Lab 3: .NET 3.5 Graphical Application Client for *secure* caTissue caGrid Service

caBIG 2009 Annual Meeting Hack-a-thon

University of Virginia eScience Group
Marty Humphrey, Director

Overview: Create a graphical .NET application as a client for the *secure* caTissue caGrid Service. We use a ".NET Dorian Login" client (provided) to first obtain the necessary credentials. As with Lab 1 and Lab 2, we use the built-in features of Visual Studio to create the client-side code, including the CQL query. A key feature of the client-side code is a graphical "certificate picker". In this lab, we'll use the Windows Presentation Foundation (WPF) to create the visual representation (Microsoft recommends using Expression Blend to create WPF and Silverlight applications, but for this simple application we'll just use Visual Studio).

NOTE: This lab assumes that you already have a caGrid account

Expected duration: 30 minutes

Preliminary 1: Download and install the Certificate Authority (CA) certificate

1. Open up a browser (these instructions assume Internet Explorer, but other browsers will work) and go to the following URL (save the file locally): http://www.cs.virginia.edu/~humphrey/caBIG_2009_AnnualMeeting/catraininggridca.cer
2. After you save the file locally, double-click on it to open it
3. Select "Install Certificate", and follow the default options to install the certificate

Preliminary 2: Download .NET Dorian Login Client

4. Download the following file that contains the .NET Dorian Login client (put this file wherever you want on your machine): http://www.cs.virginia.edu/~humphrey/caBIG_2009_AnnualMeeting/DorianClient.zip
5. Unzip the file
6. Open the "Debug" folder
7. Double-click "DorianLogin.exe" to launch the login client (if you're given a security prompt, asking you if you're sure you want to run this, select "Run")
8. Change the "Dorian Service" and "Authentication Service" as necessary, enter the username and password and then click the "Login" button. *Wait a few seconds, and if successful you will see "Login Successful!" at the bottom of the window. If you see any other behavior, ask for help.*

Preliminary 3: Confirm that you have successfully acquired a caGrid certificate

9. Start → run , type in "mmc" and hit "OK"
10. File → Add/Remove snap-in
11. Add, click on "Certificates", click "Add", the default of "My user account" is correct, and then hit "finish"

12. Hit “OK” to close the “Add/remove snap in” window
13. On the left side of the window, expand “Certificates – Current User”, then expand “Personal”, and then click on “Certificates”
14. In the right side of the window, double-click on your certificate. Explore the tabs as you wish, and then select “OK”
15. You can now close this window (without saving the “console settings”)

Part 1: Simple GUI for counting items in a secure caTissue Service

16. Run Visual Studio 2008 (Start → All Programs → Microsoft Visual C# 2008 Express Edition) and create a new C# project: WPF Application (File → New Project →WPF Application). Name it “caTissueClient” and hit “OK” in this dialogue window.
17. Our first task is to add some user controls to the empty window “Window1” and name them. The components can be dragged and dropped from the Toolbox. The Toolbox is located on the left part of the screen (if it is not visible, click on View → ToolBox). (All of the following are under the “Common” part of the toolbox. If Window1 is not shown in Visual Studio, double-click on Window1.xaml in the Solution Explorer.)
    a. Upper left of Window1: Label (give it the content “caTissue Service”)
    b. Immediately to the right the label: Textbox (rename it “caTissueServiceBox” and change the “Text” to https://128.252.227.214:58443/wsrf/services/cagrid/CaTissueCore, which is the WashU caTissue Core v 1.2.2, but any caTissue should work). It also makes sense to make “Window1” larger horizontally to fit everything.
    c. Immediately below the first label: Label (give it the content “Count the items of:”)
    d. To the right of this second label: ListBox
    e. Centered below this: button (give it the content “Click here to start”). Stretch the button horizontally to show all of the text.
    f. Centered below this: textbox (rename it “OutputBox”). Make this box fairly big, both horizontally and vertically. Make the entire “Window1” larger as well. Change the “HorizontalScrollBarVisibility” and “VerticalScrollBarVisibility” of this textbox to be “auto”.
    g. Close the “toolbox” pane.
18. Click on “Window1” and change the “title” to be “caTissue Client”. Also, if you want to, change the “background” of Window1 to be your favorite color.
19. Now let’s add a few caTissue items from which the user can select. *This is probably not the way to do this if you were building a real application, but this works nicely for this tutorial.* Click on ListBox1, scroll to the “Items” in the properties pane, and click on the “…” box, which will bring up the “Collection Editor: Items” window. Hit the “Add” and then change the “Content” for each of the following (don’t hi t “OK” at the bottom of the window until you’re all done with all 4 items)
    a. edu.wustl.catissuecore.domain.MolecularSpecimen
    b. edu.wustl.catissuecore.domain.TissueSpecimen
    c. edu.wustl.catissuecore.domain.FluidSpecimen
    d. edu.wustl.catissuecore.domain.CellSpecimen
20. Resize ListBox1 to only show these 4 items

Part 2: Adding the caGrid Service reference

21. Our second task is to generate the client-side proxy objects and configure the service endpoint. This is essentially what we did in Lab 1, so we have made this available so you don’t have to do it again.

Open up a browser to http://www.cs.virginia.edu/~humphrey/caBIG_2009_AnnualMeeting/ and right-click caTissueSvc.cs and select "Save File As.." and store the file locally (it doesn't matter where). Note that we don't need an "app.config", as in Lab 1 and Lab 2, because we'll configure the service endpoint in code. Now:

a. Right-click on caTissueClient project in the Project Explorer -- it's the line immediately below the "Solution 'caTissueClient' (1 solution)", select "Add → Existing Item" and navigate to where you put caTissueSvc.cs
b. We now need to add the appropriate libraries to the solution (we didn't have to do this in Lab 1 and Lab 2 because we did "Add Service Reference", which adds them automatically):
    i. Right-click on caTissueClient project in the Project Explorer, "Add Reference" and select "System.Runtime.Serialization"
    ii. Do the same for "System.ServiceModel"
    iii. Do the same for "System.security"
c. Just to make sure everything is okay, compile everything by selecting "Build → Rebuild Solution" (don't run the application yet). The "Error list" at the bottom of Visual Studio should say "0 errors" and "Rebuild All succeeded". Note: after this step, Visual Studio *might* put up a small bar on the top of "window1.xaml" stating "An assembly or related document has been updated which requires the designer to be reloaded. Click here to reload". You should click on this bar here and whenever you see this during the lab.

Part 3: Adding client code to the GUI

22. Our third task is to add the necessary code to generate the CQL query based on the user input (the selection in the GUI), get the response from the server and display the results on the GUI. Note that we have designed the GUI to present more debugging-type information – if this were a real application, we would probably replace the textbox with a much-smaller box to display just the count returned. *Begin this part by double-clicking the button you created in Visual Studio* ("Click here to start").This will show you the code that will execute when the user hits this button.
23. Let's add all of the "using" statements all at once, by scrolling to the top of the file and adding:

```
using System.ServiceModel;
using System.Security.Cryptography.X509Certificates;
using System.Net.Security;
```

24. Scroll back down to the body of button1_Click(), which is where we'll do most of our modifications. The first code we add will do a little error-checking to make sure that the user selected an item (cut-and-paste this):

```
if (listBox1.SelectedItem == null)
{
   OutputBox.Text += "Please select a specimen\n\n";
   return;
}
```

25. Right after this code, add the following code to create the proxy to the service, specifying that all communications will take place over Transport Level Security (TLS) and that the client will use a certificate for authentication:

```
BasicHttpBinding binding =
                new BasicHttpBinding(BasicHttpSecurityMode.Transport);
binding.Security.Transport.ClientCredentialType =
                HttpClientCredentialType.Certificate;

CaTissueCorePortTypeClient proxy =
      new CaTissueCorePortTypeClient(binding,
            new EndpointAddress(caTissueServiceBox.Text));
```

26. Next, right after the previous code, insert the following code to enable the selection of a certificate to use:

```
X509Store myStore = new X509Store("My", StoreLocation.CurrentUser);
myStore.Open(OpenFlags.ReadOnly | OpenFlags.OpenExistingOnly);
X509Certificate2Collection selectedCerts =
      X509Certificate2UI.SelectFromCollection(myStore.Certificates,
                "Certificate selector",
                "Choose your caGrid credential to use.",
                X509SelectionFlag.SingleSelection);
X509Certificate2 selectedCert = null;
if (selectedCerts.Count > 0)
{   // if user hits cancel 0 certs are returned.
    selectedCert = selectedCerts[0];
}
if (selectedCert == null)
{
    OutputBox.Text += "No certificate selected. Ending. \n";
    return;
}

proxy.ClientCredentials.ClientCertificate.Certificate = selectedCert;
```

27. We complete the client by adding the code to construct the CQL query, present the query to the service over the secure channel (*note that we know that that this is secure because the service will reject the request if the client does not present a valid credential*), and print the results to the GUI:

```
ListBoxItem specType = (ListBoxItem)listBox1.SelectedItem;

OutputBox.Text += "Asking " + caTissueServiceBox.Text +
                " for number of " + specType.ContentStringFormat +
                " samples\n\n";

QueryRequestCqlQuery arg = new QueryRequestCqlQuery();
arg.CQLQuery = new CQLQuery();
arg.CQLQuery.Target = new Object();
arg.CQLQuery.Target.name = (String) specType.Content;

arg.CQLQuery.QueryModifier = new QueryModifier();
arg.CQLQuery.QueryModifier.countOnly = true;

CQLQueryResults result = proxy.query(arg);

CQLCountResult count = (CQLCountResult)result.Items[0];

OutputBox.Text += "Answer: " + count.count + " samples\n";
```

28. Lastly, by default, Windows expects all host certificates (such as for e-commerce sites) to have names such as amazon.com, www.amazon.com, buy.com, etc., which is NOT the format that caGrid services use. In this and the next step, we'll add simple code in the client to bypass this check. Note that the client will still confirm that the service has the private key associated with the public key in the service's certificate. Also note that there is a more sophisticated check that we'll ignore for tutorial simplicity. First, add this code as a new method in the Window1 class (e.g., right before *public Window1()*):

```
static private bool IgnoreCertificateChainHandler
   (object sender,
    System.Security.Cryptography.X509Certificates.X509Certificate
        certificate,
    System.Security.Cryptography.X509Certificates.X509Chain chain,
    System.Net.Security.SslPolicyErrors sslPolicyErrors)
  {
     return true;
  }
```

29. In the constructor for Window1, right after the invocation of "InitializeComponent()", add:

```
System.Net.ServicePointManager.ServerCertificateValidationCallback =
  new RemoteCertificateValidationCallback(IgnoreCertificateChainHandler);
```

30. You're done! Rebuild the solution and execute the client by hitting F5. Select one of the specimen to count and hit the "Click here to start" button. After you select your certificate, it could take as long as 30-45 seconds for the service to respond, so be patient.