```
/**
 * プロジェクトフレームワークサーブレットクラス。プロジェクト共通の基本処理フロー
 * を定義します。
 */
public  abstract    class   EZGenericServlet    extends EZServlet
{
    /**
     * 現在のセッション状態は、自サーブレットの処理を行ってよいかどうかを判定する。
     */
    public  abstract    boolean isReadyToGo (EZGenericServletContext context);

    /**
     * アプリケーション業務ロジック。
     */
    public  abstract    HTMLComponent   process (EZGenericServletContext context);

    /**
     * EZServletから呼び出される、リクエストに対応する処理。
     */
    public final    void processApplication(EZServletContext _context)
    {
        EZGenericServletContext context = (EZGenericServletContext)_context;
        EZGenericServletConfig  config = (EZGenericServletConfig)context.getConfig();
        try{
            if (! isReadyToGo (context))
            {
                //  通るべき道に戻す。
                ....
            }
            else
            {
                try{
                    HTMLComponent output = process (context);
                    ....
                    context.setOutput (output);
                } catch (com.ngMAT.servlet.EZSResponseWasRedirectedException e)
                {
                    throw e;
                } catch (Exception e)
                {
                    context.getLog().printStackTrace (e);
                    ....
                }
            }
        } catch (PageIsNotAvailableException ex)
        {
            //  使えませんページを表示
            context.setOutput (config.NotAvailableTemplate);
        }
    }
}

/**
 * プロジェクトフレームワークサーブレットコンテキストクラス。リクエストを受け
 * 取ってから、レスポンスを返すまでの間だけ一時的に使用するリソースの管理を
 * 行います。一般的には、データベース接続の割り当てなどが該当します。
 */
public class EZGenericServletContext  extends EZServletContext
{
    private Connection con = null;
    /**
     *   現在このコンテキストに割り当てられているＤＢ接続オブジェクトを取得する。
     */
    public Connection getConnection () {return con;}
    /**
     *   指定されたＤＢ接続設定ファイルを使用して、ＤＢ接続をコンテキストに割り当てる。
     *   @param  propfname    ＤＢ接続プロパティファイル名.
     */
    public void assignConnection (String propfname)
    {
        if (con != null)
        {
            getLog().println ("ALERT ! DB Connection is already assigned !");
        }
        try {
            con = DBConnection.getConnection (propfname);
        } catch (Exception e) {
            throw new com.ngMAT.Common.LowLevelException (
                    "Failed to get connection with properties file : "
                    + propfname + " : " + e.getClass().getName() + " : "
                    + e.getMessage());
        }
    }
    /**
     *   このコンテキストに割り当てられているＤＢ接続を開放する。
     */
```

```
    public void releaseConnection ()
    {
        DBConnection.releaseConnection (con);
        con = null;
    }
    public void commit() throws java.sql.SQLException
    {
        con.commit();
    }
    public void rollback() throws java.sql.SQLException
    {
        con.rollback();
    }
    /**
     *   ログイン状態の確認。true : ログインしている。
     */
    public boolean  isLoggedIn()
    {
        return super.getSessionValue ("....") != null;
    }
    ....
}

public abstract class EZGenericServletConfig extends EZServletConfig
{
    public  String  DBConnectionName = null;
    public  String  NotReadyPageURL = "/";
    public  static com.ngMAT.Common.Log debug_log = null;
    ....
    public  final   void    init()
    {
        DBConnectionName = getProperty ("Common.Database.Connection");
        ....
        initialize();
    }
    /**
     * アプリケーション毎の設定読み込みを行う。
     *//
    public  abstract    void    initialize();
}

public  class   Sample1ServletConfig extends EZGenericServletConfig
{
    public  String  template_name = null;
    public  String  next_url = null;
    ....
    public void initialize()
    {
        template_name = getProperty ("Sample1Servlet.Template.Name");
        next_url = getProperty ("Sample1Servlet.NextURL");
        ....
    }
}

public  class   Sample1Servlet extends EZGenericServlet
{
    public boolean isReadyToGo (EZGenericServletContext context)
    {
        // 常に誰でも使用できるページであれば、固定的に"true"を返す。
        return true;
    }
    public HTMLComponent process (EZGenericServletContext context)
    {
        Sample1ServletConfig config = (Sample1ServletConfig)context.getConfig();
        HTMLTemplate template = context.getTemplate (config.template_name);
        // hiddenパラメタ"do_process"の有無で処理パターンを決定
        if (context.getParameter ("do_process") == null)
        {
            // 初期表示処理
        }
        else
        {
            // 入力されたデータの処理
            // エラーチェック
            if (isValidParameters (context))
            {
                // OKなら、結果をセッションに保持して次のページへGo!
                context.setSessionValue ("hogehoge", hogehoge);
                context.sendRedirect (config.next_url);
            }
            else
            {
                // 入力値が適切でない場合はエラーメッセージと共に入力値を再表示
                // → 実は初期表示とかなり似ている処理になることが多い
            }
        }
```

```
    }
}
```