Creating Simple Admin Tools Using Info*Engine and Java

Dennis G. Kapatos
Cory Skradski
J.D. Felkins (FELCO Solutions, Inc.)
Date: June 6
The Engineering Services Contract (ESC) at Kennedy Space Center provides services to NASA for the design and development of flight and ground systems in support of manned space flight. The ESC process support team provides for efficient optimized design and development processes through development, configuration, and implementation of software tools, training, documentation and standards. The team of 7 people supports over 200 engineers and design specialists using Windchill, Pro-E, NX, AutoCAD, and other design and analysis tools.
Agenda

- Common Business Needs
- The Info*Engine and JSP Solution
- Process Overview
- Creating Info* Engine Tasks
- Creating JSP Pages
- Resources and Help
Common Business Needs

- Get Information Out of Windchill:
  - Generate reports
  - Query Windchill for information and return it in a webpage
  - Exporting Windchill object information automatically to a third party application
  - Create a simple webpage for non-technical users to perform specific actions

- Create, Update, or Perform Actions on Many Objects At Once
  - Duplicating or renaming objects
  - Updating object contents, attributes, lifecycle states, etc.

- Perform Administrative Tasks:
  - Checkin or undo checkout all users' objects at once
  - Correct or update Windchill Object links
  - Changing teams, domain polices, etc....
The Info*Engine and JSP Solution

Capabilities of Info*Engine and JSP

- **Info*Engine**
  - Perform many actions available in standard Windchill UI
  - Perform some actions not available in standard Windchill UI

- **JSP Pages**
  - Provide a web-based UI for custom tools
The Info*Engine and JSP Solution

Uses for Info*Engine and JSP

- How often is it used?
  - One-time use
  - Frequently performed actions
  - Automated tasks seamlessly integrated into Windchill

- Who can use it?
  - Windchill administrators
  - Specific groups
  - All users
Demonstration

Demonstration of Info*Engine with JSP

- Renaming many objects
Process Overview

Steps to Implement Info*Engine and JSP

1. Write Info*Engine Task (XML)
2. Add XML files to Windchill codebase
3. Write JSP page (HTML and JavaScript)
4. Add JSP files to Windchill codebase
Creating Info*Engine Tasks

What is Info*Engine?

- Info*Engine - Provides a flexible, standard base foundation to automate specific tasks and transfers information to other third party applications. Info* Engine takes advantage of Service-Oriented Architecture (SOA) with the support for SOAP (Simple Object Access Protocol) and WSDL (Web Service Definition Language). Info*Engine tasks are in written in XML and do not require experience with Java.

- Info*Engine is the glue or underlying foundation of Windchill
Creating Info*Engine Tasks

Additional Information

- Info*Engine is installed in every Windchill installation and is free to use within the PTC products.

- No additional licenses are required unless using Info*Engine to connect with third party applications.

- PTC supports Info*Engine

- Does not require a skilled programmer or Java experience

- Easy to implement
Creating Info*Engine Tasks

Info*Engine Architecture

Info*Engine Server
- VDB - Virtual Database
- Task Processor
- Webject Processor

Adapter
- Windchill Adapter
- JDBC Adapter
- JNDI Adapter

- holds temporary results sets
- processes tasks
- processes webjects
- talks to Windchill
- talks to JDBC data source
  *Requires Info*Engine user licenses
- talks to any LDAP

(There are many more Adapters.)
Creating Info*Engine Tasks

Understanding Tasks and Webjects

- Info*Engine tasks are text-based xml files. They can control the retrieval and manipulation of data within the Info*Engine environment. Instead of developing a Java application or a JSP, a task can perform many of the same operations as Java applications can.

- A task includes Webjects to perform Windchill operations.

```xml
<!-- Query all containers in Winchill -->
<ie:webject name="Query-Objects" type="OBJ">
  <ie:param name="INSTANCE" data="com.ptc.ptc-training.Windchill"/>
  <ie:param name="authorization" data="$(@SERVER[AUTHORIZATION[0]])"/>
  <ie:param name="type" data="wt.inf.container.WTContainer"/>
  <ie:param name="where" data="()"/>
  <ie:param name="attribute" data="class,containerInfo.name" delim=",">
    <ie:param name="group_out" data="containers"/>
  </ie:param>
</ie:webject>
```
Creating Info*Engine Tasks

Understanding Tasks and Webjects

- The Info*Task compiler parses Info*Engine tasks and produces Java classes. This improves the performance of executing tasks by eliminating the need to parse and interpret a task each time it is called. It also facilitates embedding tasks in standalone Java applications and JSP pages.

- The task compiler produces the executable Java classes in three basic steps:
  1. Parses task sources and generates Java code that implements the task.
  2. Calls a Java compiler to produce an executable class from the generated Java source.
  3. Calls a class loader to load and instantiate the classes produced by the Java compiler.
Creating Info*Engine Tasks

Structure of a Webject

- Webjacts are the basic form to do most significant actions in Info*Engine. They are custom tag libs. Webjacts supported by the Windchill adapter accept parameters that specify database user credentials and query criteria.

```xml
<ie:webject name="Query-Objects">
    <ie:param name="INSTANCE"/>
    <ie:param name="DBUSER"/>
    <ie:param name="PASSWD"/>
    <ie:param name="type"/>
    <ie:param name="where"/>
    <ie:param name="attribute"/>
    <ie:param name="group_out"/>
</ie:webject>
```

- **A. Webject Name**
- **B. Webject Adapter Type**
- **C. Instance**
- **D. User Name and Password**
- **E. Type**
- **F. Where**
- **G. Attribute**
- **H. group_out**
Creating Info*Engine Tasks

Webject Adapters

- Webjects are the basic form to do most significant actions in Info*Engine. They are custom tag libs. Webjects supported by the Windchill adapter accept parameters that specify database user credentials and query criteria.

<table>
<thead>
<tr>
<th>Type</th>
<th>Webjects</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACT</td>
<td>Action</td>
<td>Performs actions on data</td>
</tr>
<tr>
<td>OBJ</td>
<td>Object or Query</td>
<td>Query the system</td>
</tr>
<tr>
<td>GRP</td>
<td>Group</td>
<td>Manipulate Virtual Data Base (VDB)</td>
</tr>
<tr>
<td>DSP</td>
<td>Display</td>
<td>Use data in VDB to display in HTML. This cannot be used in a standalone task but can be used in a JSP.</td>
</tr>
<tr>
<td>IMG</td>
<td>IMG</td>
<td>Use data in VDB to create a JPEG image</td>
</tr>
<tr>
<td>MGT</td>
<td>Management</td>
<td>Perform special activates such as throwing exceptions or getting properties</td>
</tr>
<tr>
<td>WES</td>
<td>Web Event Service</td>
<td>Subscribe to and manipulate messaging topics</td>
</tr>
<tr>
<td>MSG</td>
<td>Messaging</td>
<td>Subscribe to and manipulate message queues</td>
</tr>
<tr>
<td>ADM</td>
<td>Administrative</td>
<td>Perform I*E admin tasks, such as reloading cached properties</td>
</tr>
</tbody>
</table>
Creating Info*Engine Tasks

Running a Info*Engine Task

- The output of an Info*Engine task is in the form of XML. When running a task from the browser, the URL must include the **host name** and application URL prefix specified where I*E was installed. It also includes the /**servlet/IE/tasks** prefix, which directs the servlet to the task processor. And specifying the path where the xml task is located.

<table>
<thead>
<tr>
<th>Host Name</th>
<th>Prefix</th>
<th>Servlet Prefix</th>
<th>XML Path</th>
</tr>
</thead>
</table>

```xml
<?xml version="1.0" encoding="UTF-8"?>
<WC:COLLECTION xmlns:wc="http://www.ptc.com/infoengine/1.0">
  <wc:INSTANCE>
    <containerInfo.name>Site</containerInfo.name>
    <class>wt.inf.container.ExchangeContainer</class>
  </wc:INSTANCE>
  <wc:INSTANCE>
    <containerInfo.name>ptc</containerInfo.name>
    <class>wt.inf.container.OrgContainer</class>
  </wc:INSTANCE>
</WC:COLLECTION>
```
Demonstration

Demonstration of Info*Engine Task

- Update a objects attribute
- Modify Revisions of many CAD Document
- Updating Links between WTParts to CAD Documents.
Creating JSP Pages

What is a JSP?

- Java Server Pages (JSP) is a core technology of J2EE (the Java 2 Platform, Enterprise Edition) and solutions based upon EJB (Enterprise Java Beans).

- Info*Engine supports the development of enterprise custom Java applications and provides a JSP processor as an extension of the Info*Engine servlet engine. The JSP processor dynamically translates JSP pages into servlets.

- Usually, a JSP page is an HTML page with some additional JSP tags and some embedded Java code. However, inclusion of JSP tags or embedded Java is not mandatory, so a page containing only HTML is a legitimate JSP page.

- JSP pages that interact with Info*Engine usually contain a simple set of JSP tags and a set of custom Info*Engine tags that define the Webjects that are then executed when the page is accessed.
Creating JSP Pages

What is a JSP?

- JSP pages can include HTML, Java Classes, Java Scripts, Scriptlets, and Info*Engine code (Webjets).

- JSP pages are resided on a server

- A very simple example of a JSP page is shown below.

```html
HTML>
<BODY>
<%
   //This is a scriptlet.
   System.out.println("Evaluating date now");
   java.util.Date date = new java.util.Date();
%
Hello! The time is now <%= date %>
</BODY>
</HTML>
```
Creating JSP Pages

Location of a Windchill JSP

- When Info*Engine is installed, the installer specifies an Info*Engine installation directory which determines where JSP pages must be stored. All Info*Engine JSP pages must reside under the codebase directory where Info*Engine is installed.

- All JSP pages are saved on the Windchill server at the below location.

- `<Windchill>\codebase\infoengine\jsp\`
Creating JSP Pages

Executing a Windchill JSP

- The URL to execute a JSP page includes the host name and Windchill application URL with "infoengine/jsp/" and the path to the JSP page.

Example of executing a JSP page URL is below.

http://train.ptc.com/Windchill/infoengine/jsp/examples/My_Simple.jsp

When the file is executed, the Web Server passes the URL to the JSP processor.
Creating JSP Pages

Creating a Simple JSP

- There are many IDEs that help develop JSP pages. Some examples are Coffer Cup, Eclipse, and NetBeans or a developer could use a good text editor like Notepad++.

- To the right is a simple JSP page with two Webjects.
Demonstration

Demonstration of Running a JSP

- Running the My_Simple.jsp
Creating JSP Pages

Using HTML and Webjects in a JSP

- The previous demonstration displays a table output of the Webjects results, but the table display is not what you want. The below images display a HTML table to control the display.

```html
<html>
<head>
<title>JSP Display</title>
</head>
<body>
<table border="1" cellpadding="2">
<tr>
<td><input type="checkbox" name="allcheckbox"></td>
<td>Name</td>
<td>Number</td>
<td>Version</td>
<td>Status</td>
<td>State</td>
<td>Container</td>
<td>Type</td>
</tr>
<tr>
<td><input value="" type="checkbox"></td>
<td><i:getValue name="name"></td>
<td><i:getValue name="number"></td>
<td><i:getValue name="version"></td>
<td><i:getValue name="checkoutInfo.state"></td>
<td><i:getValue name="lifeCycleState"></td>
<td><i:getValue name="containerName"></td>
<td><i:getValue name="type"></td>
</tr>
</td>
</tr>
</table>
</body>
</html>
```

Page Results

JSP File Webjects are located at the top of the page
Demonstration

Demonstration of HTML and Info*Engine in a JSP

- Displaying information using HTML elements
- Reviewing a complex JSP page
- Showing PTC JSP examples
Resources and Help

- Java Adapter Development Kit: IEJADKGuide.pdf
- Felco Solutions Website: http://www.felcosolutions.com