)上で全てのサーバー機能を稼動
 - 複数サーバー上(H/W)に分割して配置
 - トランザクション数、可用性 (耐障害性)、セキュリティ、等を考慮して配置



The next generation iSeries...simplicity in an on demand world

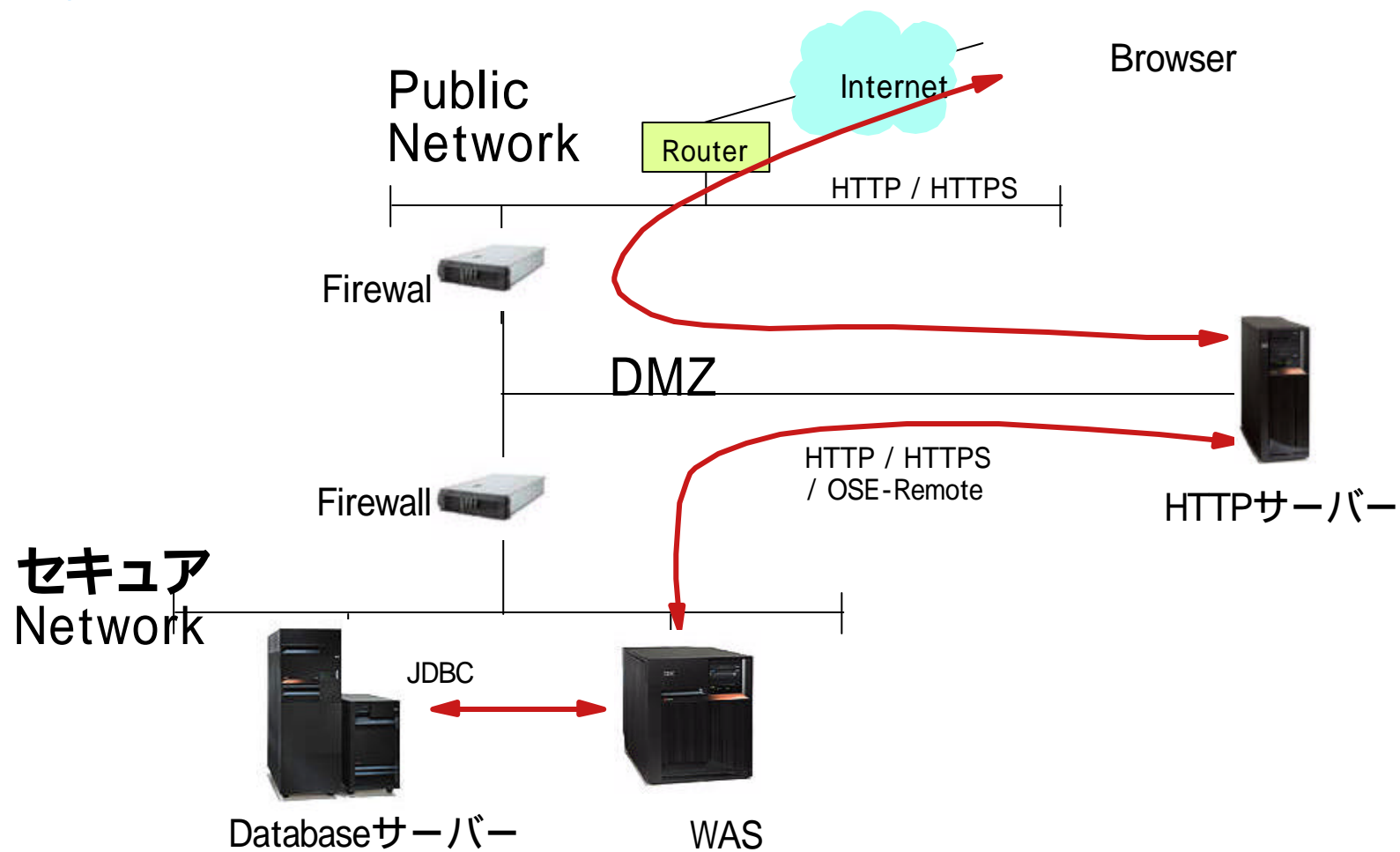
© 2003 IBM Japan Systems Engineering Co.,Ltd.

- Webアプリケーション実行に必要なサーバー機能は以下の3つがあります。
 - HTTPサーバー
 - ・クライアントからのHTTPリクエストを処理
 - 静的ファイル (HTML, イメージファイル (JPEG, GIF等)) を表示する機能(WAS上に配置する場合もあり)
 - アプリケーションサーバー
 - ・アプリケーションサーバーとはJVM上で実行されるサーブレット実行環境
 - WAS : WebSphere Application Server
 - HTTPサーバーからのリクエストでサーブレット、Java Beans, EJB等の実行、セッション管理、DB接続などを実行
 - ・サーブレット、Java Beans, EJBなどのWebアプリケーションロジックを配置
 - WASアプリケーション以外のバックエンド・ロジック (RPG, COBOLなどのビジネスロジック) を配置する場合もあり
 - DBサーバー
 - ・アプリケーションサーバーからのデータベース処理リクエストを実行。
 - バックエンド・ロジック (RPG, COBOLなどのビジネスロジック) を配置する場合もあり
- サーバー機能の配置
 - 単一サーバー上で全てのサーバー機能を稼動
 - 比較的小規模な構成ではHTTP, WAS, DBを単一サーバー上で動作させます。
 - 複数サーバー上に分割して配置
 - 通常、要件に合わせてHTTP, WAS, DBを別サーバーに分割して配置します。
 - 特にインターネット接続を前提とする場合は、HTTPサーバーを分離してDMZ上に配置します。セキュリティの観点からは必須です。
 - WAS, DBサーバーはDMZに配置しない構成がセキュリティ的には望ましい、といえます。
 - ・負荷分散、可用性向上などを考慮する場合、HTTPサーバー、WASサーバーを複数台、並列に配置(二重化)する構成も一般的です。

The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

(例)サーバー3層構造配置



The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

- HTTP・アプリケーションロジック (WAS)・データベースの完全な分離モデルの構成例です。
- Public Network
 - Firewallによって保護されず、Client Requestをまず最初に受けるセグメント
- DMZ(Demilitarized Zone)
 - DMZ(非武装地帯)と呼ばれる、インターネットからのアクセスを許可するセグメント。基本的には外部からの侵入、破壊の可能性を前提におくべきセグメント。
- HTTPサーバー
 - アプリケーションロジックは一切配置せず、HTTPリクエストの中継のみをする。HTMLファイル、イメージファイルなど静的情報を格納する場合もあるが、セキュリティ的にはこれらの静的ファイルもWAS側に配置する事が望ましい。想定ユーザー数が多い場合などは負荷分散、障害対応等を目的に複数台のHTTPサーバーを並列に配置。(この場合ネットワークディスパッチャーがインターネットからのリクエストを中継し、HTTPサーバーに転送)
 - HTTPサーバーにはWASに対してリクエストを転送するためのプラグインを導入する
 - DMZ内のサーバーは論理的に1台のFirewallにより保護される
- セキュアNetwork
 - インターネットからのアクセスを許可しないセキュアなセグメント。社内ネットワーク。
 - セキュアNetworkは論理的に2台のFirewallにより保護される。
 - WAS, DBサーバーを配置する。
- WAS
 - アプリケーションロジックを実行。想定ユーザー数が多い大規模システムでは負荷分散、障害対応、可容性向上のため複数のアプリケーション実行環境を持つ。
- Databaseサーバー
 - 実際に処理するデータベースを格納。通常JDBCインターフェースを介してWASからのリクエストを処理。

The next generation iSeries...simplicity in an on demand world

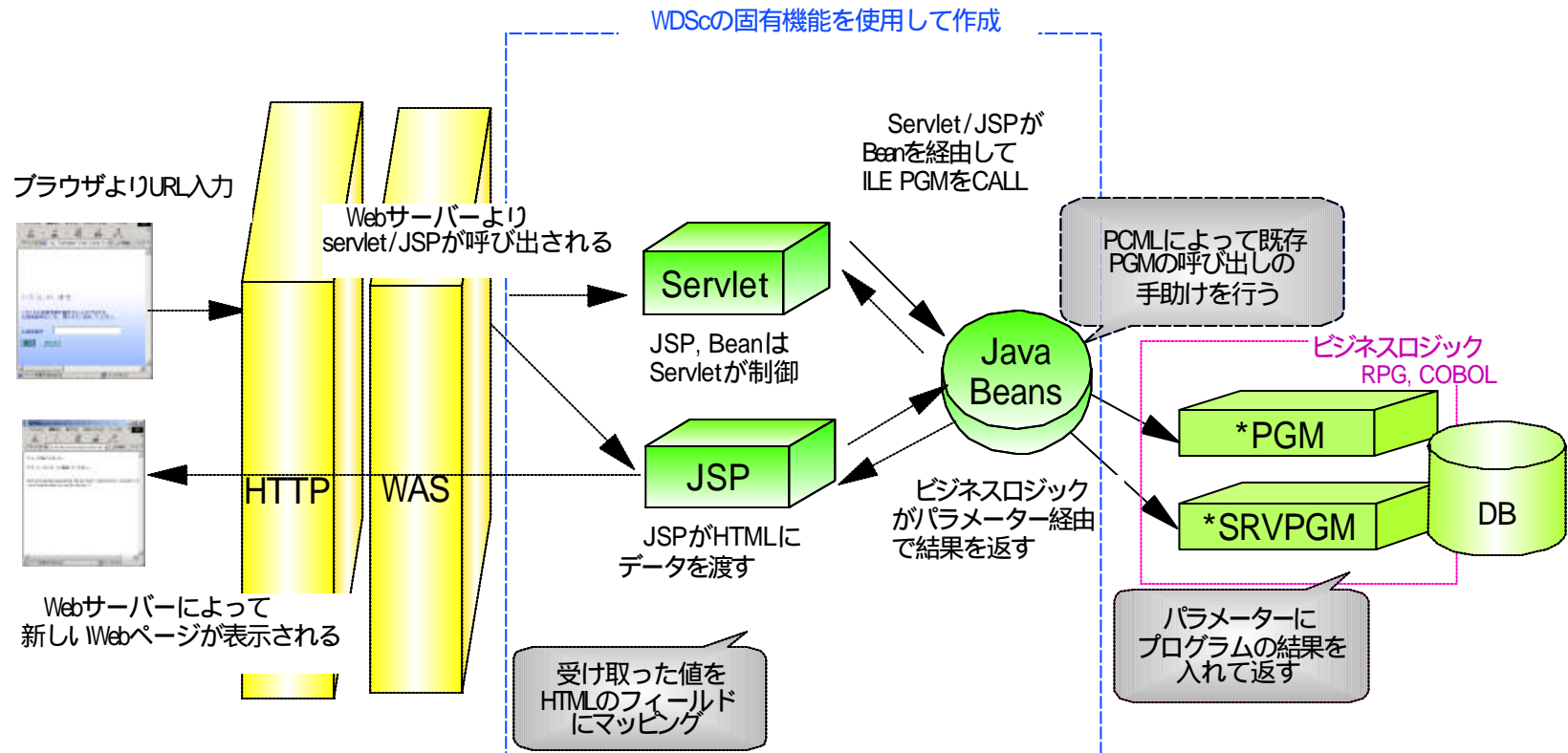
© 2003 IBM Japan Systems Engineering Co.,Ltd.

第3章.WDScでの開発

iSeriesプログラムを使用した開発

iSeries上のプログラム活用方法

■ iSeries上のプログラムはJavaBeanによって呼び出されます



The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

■ iSeries上プログラムの活用方法

- WDSのWeb対話ウィザードを使用することで、iSeries上にあるプログラムをWebアプリケーションを開発し、活用することができます。その実現方法は、ウィザードによって自動生成されるJavaBeanやPCMLを用いることで、iSeriesプログラムを呼び出すことができます。その手順としては以下のとおりです。
 - ブラウザよりURLが入力されます
 - Webサーバーより該当のservletもしくはJSPが呼び出されます
 - servlet/JSPはJavaBeanを経由してILEプログラムを呼び出します
 - (JavaBeanはPCMLを使用してプログラム呼び出しを行っています。)
 - ILEプログラムはパラメーター経由で結果を返します
 - JSPは受け取ったデータをHTMLに渡します
 - (受け取ったデータは、該当するWebページのフィールドにマッピングされます)
 - Webサーバーによって新しいHTMLが呼び出され結果が表示されます

The next generation iSeries...simplicity in an on demand world

© 2003 IBM Japan Systems Engineering Co.,Ltd.

iSeriesプログラムの活用条件

■ ILEプログラム

- RPG
- COBOL
- C++、etc...
 - *PGM
 - *SRVPGM

■ パラメーター受け渡し

- ▶ ILEプログラムとWebページはパラメーターを使用して対話
 - 入力パラメーター
 - 出力パラメーター

■ iSeriesプログラムの活用条件

- iSeriesプログラムを活用するには、いくつかの条件があります。
ウィザードで使用できるiSeriesプログラムはILEプログラムのみであるため、OPMプログラムである場合、コンバージョンを行う必要があります。RPG、COBOL、C++などで*PGMタイプでも*SRVPGMタイプであれば使用することができます。
対話ウィザードは、ILEプログラムとWebページ(jspファイル)をパラメーターを使用して対話させるため、プログラムはパラメーターの受け渡しができるものでなければなりません。Webページから受け取った値をILEプログラムが入力パラメーターとして受け取り、出力パラメーターの値をその結果としてWebページに返していき形をとります。
DSPFを使用して入力、出力を行っているプログラムの、入出力部分を全てWebページ(ブラウザから見られる画面)に置き換えて使用します。
- これらの条件を満たしたiSeriesプログラムを、WDS cの対話ウィザード機能を使用することで活用することができます。

開発の手順

1. 入出力用のWebページを作成
 - ▶ DSPFに値する入出力部分をJSPファイル置き換え
2. WebページとILEプログラムを結びつけ
 - ▶ Webページのフィールドと ILEプログラムのパラメーターをリンク
3. warファイルによるエクスポート
 - ▶ 作成したコンポーネントをwarファイルとしてパッケージ
4. Webアプリケーション稼動環境へ配置
 - ▶ Webアプリケーション・サーバー環境にwarファイルを配置

■ 開発の手順

1. 対話ウィザードを使用してWebアプリケーションを開発する場合、まずはじめにiSeriesプログラムと対話させるためのWeb画面を作成します。この画面は、本来DSPFに相当していた部分の置き換えとなります。このWeb画面はウィザードによって自動生成することもできますが、独自のWeb画面を作成し、使用することも可能です。
 2. Web画面を作成した後、Web画面とiSeriesプログラムの結びつけを行います。WebSphere Development Studio for iSeriesで提供されているウィザードで、以下の項目を指定します。
 - ・使用するiSeriesプログラム名
 - ・Webページ名
 - ・iSeriesプログラムで指定されているパラメーター
 - ・iSeriesプログラムで指定されているパラメーターとWeb画面上にあるフィールドのリンク
 この設定を基に、ILEプログラムを呼び出すJavaプログラム、Javaプログラムにより使用されるPCMLファイル等が自動生成されます。
 3. 最後にこれらの生成されたファイルをiSeriesサーバー上に配置するためにwarファイルの形式でエクスポートします。
 4. 生成されたwarファイルを実行のためにiSeriesサーバー上に発行します。発行とは、ワークベンチにおいて作られたファイルをiSeriesサーバーのディレクトリーに配置(デプロイ)することを言います。
- なお、今回は使用するiSeriesプログラムが対話ウィザード用に用意してあることを前提としていますが、iSeriesプログラムを作成する場合は、以下の内容に適しているかを確認します。
- 使用するiSeriesプログラムはILEで、且つ、パラメーター受け渡しプログラムでなければなりません。(これは、5250アプリケーションとWebページ(htmlやjspファイル)とがパラメーターを受け渡すことで連携を取るためです。)
- プログラムとDSPFとのデータの流れを把握した後に、必要に応じてプログラムの編集します。画面を使用するプログラムである場合、画面処理のコーディングは取り除き、画面との入出力データをパラメーターで受け渡しできるコーディングに変更します。必要なコーディングを済ませた後、OPMプログラムである場合にはそれをILEに変換します。(eg. CVTRPGSRC)