

Developing a Complete VI Client Plug-in with Tomcat and VI Java API*

Steve Jin, VMware Engineering

VMware VI Client plug-ins allows you to integrate your application directly into the standard VMware VI Client. The implementation centers around the XML and Web technology. Everything that shows up in a tab or a pop up windows is feed from a Web server, either a well known Web application or a specialized Web application. More technical information about this can be found in the [Getting Started with VI Client Plug-ins tech-note](#).

The most time consuming part of the development process is not in configuration, or the registration. It is the back end Web application that connects to the VC Server or ESX server, and retrieves related information and presents it to the client side plug-in component.

Since Web technology is language agnostic, you can use any languages/framework to develop your Web application. When you choose the language/framework, you should consider these two technical aspects:

1. Is the language/framework Web friendly? A good Web framework provides all the needed utilities for you to parse URL, generate HTML content, handle cookie etc.
2. Is there a good API or toolkit in that language for you to connect back to the VC Server? Currently VMware has offered VI Perl Toolkit, and VMware Toolkit (for Windows) is coming soon. All these two provide you easy toolkits to connect to the VC Server and ESX server. The open source VI Java API provides a high level abstraction over the existing Web service interfaces, and makes Java much easier to talk to VI.

Besides these two technical aspects you will also need to consider your team's language proficiency. Most of the time all the languages can pretty much achieve same results. It makes a big difference in productivity when you are familiar with one programming language. We are not going to get too much in this discussion. In the end you have to make a call which to use.

In the following several pages, I am going to show you how to develop a complete VI Client plug-in in Java that can interact with the virtual infrastructure. I assume you have read through the Getting Started with VI Client Plug-ins tech-note and understand the basics of VI plug-in development process. I also assume you are familiar with Java and Java Servlet development.

Note: The VI Java API is an open source project hosted on sourceforge.net. It is not a VMware product, nor is it supported by VMware. For more information, and for community-based support, visit the VI Java API project home at <http://vijava.sf.net>

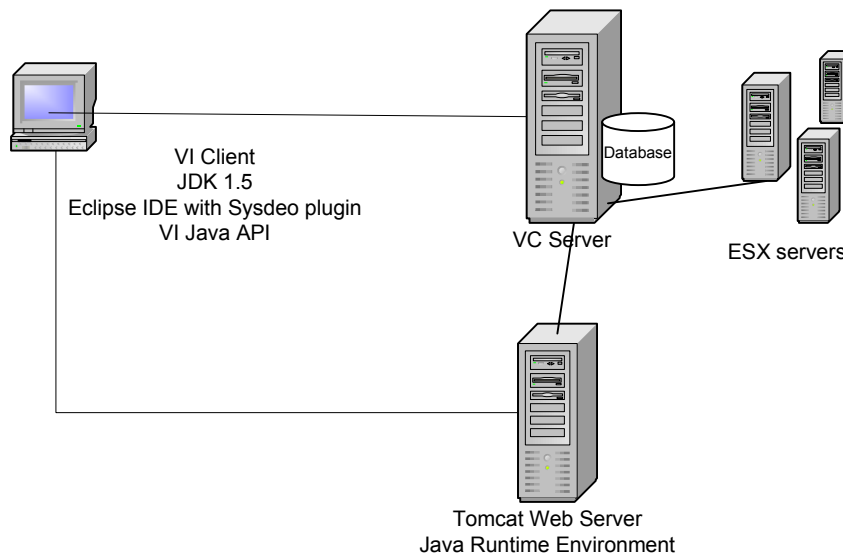
Set up the environment

Before getting started with the VI Client plug-in development, let us first set up the development environment, both the physical environment and the software environment. We will cover these two in a two parts.

Physical environment

You will need a system environment shown as follows. The VC Server, Tomcat Web server and the development environment can reside on separate physical machines, a single physical machine, or two physical machines with any combination of these major software components. Please note that even you have the VC Server and Tomcat Web server on the same machine, we recommend you to install your own Tomcat Web server even though VC Server comes with a Tomcat Web server. Hacking the Tomcat coming with VC Server could cause your VC Server reject any VI Client connection.

In the following discussion, all the major software components are installed on one physical machine. The physical topology doesn't affect the development process, but it affects your deployment process in that you need to make sure right component in right place and the configurations reflects your environment.



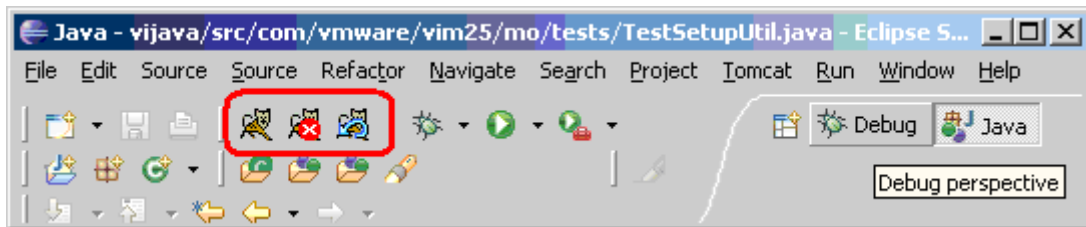
Software environment

Here is a list of components you will need:

1. VMware Virtual Infrastructure
You will have one VC Server and at least one ESX server installed and configured. Please check VMware Web site for more instructions.
2. Java SDK 1.5 or above.
You can download from <http://java.sun.com/javase/downloads/index.jsp>

Note: The VI Java API is an open source project hosted on sourceforge.net. It is not a VMware product, nor is it supported by VMware. For more information, and for community-based support, visit the VI Java API project home at <http://vijava.sf.net>

3. VI SDK 2.5. You can download it from <http://www.vmware.com/download/sdk/sdk.html>
4. Eclipse IDE
The Eclipse IDE is one of the most popular Java IDEs today. You can download a copy from Eclipse Web site (<http://www.eclipse.org>) for free. We use version 3.3.2 in the following discussion.
5. Tomcat Web server
Tomcat is one of the most popular Java servlet container plus Web server. You can download it from Apache Web site (<http://tomcat.apache.org/>). We use Tomcat version 6.0.16.
6. Sysdeo plug-in
This is an Eclipse plug-in that helps you to develop Java Web application with Tomcat. (<http://www.eclipse-totale.com/tomcatPlugin.html>). We use version 3.2.1.
When you successfully install and configure it, you should be able to see the three Tomcat icons representing start/stop/restart Tomcat server as follows.



7. VI Java API
This is an open source VI Java API hosted on sourceforge.net. You can download the latest release at <http://vijava.sf.net>. You don't need to install the jar file – it will be included into the classpath of the coming Java project as described later.

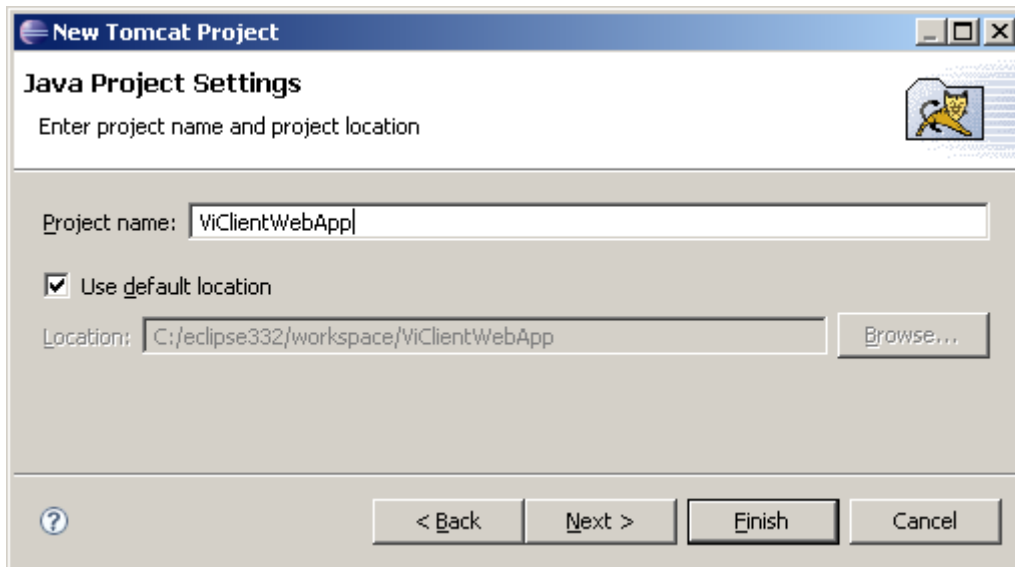
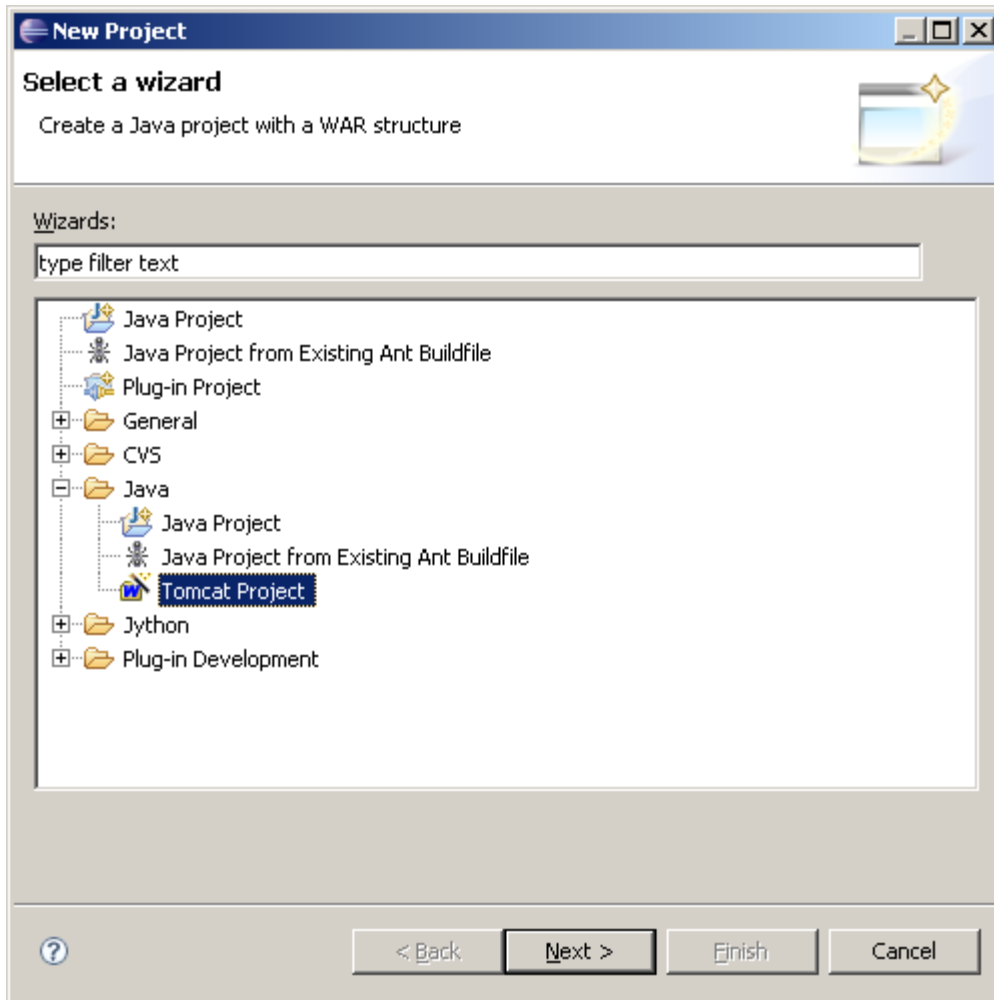
Get started with your first project

Now that you have set up the environment successfully, let us get it started. I will show you step by step how to create your first project.

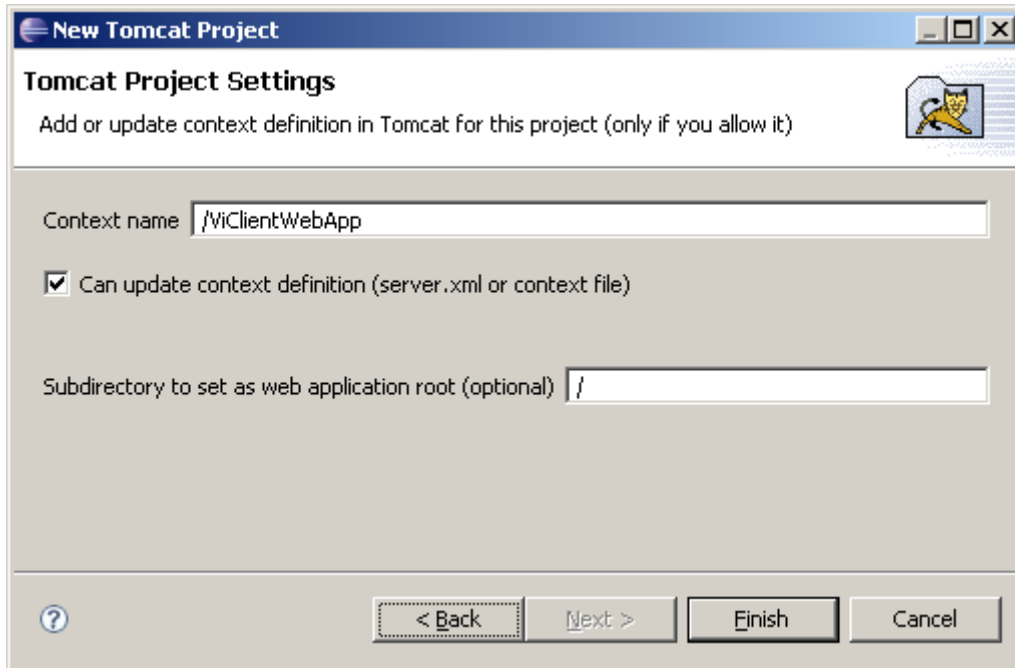
Step1: Start the Eclipse and create a Tomcat project via the New wizard.

Please select the project type as “Tomcat Project” as show in the following:

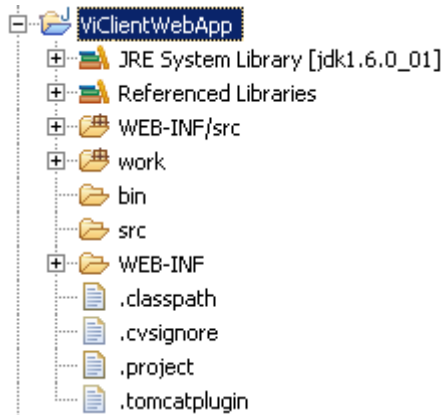
Note: The VI Java API is an open source project hosted on sourceforge.net. It is not a VMware product, nor is it supported by VMware. For more information, and for community-based support, visit the VI Java API project home at <http://vijava.sf.net>



Note: The VI Java API is an open source project hosted on sourceforge.net. It is not a VMware product, nor is it supported by VMware. For more information, and for community-based support, visit the VI Java API project home at <http://vijava.sf.net>



When you are done, you should see the ViClientWebApp show up in the package explorer with the structure like the following:



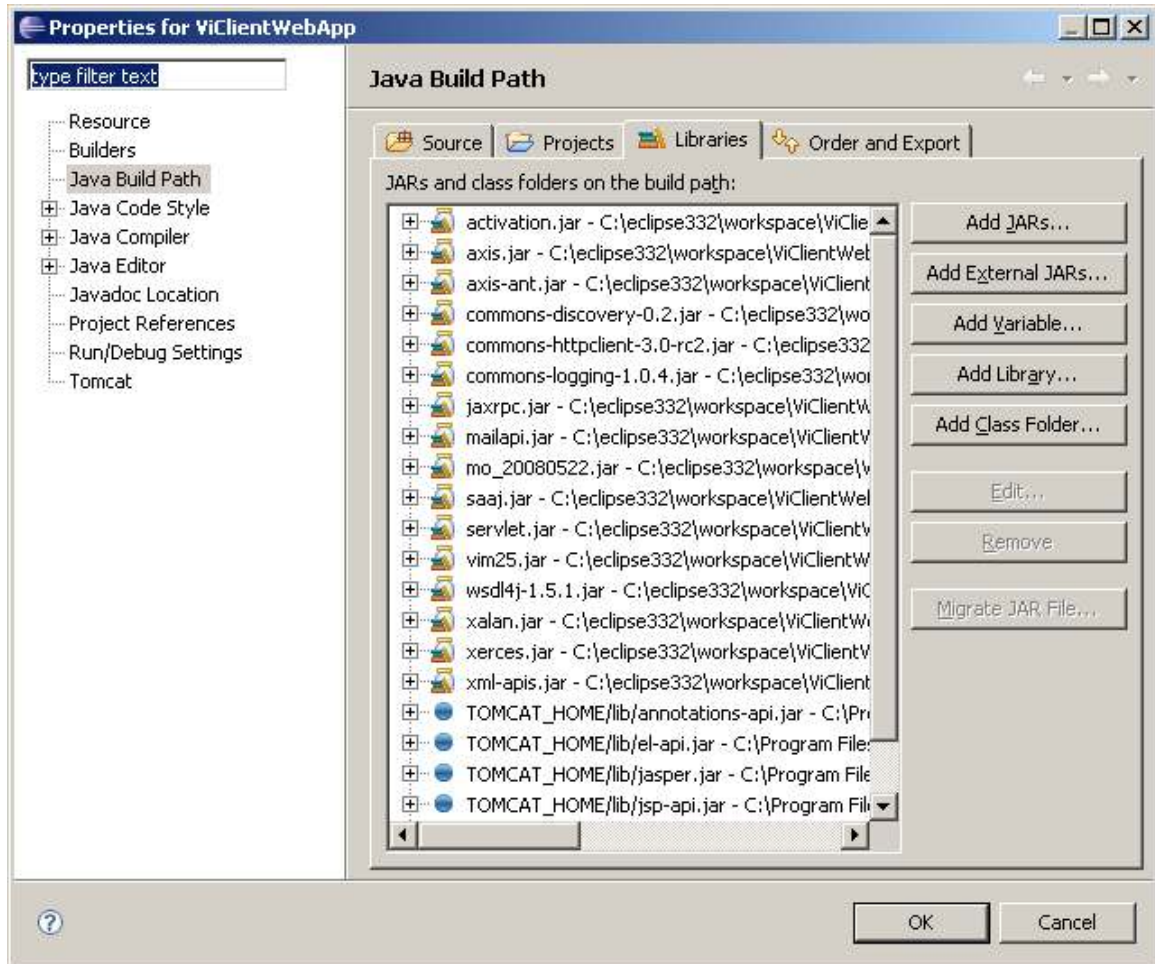
Step 2: Configure the project

First, you will need to copy all the needed JARs into the ViClientWebApp/WEB-INF/lib directory. These Jars include those needed by a typical VI SDK 2.5 project and the VI Java API jar file as mentioned #7 last section. Please refer to the [Developer's Setup Guide](#) for how to set up a typical VI SDK 2.5 project.

In the package explorer, right click the project and select **Properties** from the context menu. In the dialog box, select **Java Build Path** from left side tree, and choose **Libraries** tab. Click on

Note: The VI Java API is an open source project hosted on sourceforge.net. It is not a VMware product, nor is it supported by VMware. For more information, and for community-based support, visit the VI Java API project home at <http://vijava.sf.net>

Add External JARs... button on the right side pane, a **JAR selection** dialog box shows up. Just navigate to the WEB-INFO/lib directory and include all the jar files there.



Step 3: Write your Java code & Web.xml

Under WEB-INF/src, create a java class TestServlet.java as follows. This sample will print out the name of the selected managed entity in the inventory tree, plus a time stamp.

```
import java.io.IOException;
import java.io.PrintWriter;
import java.net.URL;
import java.text.DateFormat;
import java.util.Date;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
```

Note: The VI Java API is an open source project hosted on sourceforge.net. It is not a VMware product, nor is it supported by VMware. For more information, and for community-based support, visit the VI Java API project home at <http://vijava.sf.net>

```

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import com.vmware.vim25.ManagedObjectReference;
import com.vmware.vim25.mo.ManagedEntity;
import com.vmware.vim25.mo.ServiceInstance;
import com.vmware.vim25.mo.util.MorUtil;

public class TestServlet extends HttpServlet
{

    public final static String MOREF = "moref";
    public final static String SESSION_ID = "sessionId";
    public final static String SERVICE_URL = "serviceUrl";
    public final static String LOCALE = "locale";

    protected void service(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException
    {

        String morStr = request.getParameter(MOREF);
        String type = morStr.substring(0, morStr.indexOf(":"));
        String value = morStr.substring(morStr.indexOf(":")+1);

        ManagedObjectReference mor = new ManagedObjectReference();
        mor.setType(type);
        mor.setValue(value);

        String sessionStr = "vmware_soap_session=\"" +
request.getParameter(SESSION_ID) + "\"";

        System.out.println("morStr:" + morStr);
        System.out.println("serviceUrl" +
request.getParameter(SERVICE_URL) );
        System.out.println("session:" + sessionStr);

        ServiceInstance si = new ServiceInstance(new
URL(request.getParameter(SERVICE_URL)), sessionStr, true);

        ManagedEntity me =
MorUtil.createExactManagedEntity(si.getServerConnection(), mor);

        String name = me.getName();

        PrintWriter out = response.getWriter();
        out.println("name:" + name);
        out.println(DateFormat.getDateTimeInstance().format(new
Date()));
    }
}

```

Note: The VI Java API is an open source project hosted on sourceforge.net. It is not a VMware product, nor is it supported by VMware. For more information, and for community-based support, visit the VI Java API project home at <http://vijava.sf.net>

The first part of the code parses the URL for parameter values, and creates a ManagedObjectReference object based on the moref value. Then ServiceInstance object is created, and so is the real managed entity type. You can call either accessors or typical operations defined on the managed object type. In this sample, we just call the getName() to get the value of property "name" and prints it out, followed by a timestamp.

The VI Java API has encapsulated the tedious code of using PropertyCollector to fetch a property, therefore we can use one simple line to achieve otherwise impossible with VI SDK web services interfaces. The API offers much more than accessor methods, but we are not going to discuss it any further here.

After this part is done, create a Web.xml file under WEB-INF directory as following:

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<Web-app xmlns="http://java.sun.com/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/Web-app_2_5.xsd"
  version="2.5">

  <display-name>VI Client Plugin Demo App</display-name>
  <description> VI Client Plugin Demo App with Tomcat and VI Java
API</description>]

  <servlet>
    <servlet-name>TestServlet</servlet-name>
    <servlet-class>TestServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>TestServlet</servlet-name>
    <url-pattern>/TestServlet</url-pattern>
  </servlet-mapping>

</Web-app>
```

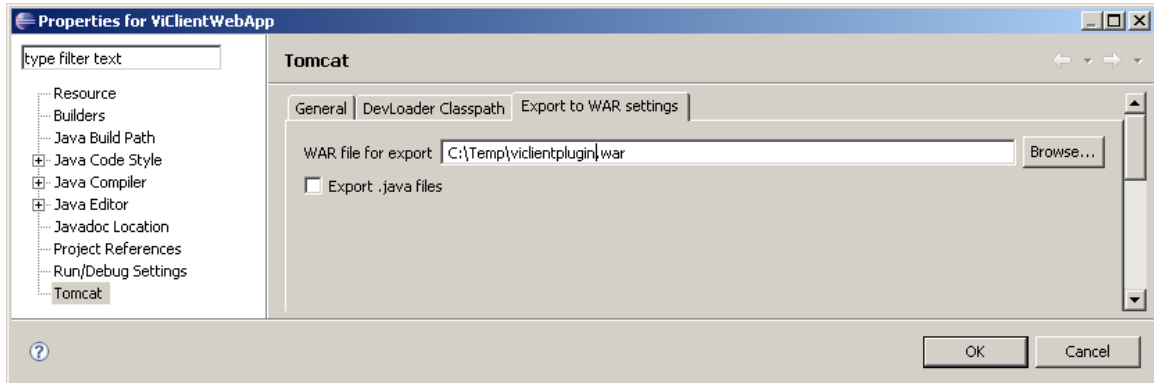
This Web.xml is mainly used for mapping the url pattern to servlet name and further to the servlet class. In this setting, I should be able to access this servlet using the following URL:

<http://10.17.216.162:8080/ViClientWebApp/TestServlet>

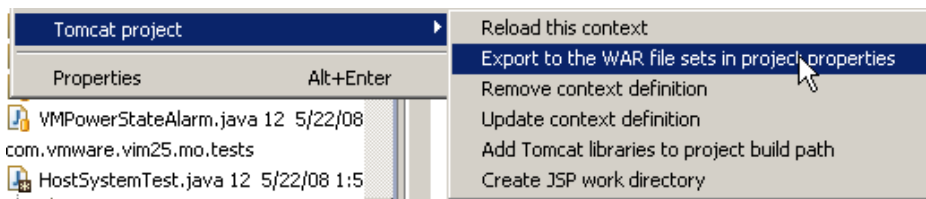
You will need this URL in your VI Client configuration file covered in next step. When you work in your test environment, you should replace the IP address (10.17.216.162) with your Web server's IP address. If you want to have a different URL pattern to access the Java servlet, you can simply change the Web.xml file.

Note: The VI Java API is an open source project hosted on sourceforge.net. It is not a VMware product, nor is it supported by VMware. For more information, and for community-based support, visit the VI Java API project home at <http://vijava.sf.net>

Optionally, if you want to deploy the Web application to another application server, you can create a war package. In the project **properties** dialog box shown as the following, you should point the **WAR file export location** to the directory where you application server save .war files, or save it local and then copy it over to its final destination.



When you are done with the setting, you can right click at the project in the package explorer and get the following context menu to **“Export to the WAR file sets in project properties”** to create the .war file.



Step 4: Modify your configuration file and register it with VC Server

The following is a sample configuration file. The highlighted part defines extension points on both HostSystem and VirtualMachine, both of type ManagedEntity, therefore a new tab will show up in the right side along with other tabs whenever a virtual machine or a HostSystem is selected in the inventory tree on left side. Again, you should replace the IP address (10.17.216.162) to your Web server’s IP address.

```
<?xml version="1.0" ?>

<scriptConfiguration version="1.0.0" processingMode="0">
  <key>com.vmware.cde.sdk.demo</key>
  <description>VI Plguin Demo </description>

  <view parent="Inventory.Global">
    <title locale="en">VMware Communities</title>
    <url>http://vmware.com/communities/content/</url>
  </view>
</scriptConfiguration>
```

Note: The VI Java API is an open source project hosted on sourceforge.net. It is not a VMware product, nor is it supported by VMware. For more information, and for community-based support, visit the VI Java API project home at <http://vijava.sf.net>

```

</view>

<view parent="Inventory.VirtualMachine">
  <title locale="en">Tester</title>
  <url>http://10.17.216.162:8080/ViClientWebApp/TestServlet</url>
</view>

<view parent="Inventory.HostSystem">
  <title locale="en">Tester</title>
  <url>http://10.17.216.162:8080/ViClientWebApp/TestServlet</url>
</view>

<view parent="Inventory.Datacenter">
  <title locale="en">MapIt</title>

<url><![CDATA[http://maps.google.com/maps?f=q&hl=en&geocode=&time=&date
=&ttype=&q=3210+Porter,+Palo+Alto,+CA]]></url>
</view>

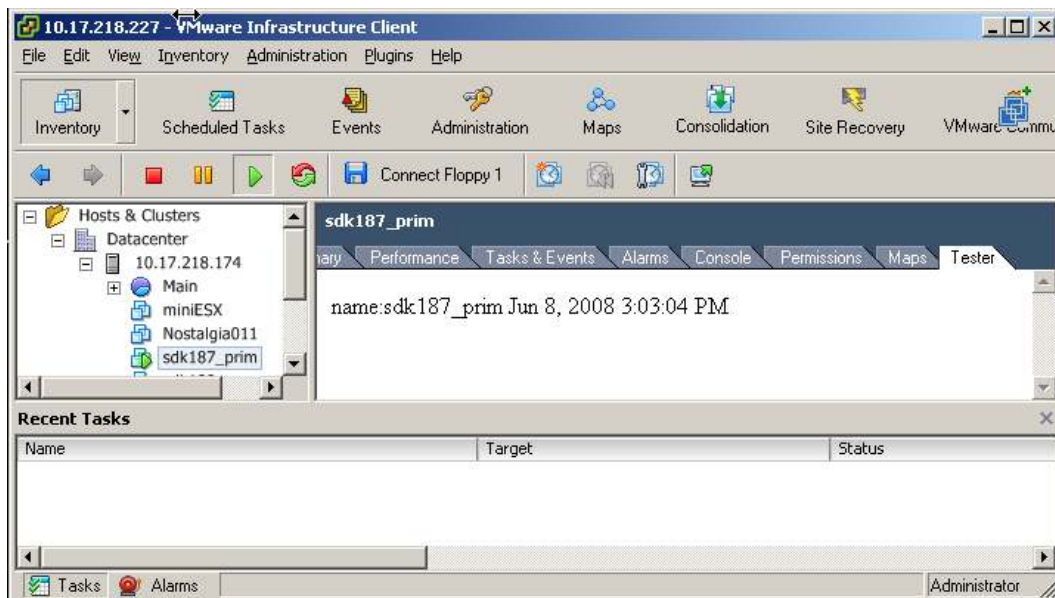
</scriptConfiguration>

```

You can use the registration utilities from the [VI Client plug-in developer community](#). You can choose either Java, Perl or Powershell version of the registration utility.

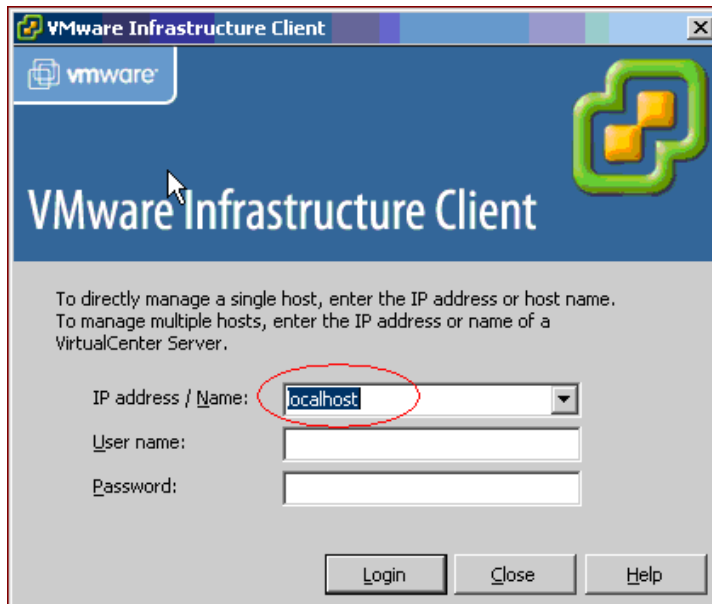
Step 5: Run the VI Client and verify the VI plug-in.

The following is the screen of the plug-in displaying the name of a virtual machine and a time stamp of the Web server.



Note: The VI Java API is an open source project hosted on sourceforge.net. It is not a VMware product, nor is it supported by VMware. For more information, and for community-based support, visit the VI Java API project home at <http://vijava.sf.net>

Please note: When you login the VI Client connecting to a VC Server, please make sure you do **NOT** use the default IP address / Name “localhost.” Whatever you enter or choose here is going to be passed on to the Web application. Unless the Web server is located on the same VC Server, the Web server has no way to resolve the real address when it gets “localhost” in the URL string from the VI Client.



Discussion

The above sample shows how to get different components together to create a VI Client plug-in that talks to VC Server in a useful way – retrieve information from the VC Server side. Although it is very simple, it has all the pieces for you to build a commercial VI plug-in in a production environment.

Some of these components are optional for example Eclipse, Sysdeo plugin, but they really help to streamline the development process, and make it easier to debug Java servlets.

Another optional component is the VI Java API. It's built on top of the existing VI SDK Web service interfaces, but provides a boost of productivity. The code using the API is much shorter, and more importantly much more readable.

With the support of VI Java API, you can not only retrieve information but also easily control the virtual infrastructure from the VI Client plug-in itself, for example power on a virtual machine. For more information on how to achieve these tasks, you can download the VI Java API and take a look at the sample code there.

Note: The VI Java API is an open source project hosted on sourceforge.net. It is not a VMware product, nor is it supported by VMware. For more information, and for community-based support, visit the VI Java API project home at <http://vijava.sf.net>