

Introduction to Servoy Headless Client (SHC)

In Servoy 2.2, we've introduced a new way to access database data and Servoy solution business logic: the 'Servoy Headless client'.

The Servoy Headless Client (SHC) enables you to develop web browser applications that access any data hosted by a Servoy Server and also execute any method in a solution hosted on that particular server. SHC uses the built-in Apache Tomcat Servlet runner to communicate with Servoy Application Server. You can access SHC through Java Applications, Java Servlets and Java Server Pages (JSP).

Before you begin

To use SHC, you must have some background in programming web applications. Experience with JSP, ASP, PHP, Cold Fusion or Lasso is highly recommended. If you don't have any experience with these programming environments, we recommend that you first get a book on JSP (there are many good books available in bookstores) or wait until we release Servoy Instant Publishing for Web Browsers.

How does SHC work?

-A headless client is started inside the Servoy application server (overhead is about 2MB per client being started) for each JSP web session.

-An interface ISessionBean is provided to access form data and call client methods.

-Each headless client consumes a normal Servoy Client license when started - and that client is 'released' (becomes available again) after the JSP session has expired (or has been programatically terminated).

In this introduction to SHC, we'll first discuss the included examples (and installation), then we'll have a look at the available functions.

Installing SHC

Prerequisite 1:

Make sure you have the latest Servoy 2.2 version installed (2.2 beta 2 or higher).

Prerequisite 2 (for Windows users only):

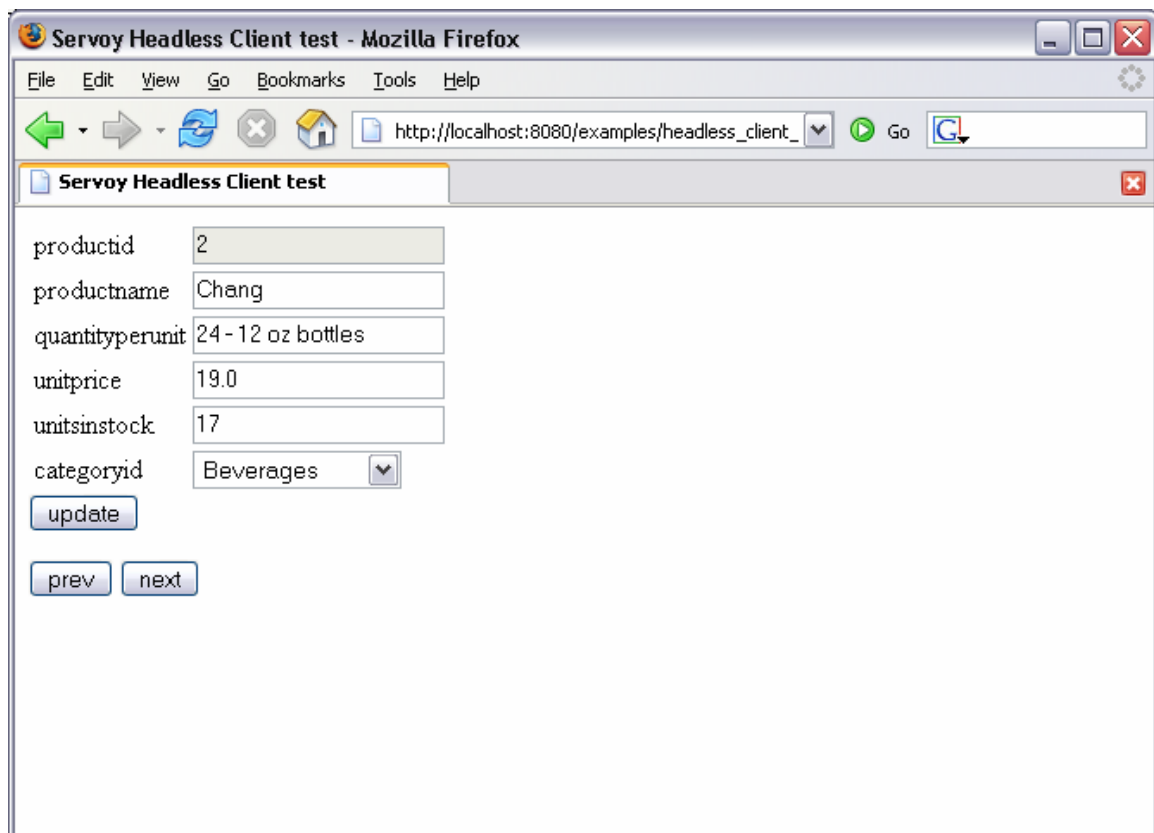
Make sure you have tools.jar (a Java library) in your Servoy\server\lib folder. The tools.jar file is located in the lib directory of your JDK install (usually in Program Files\Java). If you don't have tools.jar - it is NOT INSTALLED by default - copy the file to the Servoy lib folder.

1. Import the example solution 'headless_client_demo.servoy' located in the solution/example folder/directory. (if this example is not there, you extracted the files incorrectly in step 1). In Servoy Developer, choose File>Repository and then press the 'Import' button.

2. Open the solution in Servoy Developer and point a web browser to:

http://localhost:8080/examples/headless_client_formtest.jsp

Let's have a look at the source of this page so we can see how SHC works.



Using SHC: the examples explained

In the top section of your Java Server Page, you declare your imports - meaning the libraries you want to be available throughout your page:

```
<%@ page import = "java.util.*" %>
<%@ page import = "com.servoy.j2db.server.headlessclient.*" %>
<%@ page import = "com.servoy.j2db.dataprocessing.IDataSet" %>
```

Optionally, you can also specify an error page - the page that will be displayed if anything goes wrong:

```
<%@ page errorPage="errorpage.jsp" %>
```

To 'bind' the web browser window to a headless client (the browser now has become the head of the headless client; the part of the client that performs all the logic), this code is used:

```
ISessionBean servoy_hc = (ISessionBean)session.getAttribute("servoy");
if (servoy_hc == null)
{
    servoy_hc =
HeadlessClientFactory.createSessionBean(request, "headless_client_demo")
;
    session.setAttribute("servoy", servoy_hc);
}
```

Next we set the main form we want to use in this page:

```
boolean ok = servoy_hc.setMainForm("products");
if (!ok)
{
    out.print("error cannot work on required form");
    return;
}
```

In the next lines of code, we check the parameter 'browse'. If the parameter 'browse' contains either 'next' or 'previous', we execute the corresponding solution method via the SHC.

The rest of the code is the HTML code of the page. You can edit JSP pages with your favorite HTML editor (for example Dreamweaver) or plain text editor (for example vi). Let's have a look at the most important code snippets:

```
<%=servoy_hc.getDataProviderValue(null,"productid")%>
```

The line above retrieves the value of 'productid' in the currently selected record.

The `getDataProviderValue` function on the `ISessionBean` object retrieves the value of a dataprovider. The first parameter defines the `contextName` and the second parameter is the name of the dataprovider you want to get. If you are retrieving a dataprovider from the current form, you can leave the `contextName` null. The `contextName` can be any visible form name (for example forms in tab-less tabpanels).

The following snippet shows you how to fill a valuelist in **HTML**:

```
IDataset categories = servoy_hc.getValueListItems(null,"categories");
for (int i = 0 ; i < categories.getRowCount() ; i++)
{
    Object[] item_row = categories.getRow(i);
    String selected = "";
    if (item_row[1] != null &&
item_row[1].equals(servoy_hc.getDataProviderValue(null,"categoryid")))
    {
        selected = "SELECTED";
    }
    %>
    <option value="<%=item_row[1]%>"
<%=selected%>><%=item_row[0]%></option>
    <%
}
%>
```

On the first line, we retrieve the valuelist into an IDataset object. In the for loop, we loop through this dataset using the `getRowCount()` function on the IDataset object.

Updating data:

As you can see in the sample page, the dataset posts to the page named `headless_client_formresult.jsp`. In that page, you will find that all the updates are done with a single line of code!:

```
int setcount = servoy_hc.setDataProviderValues(null,request);
```

The line above will update all values on the form assuming that the **HTML** field name matches with the `dataprovidername` of the dataprovider on the Servoy side! Now that is some powerful code! To save the data to the database always put in a `saveData` call as well:

```
servoy_hc.saveData();
```

You can also set dataproviders based on their name using:

```
setDataProviderValue(String contextName,
                       String dataprovider,
                       Object value)
```

Eg: `setDataProviderValue(null, "customername","Servoy");`

Other important functions of the headless client are:

- `setLocale(locale)` to change the default locale
- `getI18NMessage(key,options)` to retrieve localized messages
- `executeMethod(visibleFormName, methodName, arguments)` to execute any Servoy method.

Examples

```
//to set the language to English and country to USA:  
setLocale(new Locale("en", "us")) ;
```

```
//to retrieve the i18n message for the key submitbuttonOKText:  
getI18NMessage("submitbuttonOKText", null);
```

```
//to execute a Method on the form customers named newCustomer:  
executeMethod("customers", "newCustomer", null);
```

IMPORTANT NOTE: *Make sure that you don't execute a method that displays a dialog - otherwise, the client will hang as it will be waiting for the dialog to be closed.*

A full listing of the SHC API and it's functions is available here:

<http://developer.servoy.com/docs/headless-client-api/index.html>

Notes:

- You can find the jsp sources in the <Servoy install dir>\server\webapps\ROOT\examples folder/directory.

- NEVER show modal (form) dialogs or modal splash windows in Servoy methods used by SHC - this will make the client hang.

- Always start the application server with the option -Djava.awt.headless=true as seen in servoy_server.bat (.sh on Mac/Unix/Linux).

-SHC works only with JDK installed not with JRE (needs a java compiler for jsp pages). JDK is installed by default on the Mac.

-The error page shows the error details in white-colored text on white background. To make the text visible, press CTRL-A/CMD-A to select all the text.

-It is possible to let a solution know it is used by a headless client by passing an argument array in HeadlessClientFactory to the startup method.

Example:

```
servoy_hc =  
HeadlessClientFactory.createSessionBean(request, "headless_client_demo",  
"username", "password", new Object[]{"Headless", "args2"})  
//second argument is the name of the solution  
//third and fourth username and password to login  
//fifth the arguments you are passing
```

-The SHC is stopped when the session times out (as configured in tomcat) or by explicitly stopping the session using:

```
session.setAttribute("servoy", null);
```

Further discussion

Visit the new headless client forum to discuss development and it's features!

<http://forum.servoy.com/viewforum.php?f=25>

Conclusion

Servoy Headless Client gives you many options to access Servoy data and methods from other devices and applications. You can build HTML applications that link to Servoy using industry standard JSP. SHC also allows you to develop applications for PDAs and cellphones in a very easy and straightforward way.