【セッションNo. 2】

Delphi/400開発ノウハウお教えします

「情報を守ろう!
安全性を高めたWebシステムの構築」

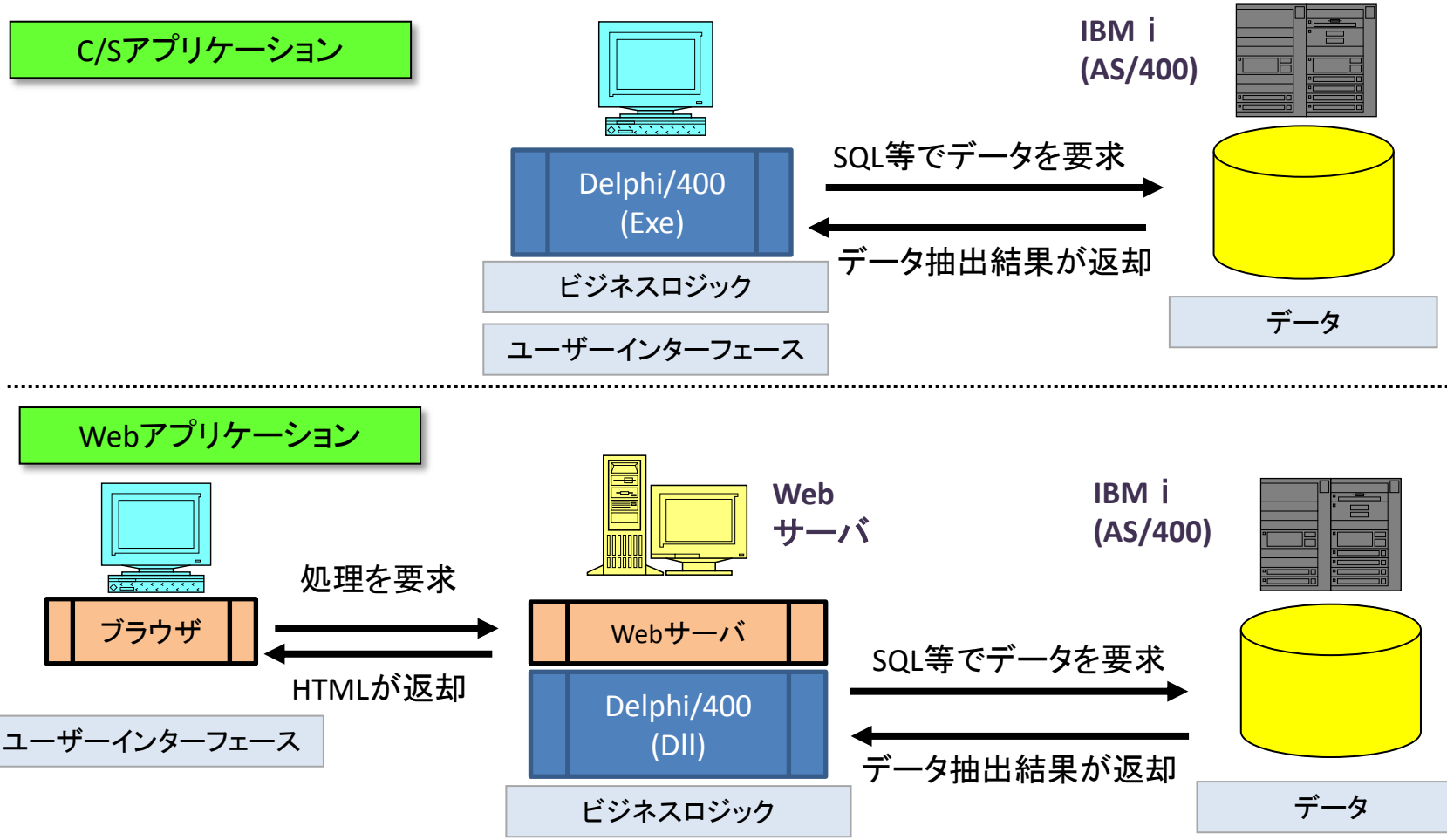
株式会社ミガロ.
システム事業部　プロジェクト推進室
小杉　智昭

【アジェンダ】

1. Webアプリケーションについて
2. Webアプリケーションの安全性向上
3. 安全性を高めた基幹システム
   データの公開

1．Webアプリケーションについて

C/Sアプリケーションと Webアプリケーションとの違い



■ C/Sアプリケーションと Webアプリケーションとの対比

	C/Sアプリケーション	Webアプリケーション
操作性	フォームを使用した柔軟な操作性 リアルタイムな応答が可能	ブラウザを前提とした操作性 HTTPリクエスト／レスポンスの繰り返し
開発生産性	TFormを使用した直観的な開発 コンポーネントにイベントを割り当ててビジネスロジックを作成	TIWFromを使用した直観的な開発 VCL for the Webを使用することで、C/Sアプリケーション同様の開発手法が可能
実行環境	Windows　PCに限定 GUIはWindowsのみで稼働	ブラウザが稼働するPC、スマートフォン等 ブラウザ上で実行される為、WIndows,、Linux、スマートフォン等あらゆる環境で稼働
アプリケーション配布	各クライアントPCへインストールが必要 更新の際も各クライアントPCへモジュールの置き換えが必要	サーバ環境のみインストール 更新の際は、Webアプリケーションサーバ環境のモジュール置き換えのみで良い

企業環境に従来型PC以外のスマートフォンやタブレットの重要性が向上！

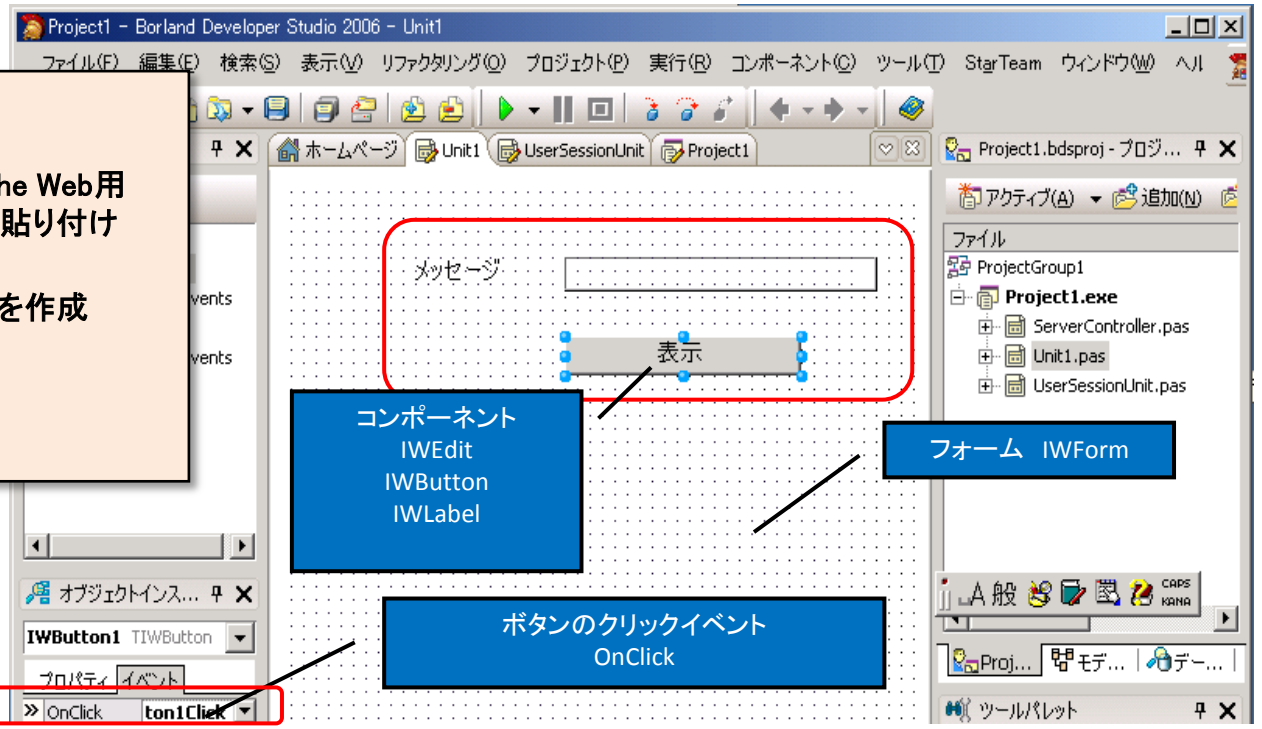
⇒　Webアプリケーションへの取り組みが重要

Delphi/400におけるWebアプリケーション開発

- VCL for the Web (IntraWeb)

＜開発手順＞

1. フォーム(IWForm)にVCL for the Web用コンポーネント(IWButton等)を貼り付けて、プロパティの設定
2. 必要に応じてイベントハンドラを作成
3. コンパイル
4. 実行（テスト）



C/Sアプリケーション同様の手法で開発可能！

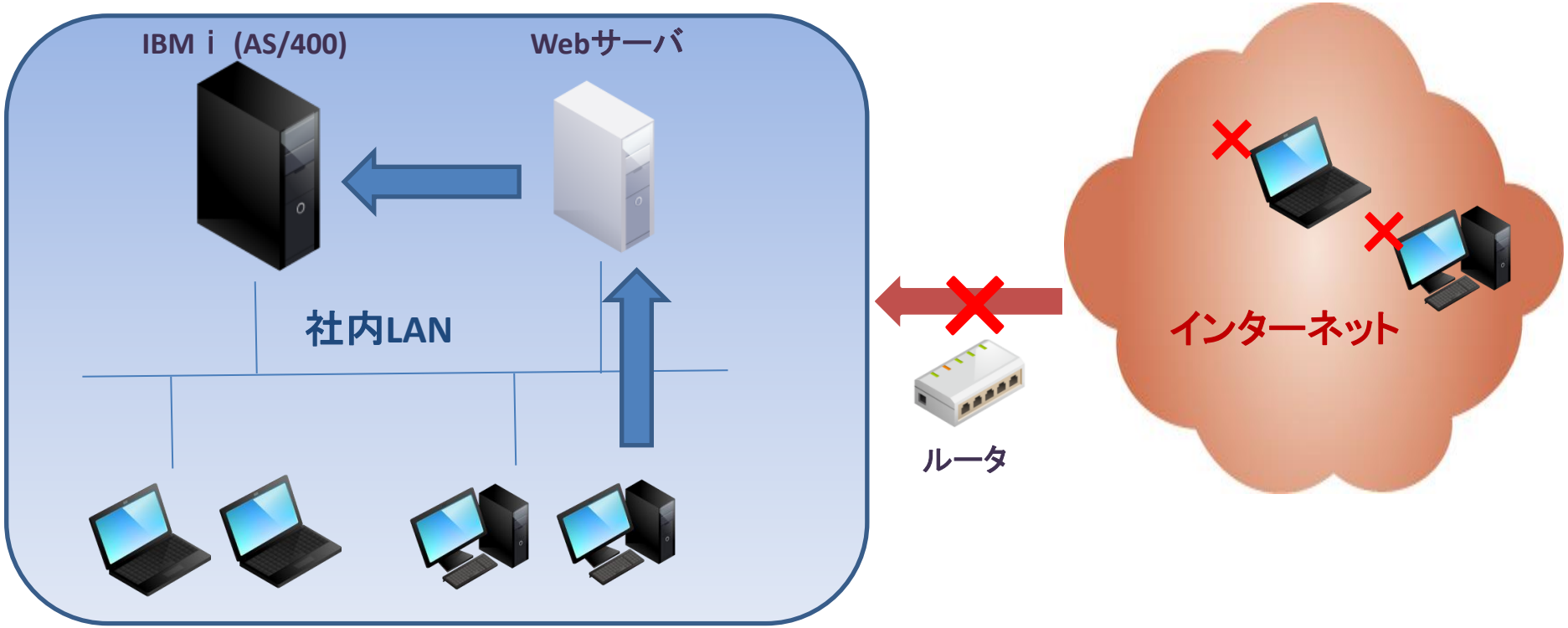
■ Webアプリケーションのパターン

- イントラネットアプリケーション
  - 社内ネットワークに限定
  - 社外秘情報を含むアプリケーションのWeb化
    - 販売管理システム
    - 社内掲示板

- インターネットアプリケーション
  - 外部ネットワークに公開
  - 情報公開により利便性が向上するアプリケーション
    - ネットショッピング
    - インターネットバンキング

■ イントラネットアプリケーション

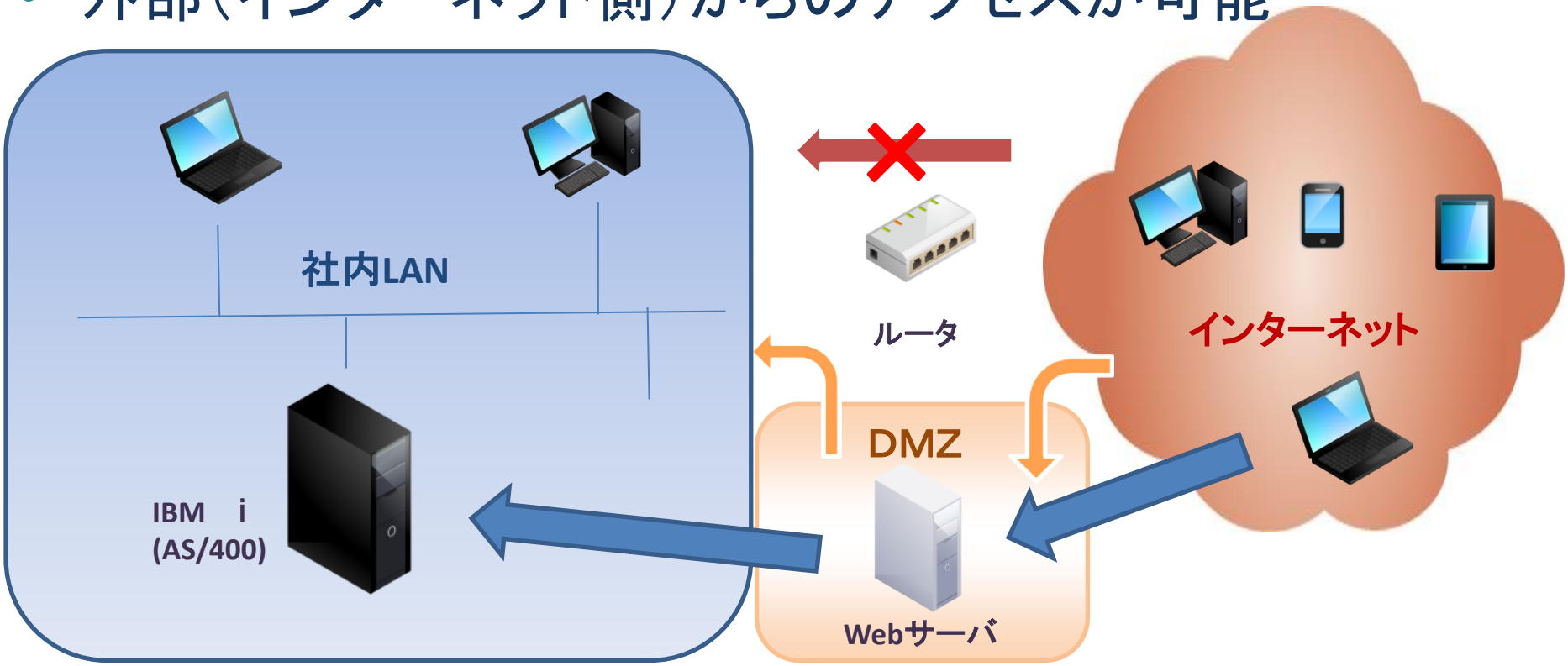
- 外部（インターネット側）からのアクセスは不可



社内利用限定のため、C/Sアプリケーション同様のセキュリティ考慮で
Webアプリケーションが開発可能

■ インターネットアプリケーション

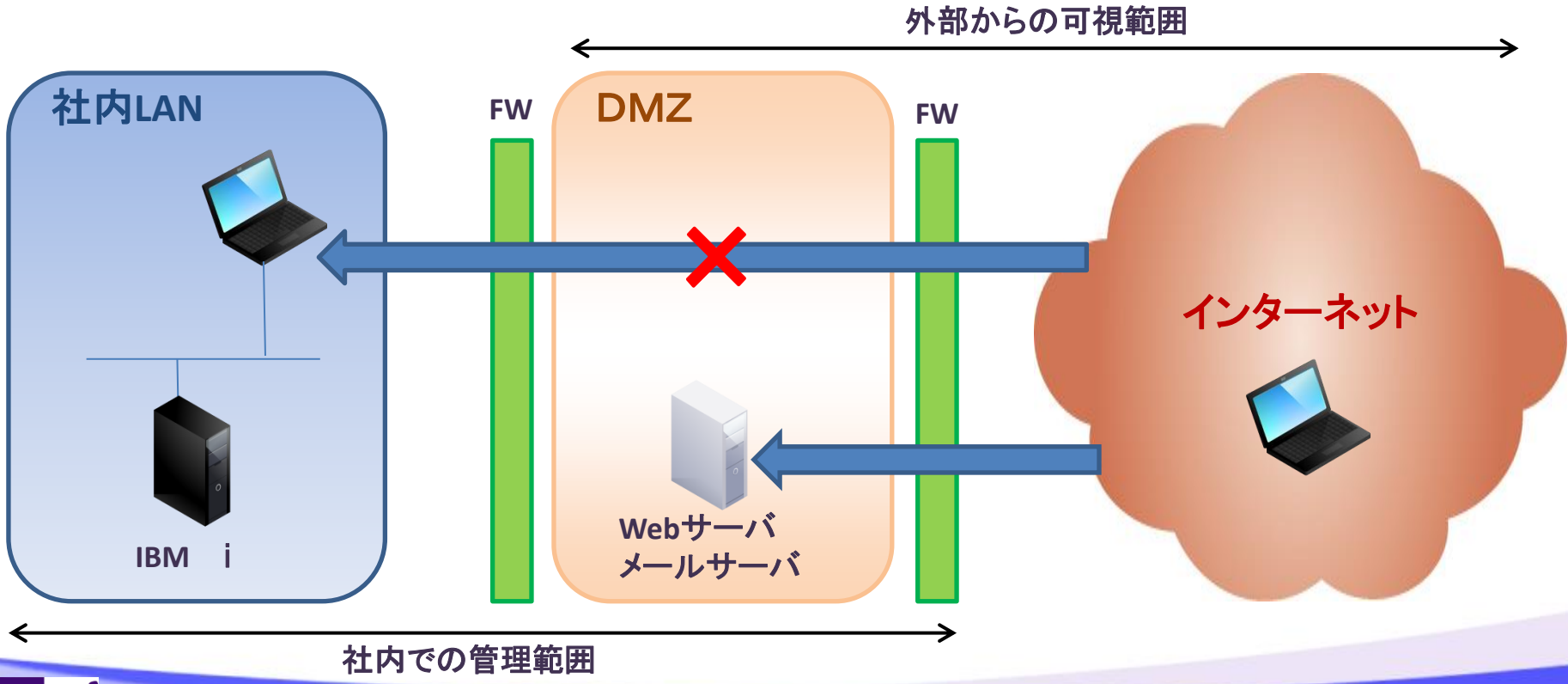
- 外部（インターネット側）からのアクセスが可能



Webサーバが公開されるため、外部からの不正アクセス等セキュリティの確保が必要

■ 外部公開用サーバの構成

- DMZ（非武装地帯）
  - ネットワーク上でファイアウォールによって包囲された、外部からも内部からも隔てられた特殊な領域のこと。



2. Webアプリケーションの安全性向上

■ Webサーバの公開による脅威について

サーバの乗っ取り

OSのセキュリテイホールをついてサーバへ侵入する

- サーバOSに対する最新セキュリティパッチの適用
- ファイアウォール等による通信ポートの制御
- 公開に不要なサービスの停止

データの漏えい

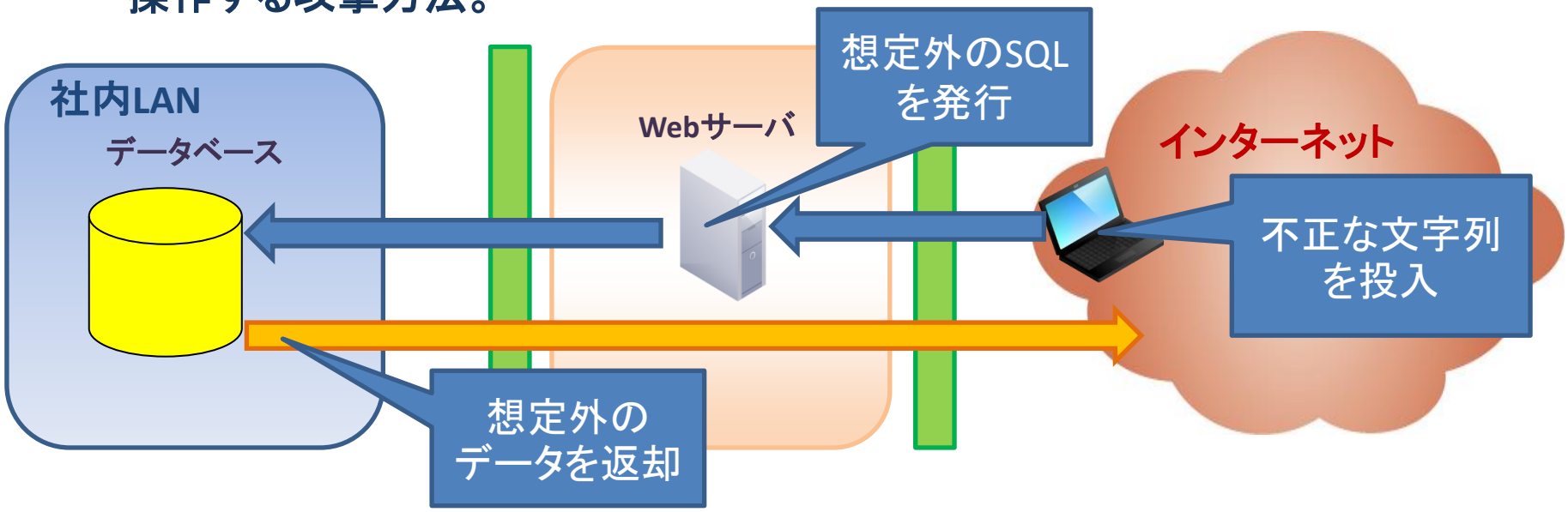
不正にサーバにアクセスし、データを盗み出す

- Webアプリケーションの脆弱性対策の適用

Webアプリケーション開発では脆弱性に対する考慮が必要！

■ Webアプリケーションの脆弱性

- SQLインジェクション
  - **アプリケーションのセキュリティ上の不備を意図的に利用し、アプリケーションが想定しないSQL文を実行させることにより、データベースシステムを不正に操作する攻撃方法。**



- 手法は古くから存在し、2005年頃　大手サイトで被害が多発
- 近年においても被害が発生
  - ソニーグループ個人情報漏洩問題　　（2011年）

SQLインジェクションの例

- ユーザーマスター（MCERTP）

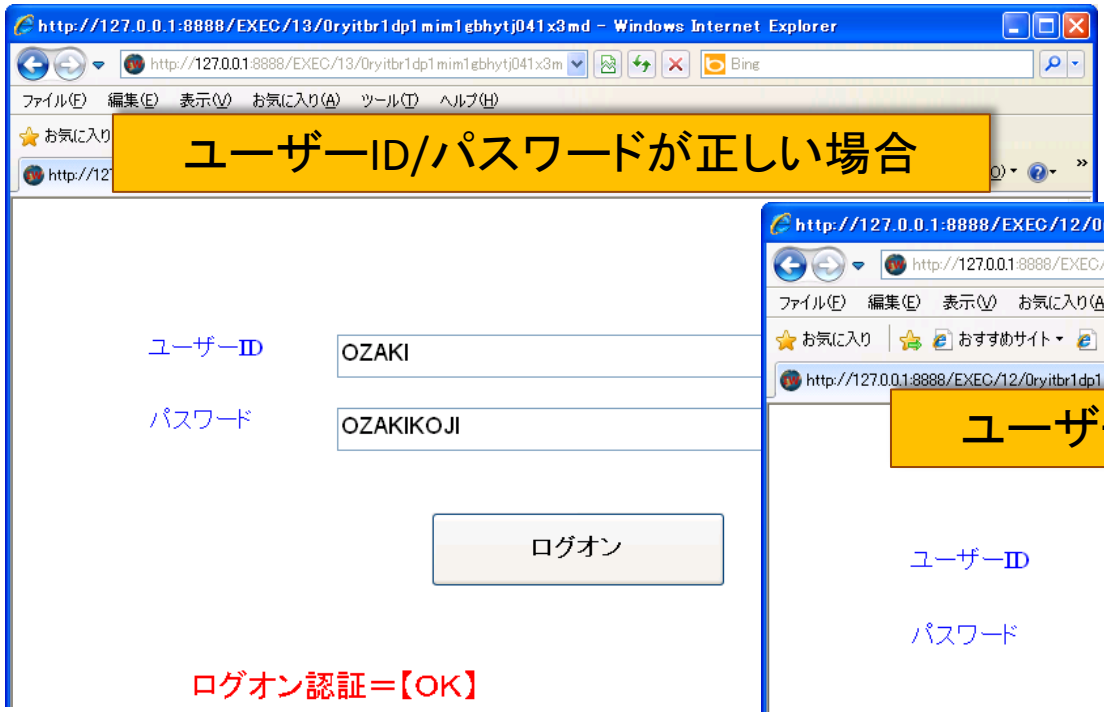
RTUSER （ユーザーID）	RTPSWD （パスワード）
KOSUGI	PASSWORD
OZAKI	OZAKIKOJI
YOSHIWARA	RAD
YAMADA	TARO

- 次のSQLを実行し、対象レコードが存在した場合ログオンOKとする。

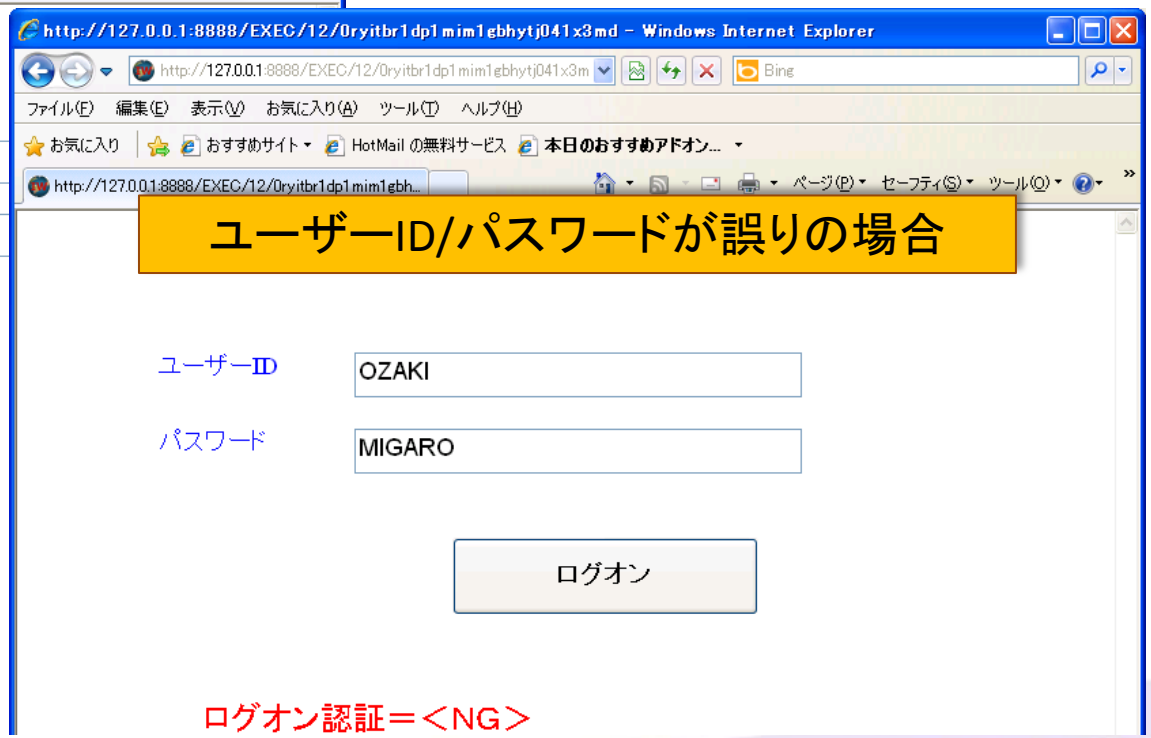
```
SELECT  *  FROM  MCERTP
WHERE   RTUSER  =  [画面入力ユーザーID]
AND     RTPSWD  =  [画面入力パスワード]
```

SQLインジェクションの例

- 想定された動作の場合



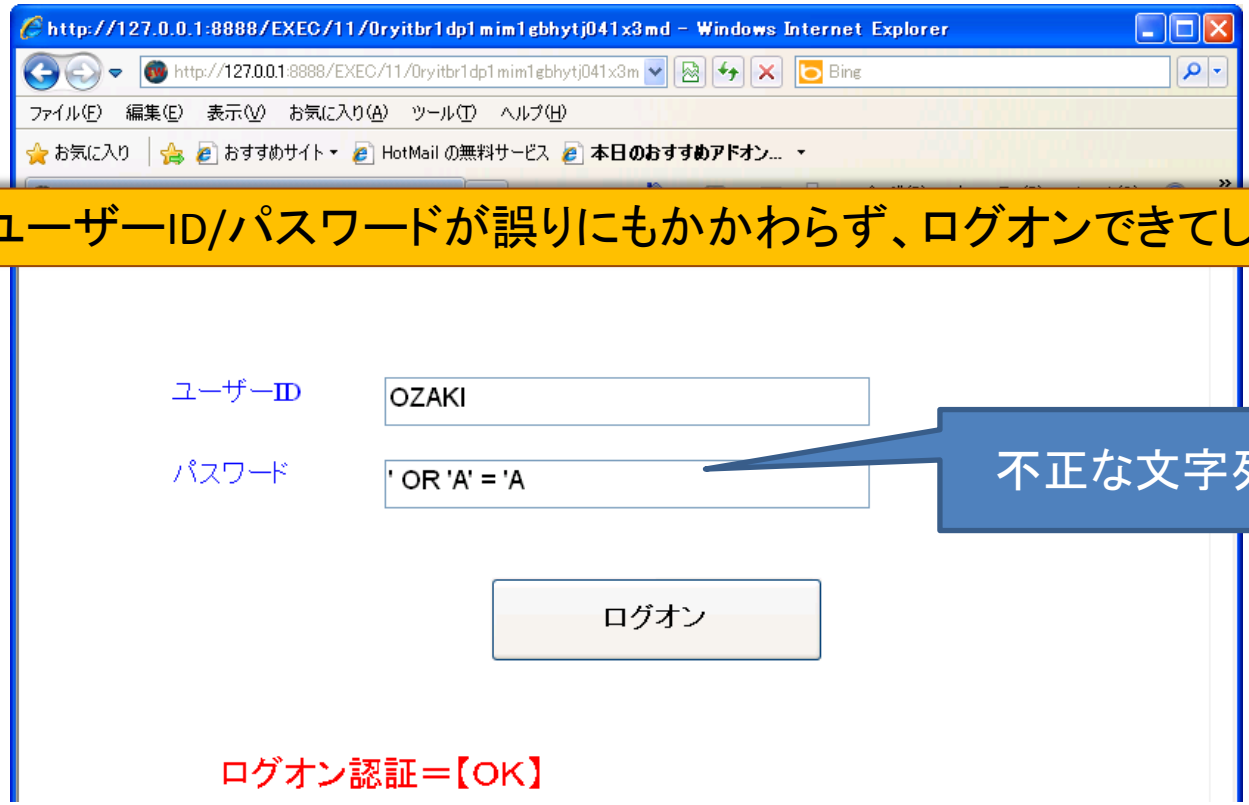
ユーザーID/パスワードが正しい場合



ユーザーID/パスワードが誤りの場合

SQLインジェクションの例

- 想定外の動作



ユーザーID/パスワードが誤りにもかかわらず、ログオンできてしまう

不正な文字列を投入

ユーザーID/パスワードが不明であっても、システムにログオンできてしまう危険性（脆弱性）がある

■ SQLインジェクションの例

```
procedure TfrmLogon.btnLogonClick(Sender: TObject);
var
  sSQLStr: String; // SQL文字列
begin
  //SQL文字列の作成
  sSQLStr := 'SELECT * FROM MCERTP '
           + 'WHERE RTUSER = ''' + edtUSERID.Text + ''' '
           + 'AND RTPSWD = ''' + edtPASSWORD.text + ''' ';
    ～（以下省略）～
```

画面で入力された文字列をSQL文字列にセット

※ 文字列項目のため、前後に引用符を使用

不正な文字列をセットしたSQL文 （入力文字列は下線部分）

```
SELECT * FROM MCERTP
WHERE RTUSER = 'OZAKI' AND RTPSWD = '' OR 'A' = 'A'
```

偽(False)　　真(True)

⇒ OR(論理和)であるため、常に処理結果が真(True)となってしまう！

■ SQLインジェクションの対策

入力文字列の妥当性チェックを強化

- 「‘」「“」「;」「%」等の文字列に対しエスケープを考慮する。
  - ◆ 「’」 → 「’’」

パラメータクエリーの活用

- SQL文のパラメータを使用することで、特殊文字を含んでも確実に一つの文字列として処理可能にする。
  - ◆ **パラメータ（ParamByName等） の使用**

SQLを使用する場合、パラメータクエリーで処理を行う！

■ SQLインジェクションの対策例

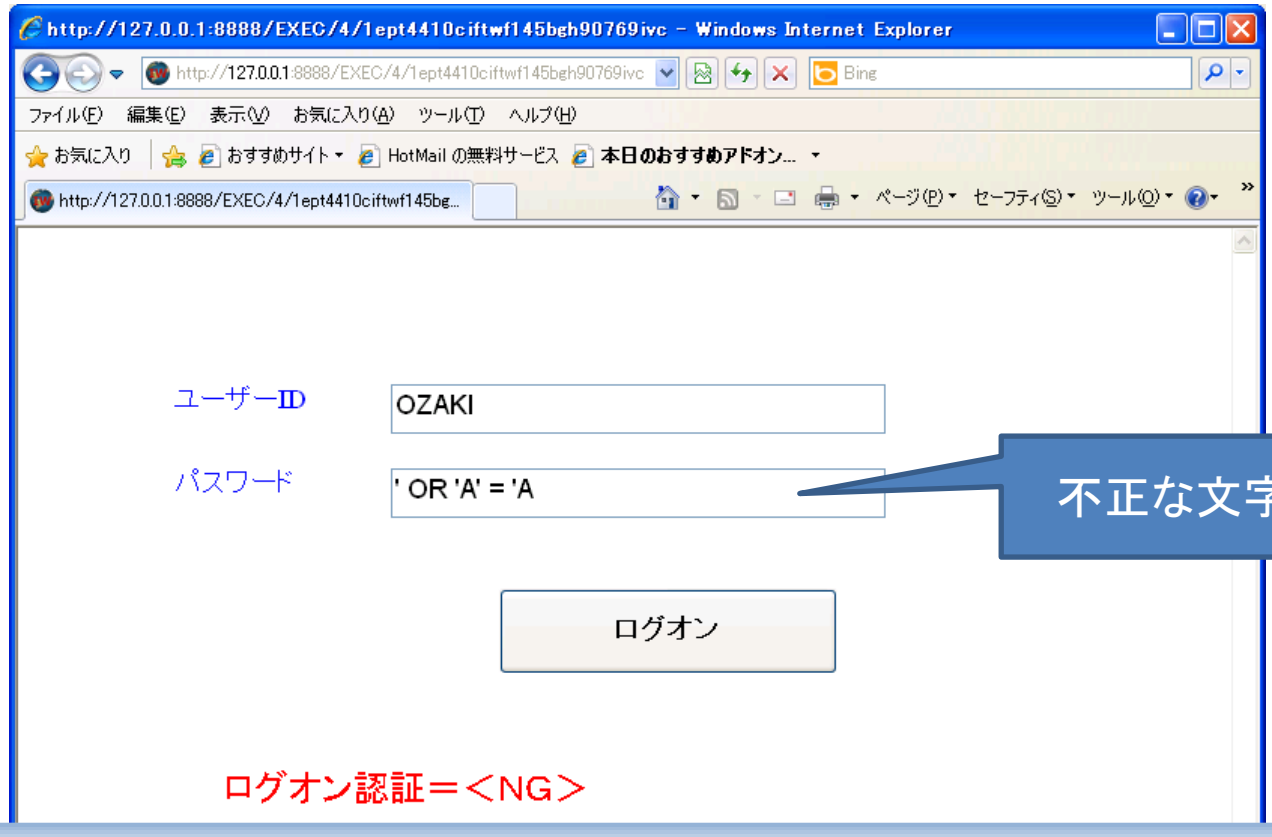
```
procedure TfrmLogon.btnLogonClick(Sender: TObject);
var
  sSQLStr: String;    // SQL文字列
begin
  //SQL文字列の作成
  sSQLStr := 'SELECT * FROM MCERTP '
           +  'WHERE RTUSER = :USERID '     // <-- パラメータ名：USERID
           +  'AND RTPSWD = :PSWD ' ;       // <-- パラメータ名：PSWD

  qryLogon.SQL.Text := sSQLStr;

  // パラメータのセット
  qryLogon.ParamByName('USERID').AsAnsiString := edtUSERID.Text;
  qryLogon.ParamByName('PSWD').AsAnsiString := edtPASSWORD.Text;

     ～（以下省略）～
```

画面で入力された文字列をパラメータにセット

画面入力値は、確実に一つの値として処理されるため、正しく動作する

SQLインジェクションの対策例

- 対策後のアプリケーション

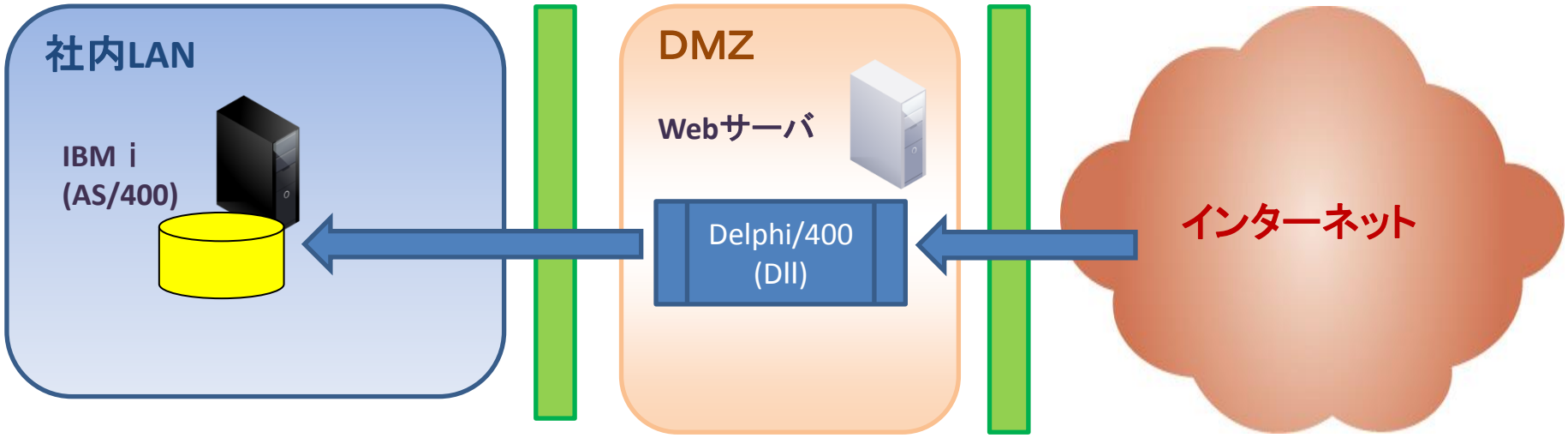


特殊文字が含まれていても、一つの値として処理されるため、想定どおりの結果を返す。

3. 安全性を高めた基幹システム データの公開

Delphi/400を使用したWebアプリケーション

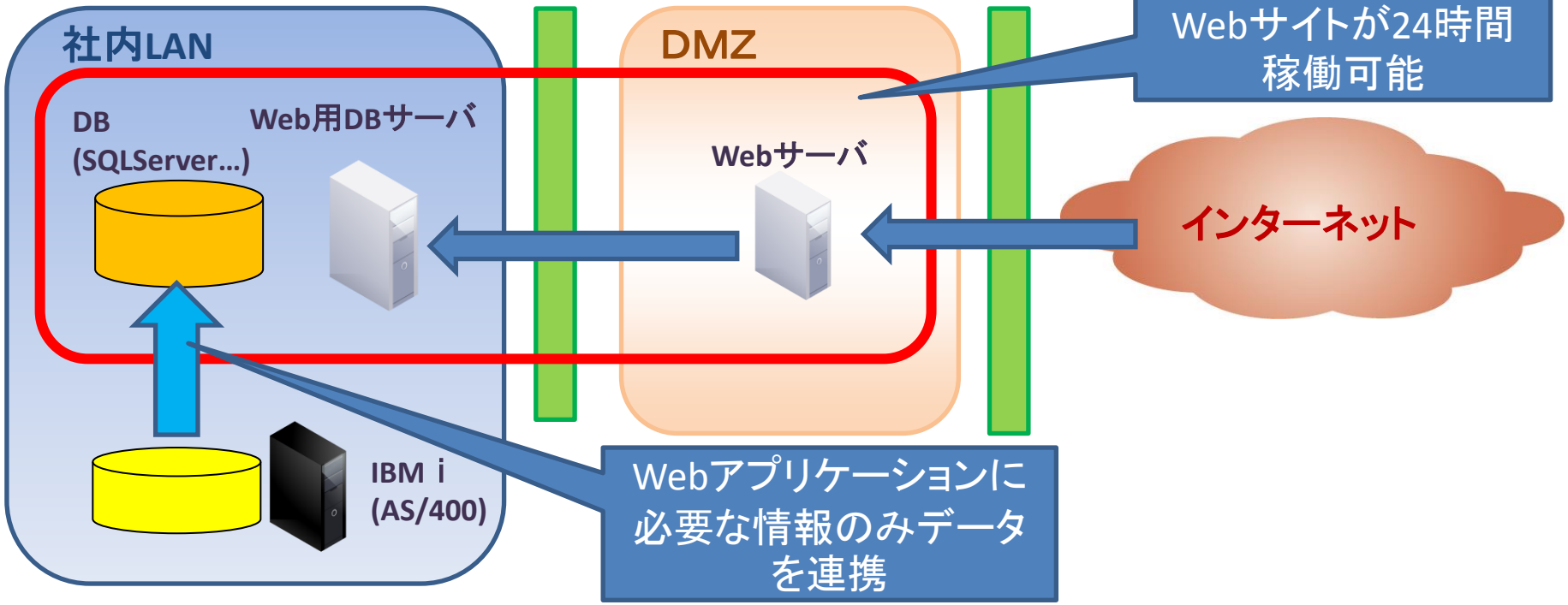
- **WebサーバアプリケーションからIBM i (AS/400) にアクセス**
  - DMZ領域から社内LAN向きに、Delphi/400用ポートを解放することで、IBM i にアクセスするWebアプリケーションが構築可能



サービス提供時間

- Webサイトを24時間オープンさせるには、IBM i を24時間稼働させなければならない。

■ Web用DBサーバとの連携

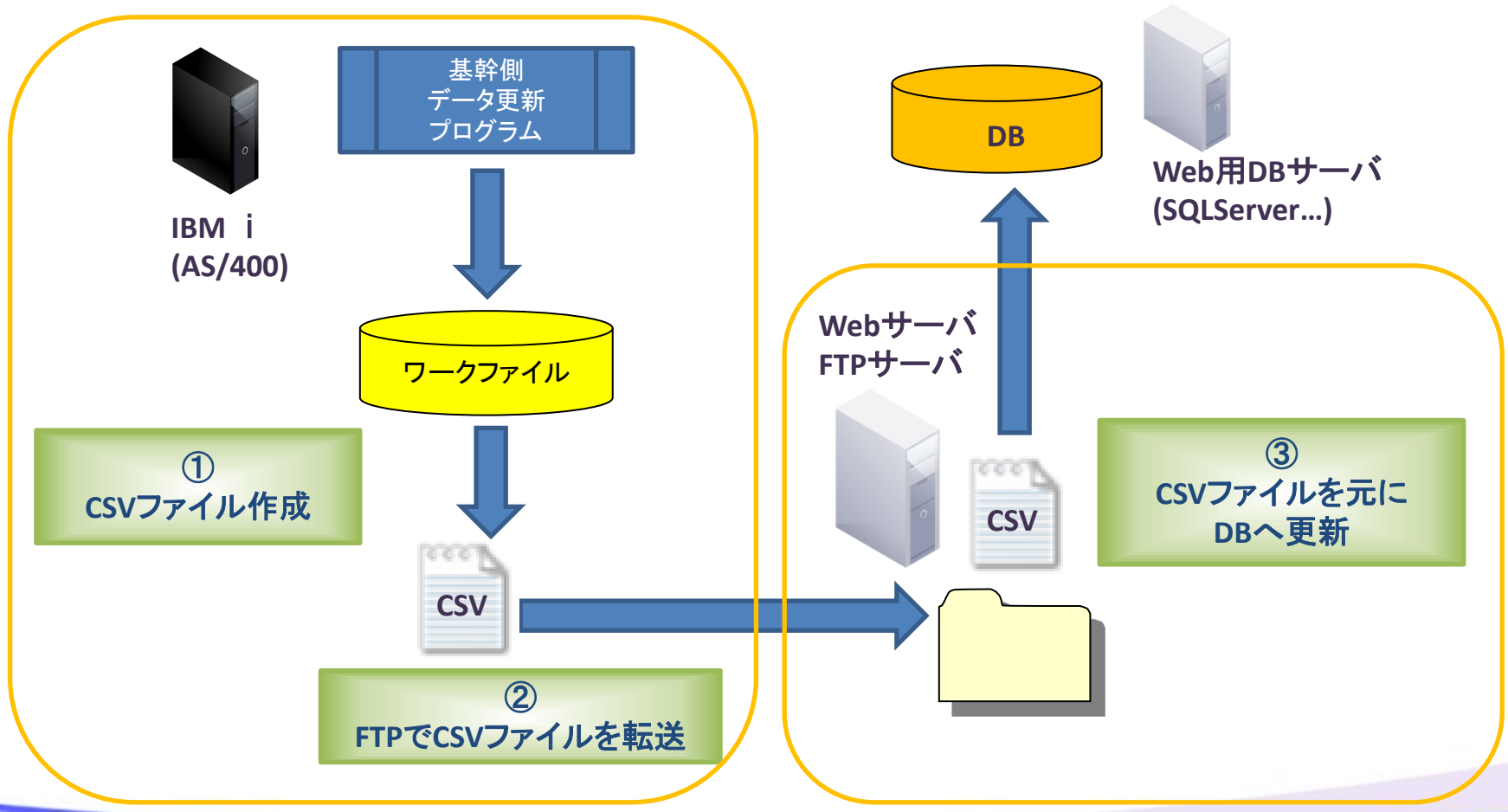


Web用DBサーバを利用することにより24時間運用が可能

- Web用DBサーバを用意すれば、IBM i（AS/400)はバックアップ等で停止しても、Webサイトを24時間稼働させられる。
- 仮にDBサーバへの不正アクセスがあったとしても、被害対象となるデータがWeb公開分のみに限定される。

■ データ連携の仕組み

- Web用DBサーバへのデータ受渡し



■ ① IBM i（AS/400）上でのCSVファイル作成

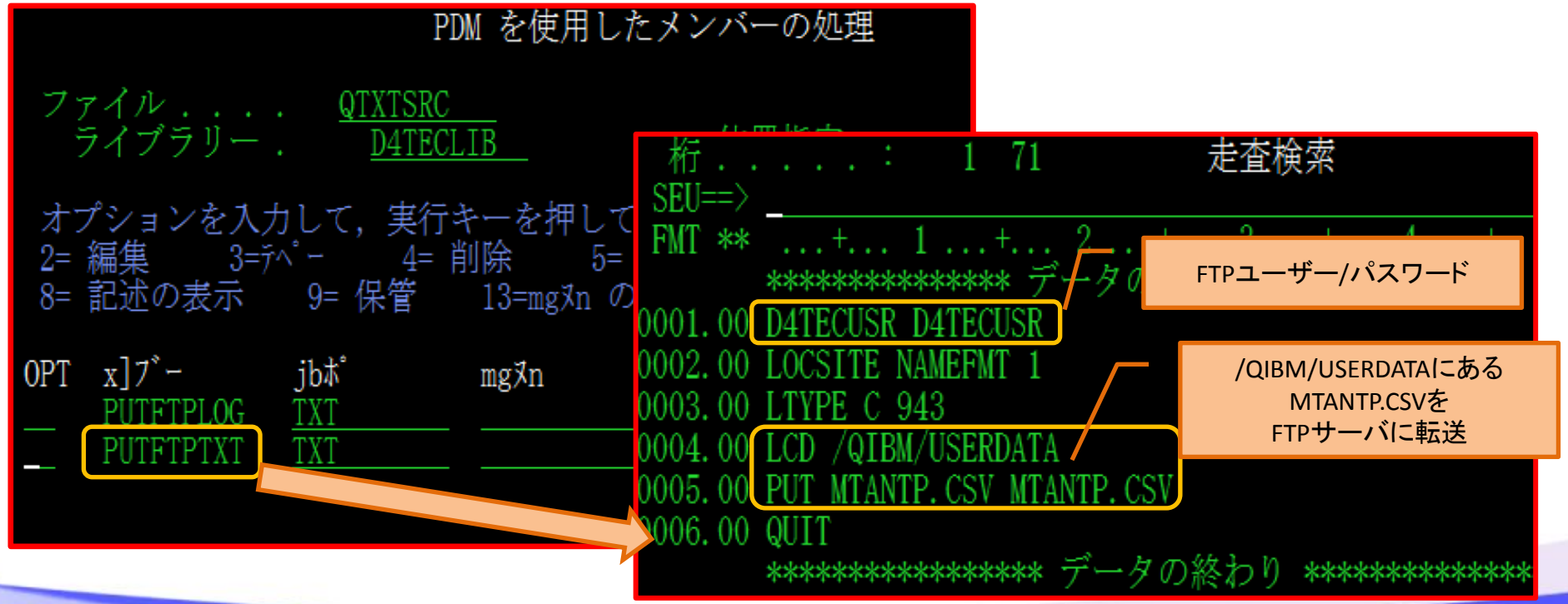
- CPYTOIMPFコマンド
  - コピー元ファイルからインポート・ファイルにデータをコピー
    - （例） コピー元物理ファイル : D4TECLIB/MTANTP (担当者マスタ)
      コピー先CSVファイル : QIBM/USERDATA/MTANTP.CSV (IFS上)


```
0015.00
0016.00      /*  ＣＳＶファイル作成  */
0017.00                CPYTOIMPF  FROMFILE(D4TECLIB/MTANTP) +
0018.00                           TOSTMF('QIBM/USERDATA/MTANTP.CSV') +
0019.00                           MBROPT(*REPLACE) STMFCODPAG(943) +
0020.00                           RCDDLM(*CRLF) DTAFMT(*DLM) STRDLM(*NONE) +
0021.00                           NULLIND(*NO)
0022.00                  MONMSG    MSGID(CPF0000)
0023.00
```

■ ② IBM i（AS/400）からFTP転送 （PUT）

- FTPコマンド
  - 指定したFTPサーバにデータを転送
    - ➢ FTP実行コマンドをINPUTファイルに登録

INPUT： FTP実行コマンド（TXT） OUTPUT： FTP実行ログ（TXT）



■ ② IBM i（AS/400）からFTP転送 （PUT）

- FTPコマンド
  - ➢ FTP RMTSYS('[接続先FTPサーバ]')

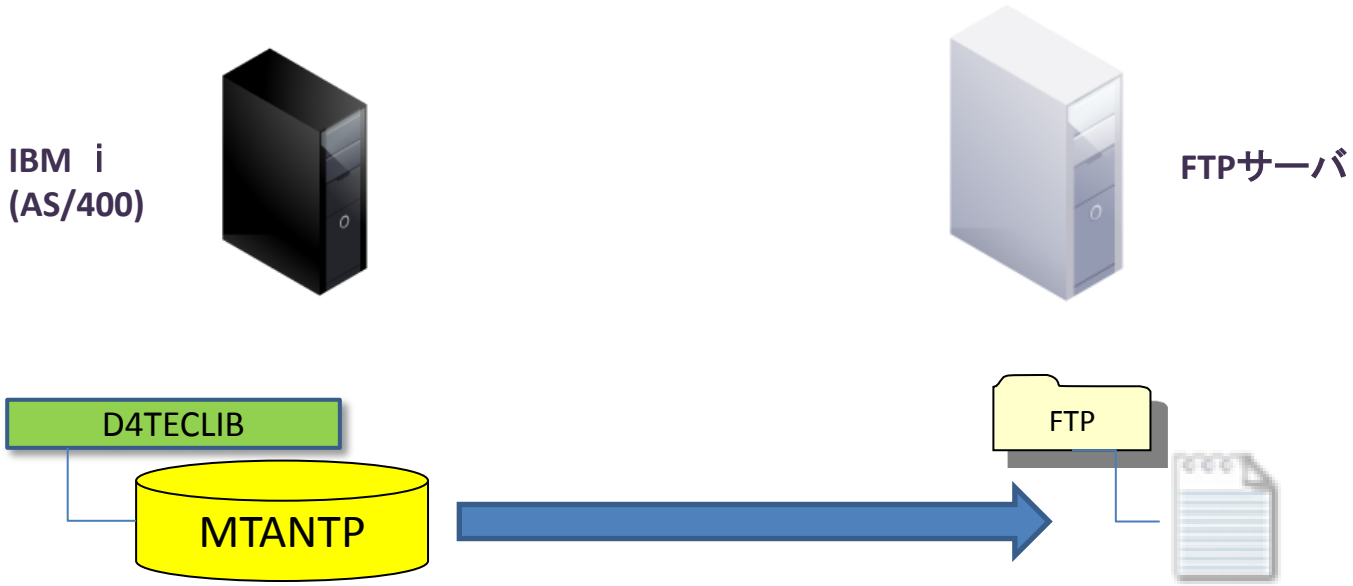
```
 桁 . . . . . :   1  71          編集                    D4TECLIB/QCLSRC
SEU==>                                                    FTP02C
FMT **  ...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7
0008.00  PGM
0009.00     /*  FTP 関連ファイル消去  */
0010.00       CLRPFM FILE(D4TECLIB/QTXTSRC) MBR(PUTFTPLOG)
0011.00
0012.00     /*  ファイル一時変更  */
0013.00       OVRDBF FILE(INPUT)  TOFILE(D4TECLIB/QTXTSRC) MBR(PUTFTPTXT)
0014.00       OVRDBF FILE(OUTPUT) TOFILE(D4TECLIB/QTXTSRC) MBR(PUTFTPLOG)
0015.00
0024.00     /*  FTP データ転送  */
0025.00       FTP RMTSYS('192.168.0.XXX')
0026.00
0027.00     /*  一時変更削除  */
0028.00       DLTOVR  FILE(INPUT OUTPUT)
0029.00
0030.00  ENDPGM
         ********** データの終わり **********
```

FTPログファイルのクリア

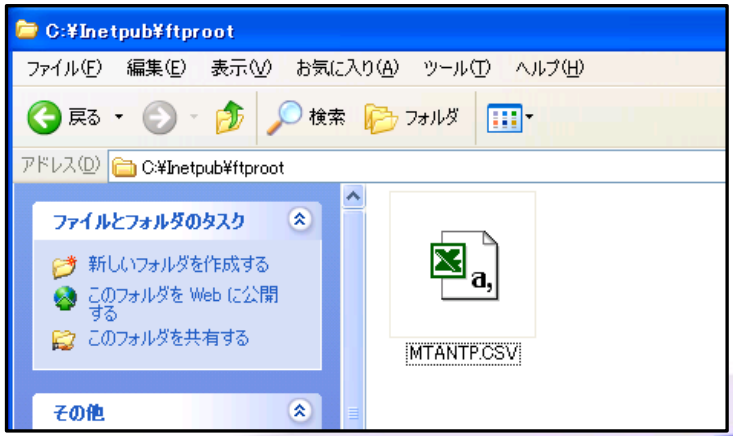
INPUT/OUTPUTファイルの指定

FTP実行

■ （実演）CSVファイル作成からFTP転送までの実行



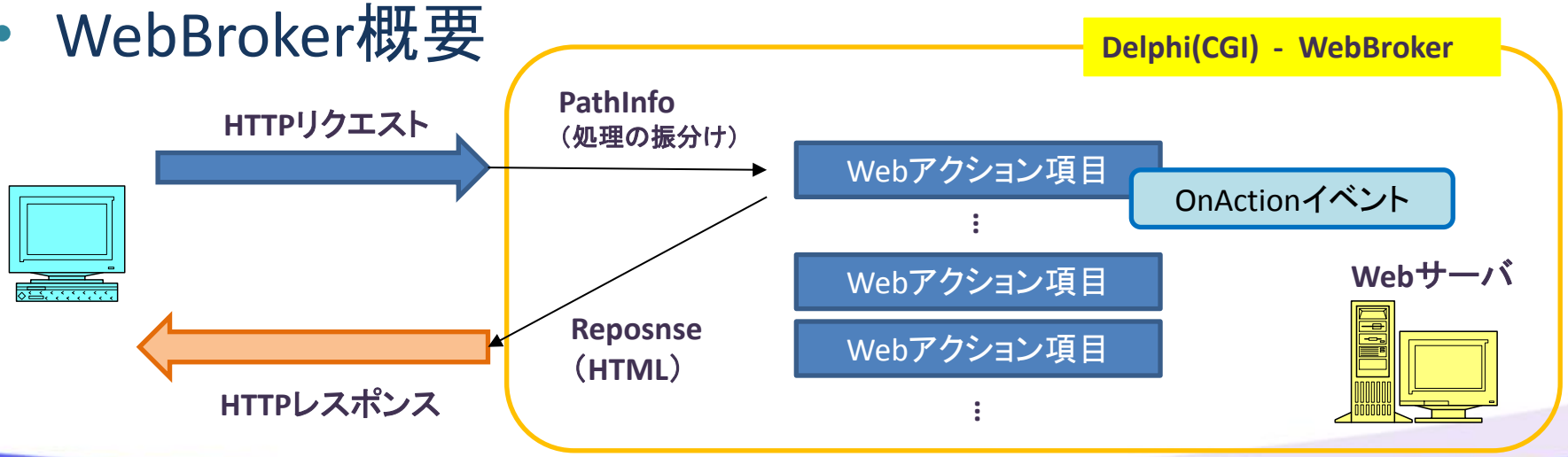

```
                    プログラム呼び出し  (CALL)

選択項目を入力して，実行キーを押してください。

プログラム . . . . . .   FTP01C_       名前
  ライブラリー . . . . . .   D4TECLIB     名前 , *LIBL, *CURLIB
パラメーター . . . . . .
____
            値の続きは＋
____
```



■ ③ CSVファイルからDBを更新

- CGIプログラム（Delphi）
  - CSVファイルを読み取り、DBサーバ上のテーブルを更新
    - （例） 元CSVファイル : MTANTP.CSV (担当者マスタ CSVファイル）
      更新先DB : MTANTP.db (Paradox)

- WebBroker概要



■ CGIプログラム（ご参考）

http://192.168.0.XXX/bin/DataConv.exe

処理結果（HTML）

```
<HTML>
<HEAD><TITLE>DataConv</TITLE></HEAD>
<BODY>SUCCESS</BODY>
</HTML>
```

```
procedure TWebModule1.WebModule1WebActionItem1Action(Sender: TObjec
  Request: TWebRequest; Response: TWebResponse; var Handled: Boolean
const
  RetStr = '<HTML><HEAD><TITLE>DataConv</TITLE>' +
           '</HEAD><BODY>%s</BODY></HTML>';          // レスポンス用HTML
  FTPPath = 'C:¥Inetpub¥ftproot¥';                   // FTPファイルPATH
  CSVFileName = 'MTANTP.CSV';                        // CSVファイル名
var
  i, j: Integer;
  slCSVList: TStringList;     // CSVファイル用文字列リスト
  slLineList: TStringList;    // 行情報用文字列リスト
  sMsgTxt: String;            // レスポンスHTML用文字列
begin
  //レスポンス種類（HTML）の指定
  Response.ContentType := 'text/html; charset=Shift_JIS';

  try
    //CSVが存在する場合、最新の内容に置き換える
    if FileExists(FTPPath + CSVFileName) then
    begin
      // 現在のレコードをクリア
      Table1.EmptyTable;

      //テーブルをオープン
      Table1.Active := True;
      try
        // 文字列リストを生成
        slCSVList := TStringList.Create;
        slLineList := TStringList.Create;
        try
          //CSVファイルを文字列リストに読み込み
          slCSVList.LoadFromFile(FTPPath + CSVFileName);
```

レスポンスに返すデータ種類（HTMLファイル）

CSVファイルを文字列リストに読み込む

```
        //レコード数分繰り返す
        for i := 0 to slCSVList.Count - 1 do
        begin
          //行ごとの情報（カンマ区切り文字列）を文字列リストにセット
          slLineList.CommaText := slCSVList[i];

          //テーブルにレコードを追加
          Table1.Append;
          //フィールドに値をセット
          for j := 0 to slLineList.Count - 1 do
            Table1.Fields[j].AsString := slLineList[j];

          //テーブルを更新
          Table1.Post;
        end;
      finally
        //文字列リスト開放
        slLineList.Free;
        slCSVList.Free;
      end;
    finally
      //テーブルを閉じる
      Table1.Active := False;
    end;

    //CSVファイルの削除
    DeleteFile(FTPPath + CSVFileName);
  end;
  // 正常終了レスポンスの返却
  sMsgTxt := Format(RetStr, ['SUCCESS']);
  Response.Content := sMsgTxt;
  Response.StatusCode := 200;
except
  //例外処理
  on E: Exception do
  begin
    // エラー終了レスポンスの返却
    sMsgTxt := Format(RetStr, [E.Message]);
    Response.Content := sMsgTxt;
    Response.StatusCode := 599;
  end;
end;
end;
```

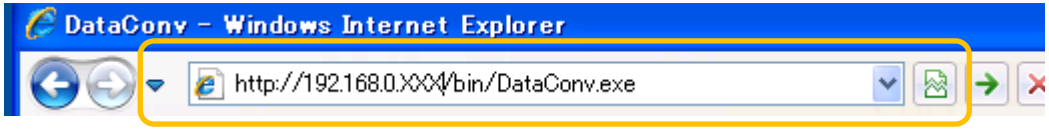
文字列リストを行ごとに読み取り、データセットにレコードを追加

処理に使用したCSVファイルを削除

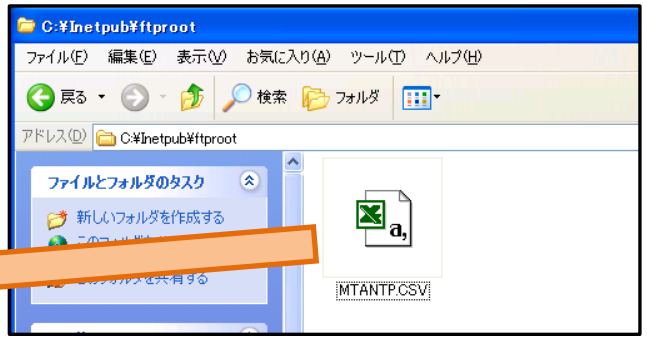
正常終了時、ステータスコード=200 (Success)を返却

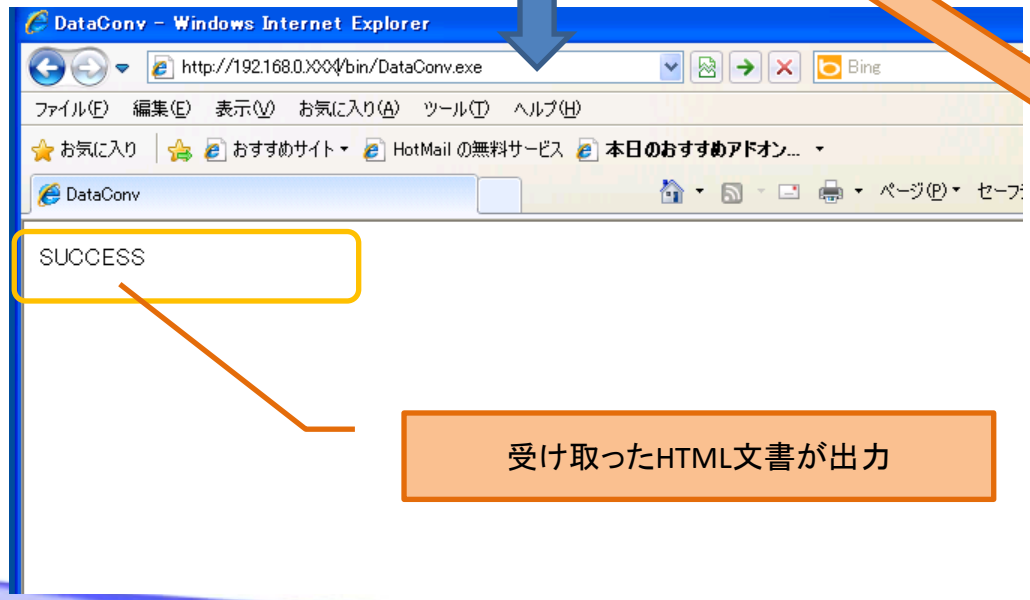
例外発生時、HTMLのBODY部にエラーメッセージをセット

■ （実演）CGIプログラムの実行

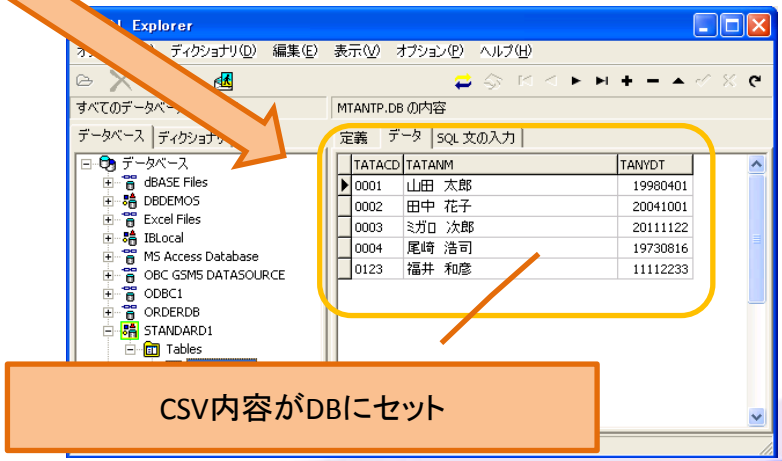


DataConv.exe
（CGI）



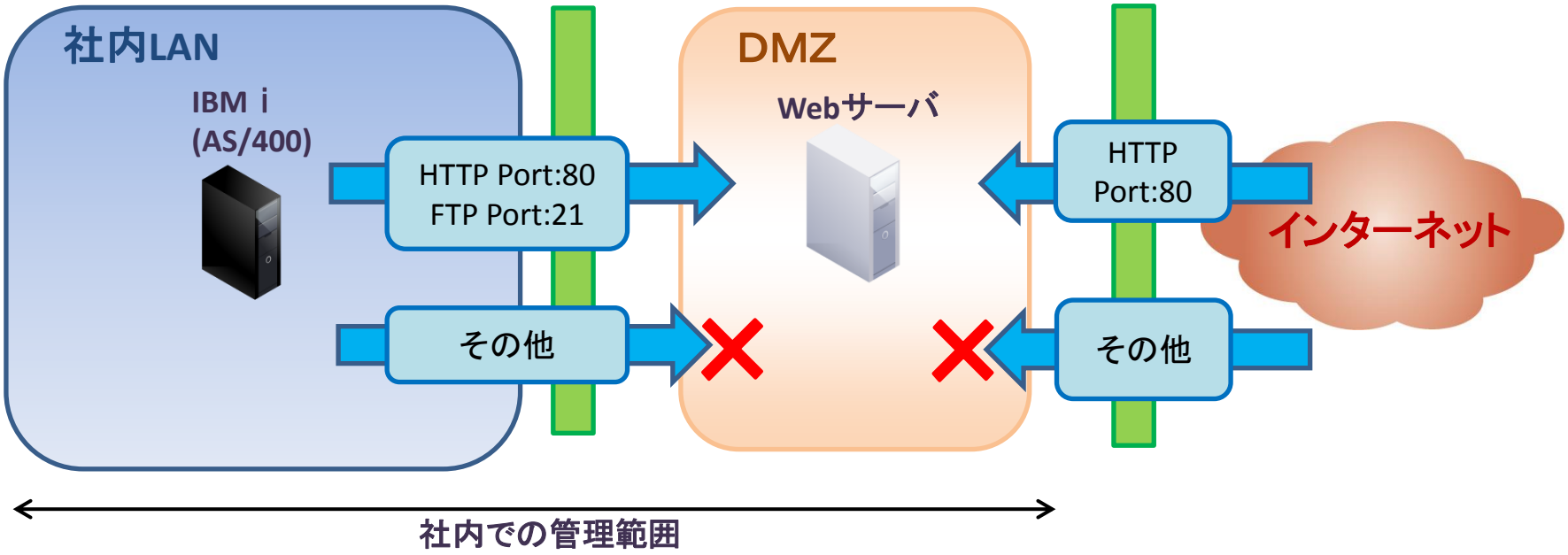


受け取ったHTML文書が出力



CSV内容がDBにセット

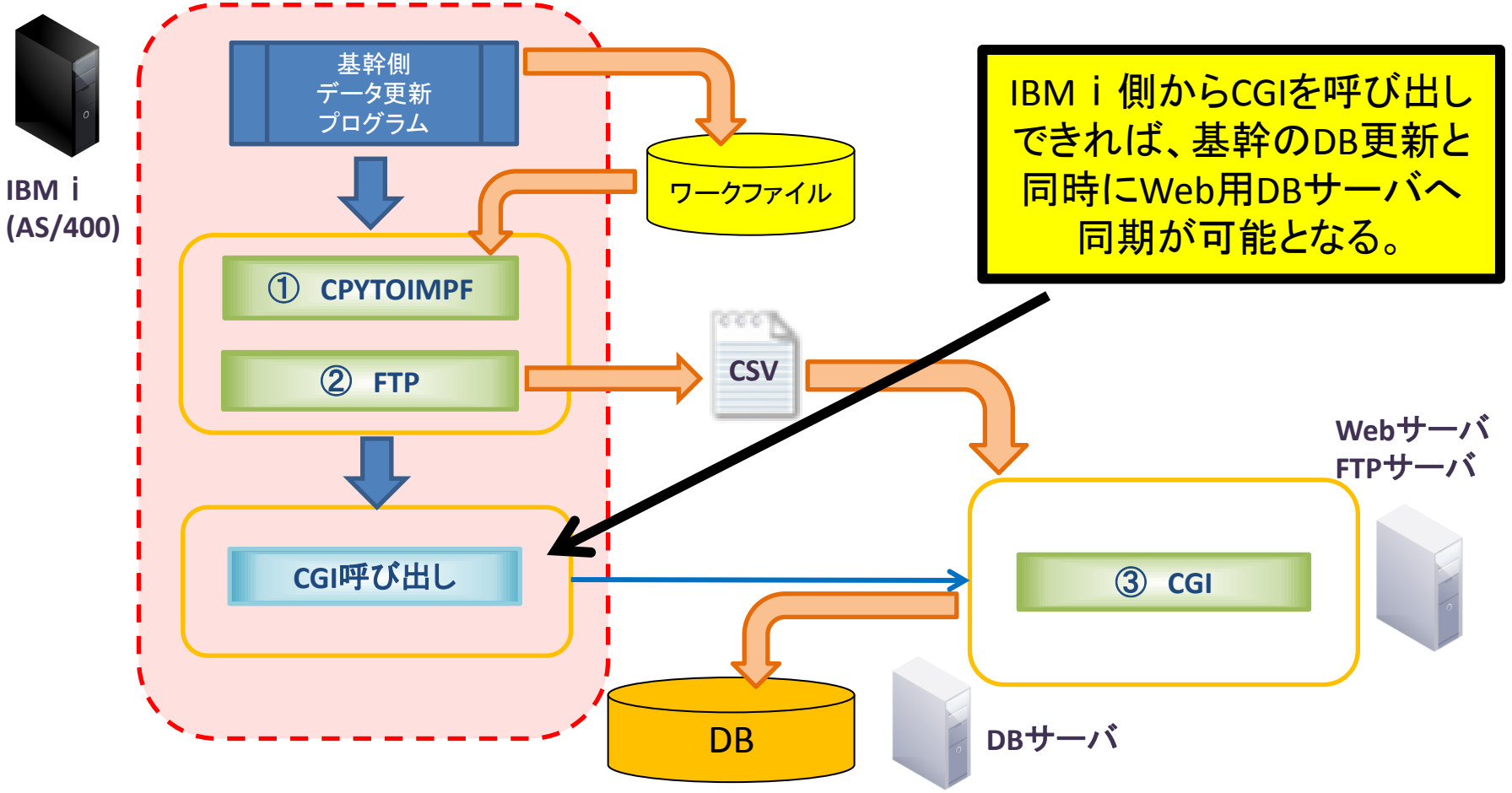
■ なぜCGIプログラムか？



社内側からDMZ領域へのアクセスにおいても安全性が確保可能

- CGIプログラムは、HTTPプロトコルでの通信となるため、不要なポートを開ける必要がなくなる。

IBM i 更新データとDBサーバとの同期



IBM i 側からCGIを呼び出しできれば、基幹のDB更新と同時にWeb用DBサーバへ同期が可能となる。

IBM i 上から、CGIプログラムを直接CALLできないか？

■ IBM i からCGIプログラムの呼び出し

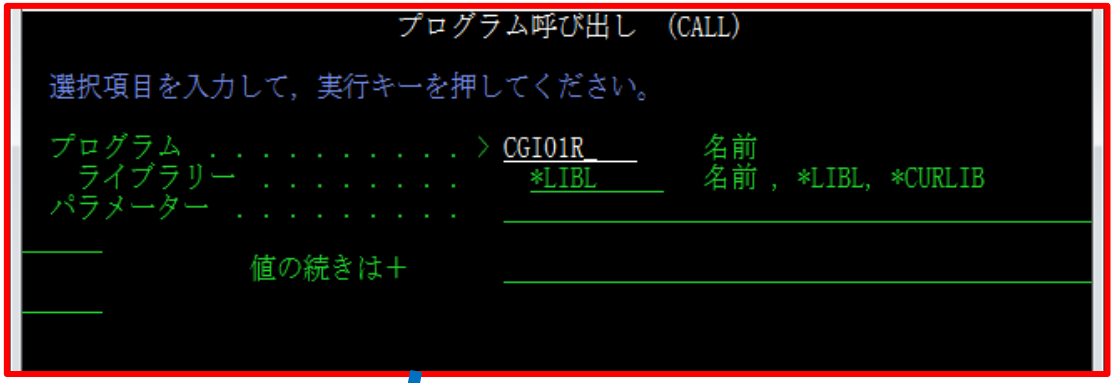
- HTTP APIを使用　⇒IBM i 上で動作するHTTPクライアント
  - HTTPAPI (オープンソース) http://www.scottklement.com/httpapi/httpapi_xml.html

```
0001.00 H DFTACTGRP(*NO) ACTGRP(*NEW) BNDDIR('HTTPAPI')
0003.00
0004.00 D/copy qrpglesrc,httpapi_h
0006.00 D rc              s              10I 0
0007.00 D msg             s              52A
0008.00 D URL             s             256A
0009.00
0010.00 c                 callp     http_debug(*ON)
0011.00
0012.00 c                 eval      URL = 'http://192.168.0.XXX' +
0013.00 c                           '/bin/DataConv.exe'
0015.00 c                 eval      rc = http_url_get(URL:
0016.00 c                           '/tmp/DataConv.html')
0017.00
0018.00 c                 if        rc <> 1
0019.00 c                 eval      msg = http_error
0020.00 c****              <<実行時エラー処理を記述>>
0021.00 c                 endif
0023.00 c                 eval      *inlr = *on
```

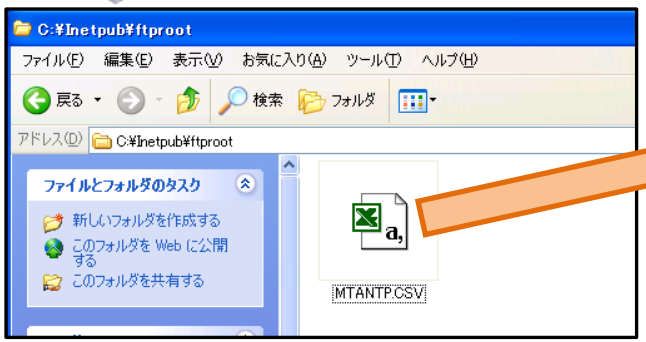
HTTPリクエストを行う

■（実演）IBM i からCGI呼び出しプログラムの実行

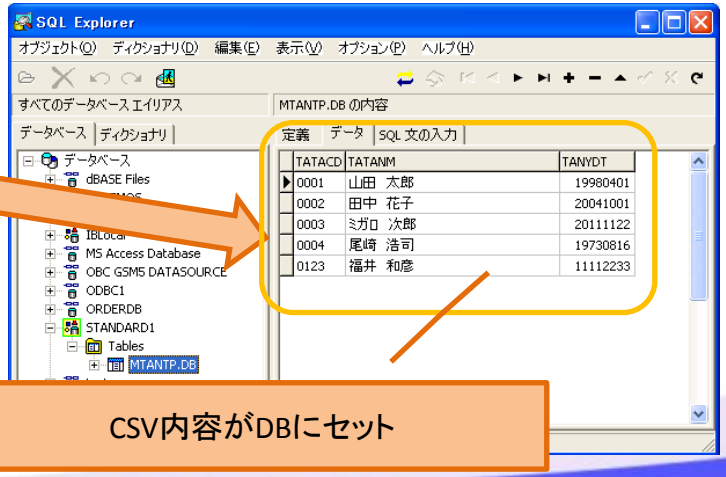
IBM i
(AS/400)



Webサーバ



DataConv.exe
（CGI）



CSV内容がDBにセット

まとめ

■ まとめ

1. Webアプリケーションについて

- C/SアプリケーションとWebアプリケーションとの違い
- イントラネットとインターネットについて

2. Webアプリケーションの安全性向上

- Webアプリケーションにおける脆弱性対策
  - SQLインジェクション対策手法

3. 安全性を高めた基幹システムのデータ公開

- IBM i とWeb用DBサーバとの連携手法
  - CSVファイル作成
  - FTP転送
  - CGIプログラムの呼び出し