Deploy 3-Tier

Some WPF applications and all Silverlight applications have three (or more) tiers:

1. Remote client
2. Application server (DevForce)
3. Database server

There are an enormous variety of ways to locate and configure the server tiers. We’ll take the first (and easiest) step in that direction by re-arranging the *CodeFirstWalk* sample into three projects (application, model, and server).

We’ll deploy the DevForce application server to the Visual Studio development server, simulating an IIS locally hosted deployment. Then we configure the application to communicate with that server. Finally, we run two instances of the application and see evidence of their requests to the server recorded in the server-side DevForce “Debug Log”.

The whole process takes about five minutes.

[Learn more about DevForce n-Tier deployment](http://drc.ideablade.com/xwiki/bin/view/Documentation/configure%2Ddeploy).

# Add and configure the Server

**Add** | **New Project** | **DevForce BOS Web Application**

Call it “CodeFirstWalk.Server”

**Go to *Web.config***

We’re about to clean up the *Web.config*. Leaving it alone is ok; the app will still run. But you don’t want unused XML cluttering your configuration and this XML would impede an attempt to simultaneously support Silverlight clients from this same server.

**Delete entire <*services*/> , <*bindings*/>, <*extensions*> tags inside the <*serviceModel*/> tag**

Final Web.config:

<?xml version="1.0"?>

<configuration>

  <configSections>

    <section name="ideablade.configuration"

type="IdeaBlade.Core.Configuration.IdeaBladeSection, IdeaBlade.Core"/>

  </configSections>

  <connectionStrings>

    <!-- Remember to add your connection string(s) here -->

  </connectionStrings>

  <ideablade.configuration version="6.00" xmlns="http://schemas.ideablade.com/2010/IdeaBladeConfig" >

    <logging logFile="log\DebugLog.xml"/>

    <!-- Additional configuration can be added to override defaults.

         See the sample config files in the Learning Resources for more information.

    -->

  </ideablade.configuration>

  <system.serviceModel>

    <!-- Set aspNetCompatibilityEnabled to true to allow use of ASP.NET security features.

         Set multipleSiteBindingsEnabled to true for Azure or if your web site has multiple http bindings.

    -->

    <serviceHostingEnvironment aspNetCompatibilityEnabled="false"

                               multipleSiteBindingsEnabled="true" />

  </system.serviceModel>

  <system.web>

    <compilation debug="true" targetFramework="4.0" />

  </system.web>

</configuration>

**In *Global.asax***, **uncomment** the following line:

System.Web.Hosting.HostingEnvironment.RegisterVirtualPathProvider(

new IdeaBlade.EntityModel.Web.ServiceVirtualPathProvider());

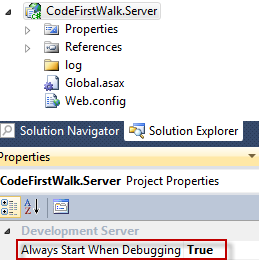
Notice that the web project is configured to start automatically when we’re debugging. The *CodeFirstWalk* application project remains the “StartUp Project”

**Delete the two .svc files**; we won’t need because we’ll create the services dynamically thanks to the change we just made in *Global.asax*.

**Add a reference to the model project**,*CodeFirstWalk.Model*.

**Remove reference to *IdeaBlade.EntityModel.Edm*** (optional) because we don’t use the EDM in Code First.

The web project now looks like this in Solution Explorer.



The “Always Start When Debugging” flag ensures that the server is running even when a different project is the “StartUp Project”.

# Reconfigure the application project

**Open the *App.config*** in the CodeFirstWalk application project,

**Find <ideablade.configuration> tag** and **paste** within it:

 <objectServer remoteBaseURL="http://localhost"

             serverPort="9009"

             serviceName="EntityService.svc"

             >

   <clientSettings isDistributed="true" />

 </objectServer>

These few lines of XML are all it takes to run as a distributed application. You can revert to the original 2-tier mode anytime by changing the “*isDistributed*” flag to “false”.

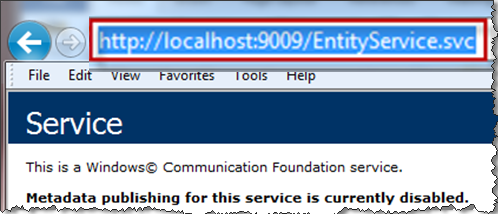
As you relocate the server, the *BaseURL*, *serverPort*, and *serviceName* will change accordingly.

# Run as 3-tier

**Select** the *CodeFirstWalk.Server* web project in Solution Explorer.

**Right-click** | **Debug** | **Start New Instance** to launch the service.

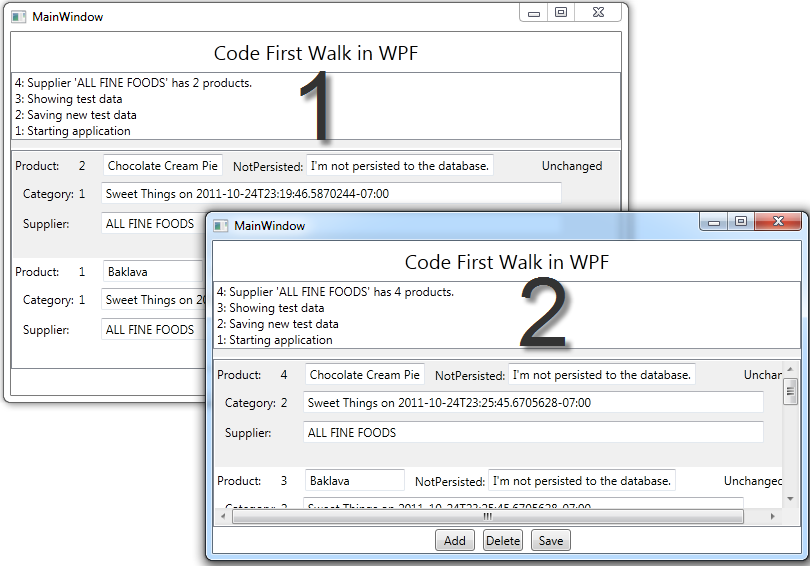
**Enter the service address in the browser address bar**. When you see a page like the following, the web application is alive and ready to respond to requests



**Close this browser** as it is not needed.

**[F5] to build and run a first instance of the client application.**

**Execute a second instance** by opening the “***bin/Debug***” for the *CodeFirstWalk* application project and launching “*CodeFirstWalk.exe*”. The two running instances look like this.



**Open the *DebugLog.xml*** in a browser by navigating to the “log” folder of the web project (purple box).

The log shows the server activity generated by requests from the two client application instances (*Guest – 1*, *Guest – 2*).

The *Guest-1* entry in the red box shows that it retrieved all *Product*s. The *Guest-2* entries in the blue box show the second application instance retrieving the first *Supplier* and then all *Product*s:

