

Silverlight 프로그램 개발 (Logging, TDD, Build, 통신)

2011.11

리얼웹 개발본부



0. 문서 이력

◆ 문서 요약

프로젝트 명	RCL.NET	고객사	리얼웹
문서 제목	Silverlight 프로그램 개발	작성자	배성혁
작성일자	2011.11.04	버전	1.0
단계	교육	기타	

◆ 문서 이력

번호	변경일자	변경자	승인자	내용
1.0	2011.11.04			최초 작성

목차

1. Silverlight 개요
2. Silverlight 프로그램 개발
 1. Logging by NLog
 2. Testing by Nunit and Microsoft.Silverlight.Testing
 3. Build by NAnt
 4. Create .NET & Silverlight Code Both
3. Silverlight 통신
 1. Web Service와의 통신
 2. WCF Service와의 통신
 3. 일반적인 HttpHandler와의 통신
4. Silverlight 용 Library 소개
 1. RCL.Core for Silverlight
 2. 기타 (IoC, Compression, Database, Parallelism (<http://debop.egloos.com/4063532>))

1. Silverlight 개요



Microsoft Silverlight 는 웹에서 차세대 **Microsoft.NET** 기반 미디어 환경 및 풍부한 대화형 어플리케이션 (RIA)을 제공하기 위한 **다중 브라우저, 다중 플랫폼 플러그인**입니다.

- ◆ 웹에서 미디어 환경 및 풍부한 대화형 애플리케이션 제공 - 비디오, 애니메이션, 대화형 작업, 매력적인 사용자 인터페이스를 통합합니다.
- ◆ 사용자가 매끄럽고 빠르게 설치 - 크기가 2MB 미만이고 모든 주요 브라우저에서 작동하는 설치가 쉬운 작은 주문형 플러그인입니다.
- ◆ Windows 기반 컴퓨터와 Macintosh 컴퓨터 간에 일관된 경험 - 추가 설치 요구 사항은 없습니다.
- ◆ 보다 풍성하고 뛰어난 웹 환경 구성 - 클라이언트를 더욱 활용하여 향상된 성능을 제공합니다.
- ◆ 매력적인 벡터 기반 그래픽, 미디어, 텍스트, 애니메이션 및 오버레이 사용 - 그래픽 및 효과를 기존 웹 응용 프로그램으로 매끄럽게 통합할 수 있습니다.
- ◆ 기존 표준/AJAX 기반 응용 프로그램 활용 - 그래픽과 미디어로 풍부하게 만들고 Silverlight를 사용하여 성능과 기능을 향상시킵니다.
- ◆ <http://msdn.microsoft.com/ko-kr/silverlight/default.aspx>



1. Logging by NLog
2. Testing by NUnit and Microsoft.Silverlight.Testing
3. Build by NAnt
4. Create .NET & Silverlight Code Both

2 SILVERLIGHT 프로그램 개발

2.1 Logging 설정

1. NLog 참조 (libWSilverlight-4.0 에 있음)

```
private void Application_Startup(object sender, StartupEventArgs e)
{
    // 로깅 시스템 초기화
    InitializeNLog();

    // NUnit for Silverlight 초기화
    UnitTestSystem.RegisterUnitTestProvider(new NUnitProvider());

    RootVisual = UnitTestSystem.CreateTestPage();
}
```

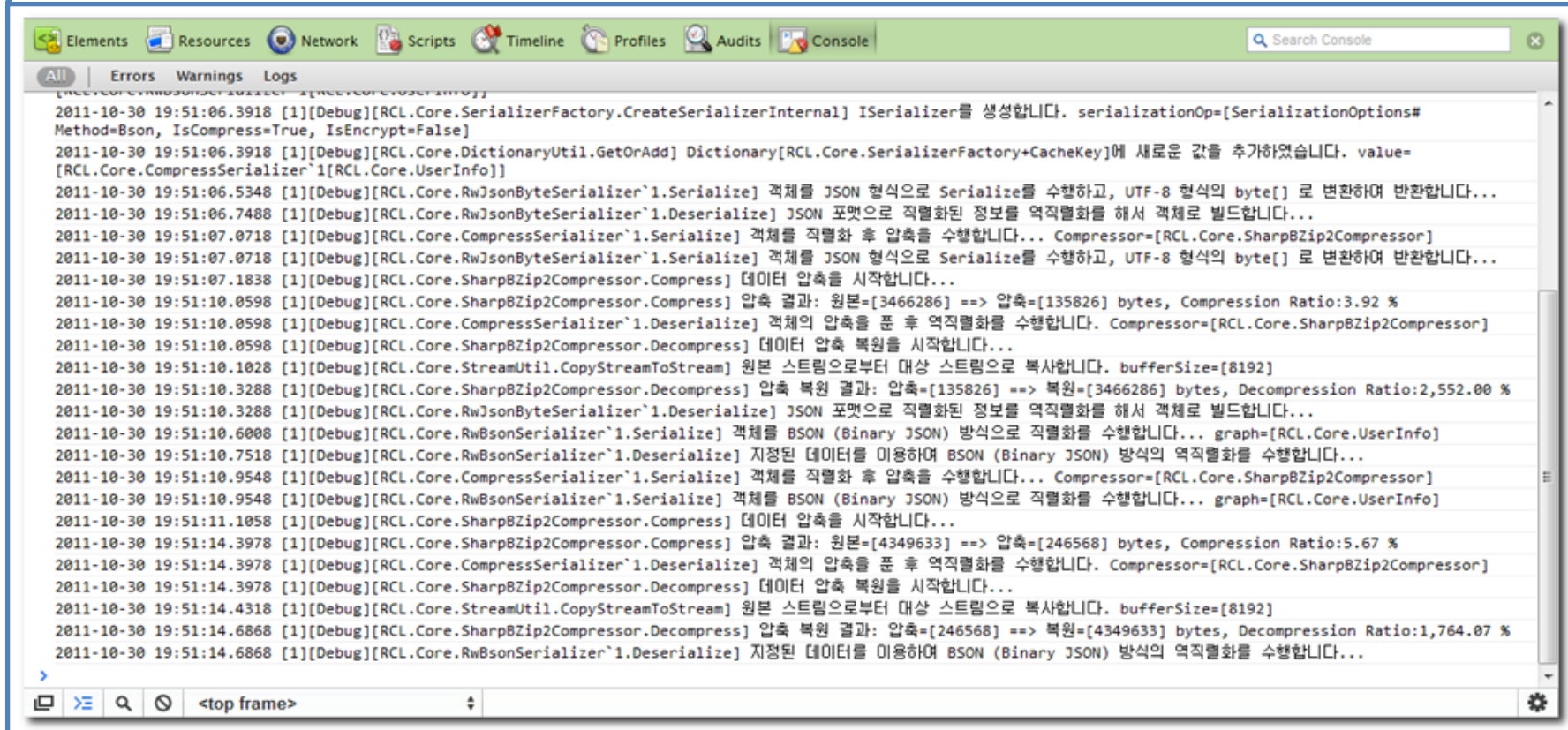
```
/// <summary>
/// 참고 : http://nlog-project.org/2010/04/16/nlog-2-0-for-silverlight-4-and-net-framework-4-0-preview-builds.html
/// </summary>
private static void InitializeNLog()
{
    // NOTE: 파일로 쓰는 것은 보안상에 문제가 있다. 그래서 Browser console.log 객체에 쓰도록 하는 BrowserConsoleTarget을 제작하였습니다.
    //
    SimpleConfigurator.ConfigureForTargetLogging(
        new BrowserConsoleTarget
        {
            Layout = new SimpleLayout("${longdate} [{threadid}][{level}][{callsite}] " +
                                      "${message} ${onexception:inner=${newline}${exception:format=tostring}}")
        },
        NLog.LogLevel.Debug);
    << 파일로 로그를 쓸 경우 - 아직 지원 안됨 >>
}
```


2.1 NLog 실행 결과

실버라이트 응용프로그램 또는 테스트 프로그램 실행 전에 아래 개발자 도구의 Console을 열어 놓은 상태에서 실버라이트 응용프로그램을 실행하면 로그가 찍힌다.

IE9 : 개발자도구 (F12) → 콘솔

크롬 : 설정 및 관리 → 도구 → 개발자도구 → Console



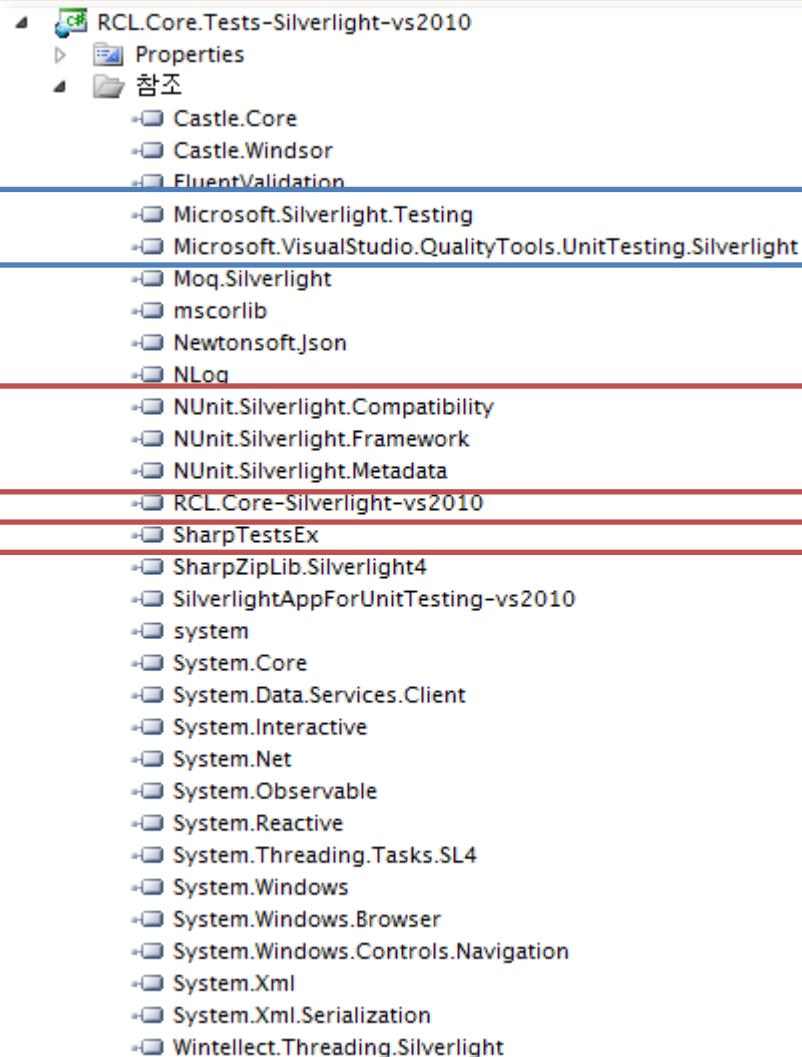
```
[RCL.Core.CompressSerializer`1][RCL.Core.UserInfo]
2011-10-30 19:51:06.3918 [1][Debug][RCL.Core.SerializerFactory.CreateSerializerInternal] ISerializer를 생성합니다. serializationOp=[SerializationOptions#
Method=Bson, IsCompress=True, IsEncrypt=False]
2011-10-30 19:51:06.3918 [1][Debug][RCL.Core.DictionaryUtil.GetOrAdd] Dictionary[RCL.Core.SerializerFactory+CacheKey]에 새로운 값을 추가하였습니다. value=
[RCL.Core.CompressSerializer`1][RCL.Core.UserInfo]
2011-10-30 19:51:06.5348 [1][Debug][RCL.Core.RwJsonByteSerializer`1.Serialize] 객체를 JSON 형식으로 Serialize를 수행하고, UTF-8 형식의 byte[] 로 변환하여 반환합니다...
2011-10-30 19:51:06.7488 [1][Debug][RCL.Core.RwJsonByteSerializer`1.Deserialize] JSON 포맷으로 직렬화된 정보를 역직렬화를 해서 객체로 빌드합니다...
2011-10-30 19:51:07.0718 [1][Debug][RCL.Core.CompressSerializer`1.Serialize] 객체를 직렬화 후 압축을 수행합니다... Compressor=[RCL.Core.SharpZip2Compressor]
2011-10-30 19:51:07.0718 [1][Debug][RCL.Core.RwJsonByteSerializer`1.Serialize] 객체를 JSON 형식으로 Serialize를 수행하고, UTF-8 형식의 byte[] 로 변환하여 반환합니다...
2011-10-30 19:51:07.1838 [1][Debug][RCL.Core.SharpZip2Compressor.Compress] 데이터 압축을 시작합니다...
2011-10-30 19:51:10.0598 [1][Debug][RCL.Core.SharpZip2Compressor.Compress] 압축 결과: 원본=[3466286] ==> 압축=[135826] bytes, Compression Ratio:3.92 %
2011-10-30 19:51:10.0598 [1][Debug][RCL.Core.CompressSerializer`1.Deserialize] 객체의 압축을 푼 후 역직렬화를 수행합니다. Compressor=[RCL.Core.SharpZip2Compressor]
2011-10-30 19:51:10.0598 [1][Debug][RCL.Core.SharpZip2Compressor.Decompress] 데이터 압축 복원을 시작합니다...
2011-10-30 19:51:10.1028 [1][Debug][RCL.Core.StreamUtil.CopyStreamToStream] 원본 스트림으로부터 대상 스트림으로 복사합니다. bufferSize=[8192]
2011-10-30 19:51:10.3288 [1][Debug][RCL.Core.SharpZip2Compressor.Decompress] 압축 복원 결과: 압축=[135826] ==> 복원=[3466286] bytes, Decompression Ratio:2,552.00 %
2011-10-30 19:51:10.3288 [1][Debug][RCL.Core.RwJsonByteSerializer`1.Deserialize] JSON 포맷으로 직렬화된 정보를 역직렬화를 해서 객체로 빌드합니다...
2011-10-30 19:51:10.6008 [1][Debug][RCL.Core.RwBsonSerializer`1.Serialize] 객체를 BSON (Binary JSON) 방식으로 직렬화를 수행합니다... graph=[RCL.Core.UserInfo]
2011-10-30 19:51:10.7518 [1][Debug][RCL.Core.RwBsonSerializer`1.Deserialize] 지정된 데이터를 이용하여 BSON (Binary JSON) 방식의 역직렬화를 수행합니다...
2011-10-30 19:51:10.9548 [1][Debug][RCL.Core.CompressSerializer`1.Serialize] 객체를 직렬화 후 압축을 수행합니다... Compressor=[RCL.Core.SharpZip2Compressor]
2011-10-30 19:51:10.9548 [1][Debug][RCL.Core.RwBsonSerializer`1.Serialize] 객체를 BSON (Binary JSON) 방식으로 직렬화를 수행합니다... graph=[RCL.Core.UserInfo]
2011-10-30 19:51:11.1058 [1][Debug][RCL.Core.SharpZip2Compressor.Compress] 데이터 압축을 시작합니다...
2011-10-30 19:51:14.3978 [1][Debug][RCL.Core.SharpZip2Compressor.Compress] 압축 결과: 원본=[4349633] ==> 압축=[246568] bytes, Compression Ratio:5.67 %
2011-10-30 19:51:14.3978 [1][Debug][RCL.Core.CompressSerializer`1.Deserialize] 객체의 압축을 푼 후 역직렬화를 수행합니다. Compressor=[RCL.Core.SharpZip2Compressor]
2011-10-30 19:51:14.3978 [1][Debug][RCL.Core.SharpZip2Compressor.Decompress] 데이터 압축 복원을 시작합니다...
2011-10-30 19:51:14.4318 [1][Debug][RCL.Core.StreamUtil.CopyStreamToStream] 원본 스트림으로부터 대상 스트림으로 복사합니다. bufferSize=[8192]
2011-10-30 19:51:14.6868 [1][Debug][RCL.Core.SharpZip2Compressor.Decompress] 압축 복원 결과: 압축=[246568] ==> 복원=[4349633] bytes, Decompression Ratio:1,764.07 %
2011-10-30 19:51:14.6868 [1][Debug][RCL.Core.RwBsonSerializer`1.Deserialize] 지정된 데이터를 이용하여 BSON (Binary JSON) 방식의 역직렬화를 수행합니다...
```

2.2 Unit Testing for Silverlight with NUnit

- ◆ UI 테스트가 아닌, Silverlight 용 단위 테스트
- ◆ 기존 Microsoft 의 SilverlightTool 개발 시에 사용했던 단위테스트 라이브러리는 UI Control 개발 시에 적합
 - 엄청난 비동기 기법을 사용해야 함
 - 테스트 자동화는 되지만, 실제 눈으로 확인하는 과정을 거쳐야 함.
- ◆ 다양한 테스트 방법과 풍부한 Assertion을 제공하는 NUnit 을 이용한 Silverlight 용 응용프로그램/라이브러리 단위테스트 기능 필요

2.2 Unit Testing for Silverlight with NUnit

Silverlight 응용프로그램에 6개의 라이브러리를 참조시킨다.



Microsoft 에서 제공하는 Silverlight 테스트용 라이브러리
(<http://archive.msdn.microsoft.com/silverlightut>)

NUnit for Silverlight
<http://code.google.com/p/nunit-silverlight/>

1.2 Unit Testing for Silverlight with NUnit

Silverlight App.xaml.cs 의 App 클래스에서 아래 코드와 같이 NUnitProvider 를 UnitTestProvider로 등록함

```
private void Application_Startup(object sender, StartupEventArgs e)
{
    // 로깅 시스템 초기화
    InitializeNLog();

    // NUnit for Silverlight 초기화
    UnitTestSystem.RegisterUnitTestProvider(new NUnitProvider());

    RootVisual = UnitTestSystem.CreateTestPage();
}
```

2.2 Unit Testing for Silverlight with NUnit

단위테스트 예

```
[Microsoft.Silverlight.Testing.Tag("JSON")]
[TestFixture]
public class JsonUtilTestCase
{
    #region << logger >>
    private static readonly NLog.Logger log = NLog.LogManager.GetCurrentClassLogger();
    private static readonly bool IsDebugEnabled = log.IsDebugEnabled;
    #endregion

    public const int ThreadCount = 5;
    private static readonly UserInfo user = UserInfo.GetSample();
    private static readonly JsonSerializerSettings _serializerSettings = JsonUtil.DefaultJsonSerializerSettings;

    [Test]
    public void SerializeAsText_DeserializeFromText()
    {
        TestUtil.RunTasks(ThreadCount,
            () =>
            {
                var serialized = JsonUtil.SerializeAsText(user);
                var deserializedUser = JsonUtil.DeserializeFromText<UserInfo>(serialized);
                VerifySerializer(user, deserializedUser);
            });
    }
}
```

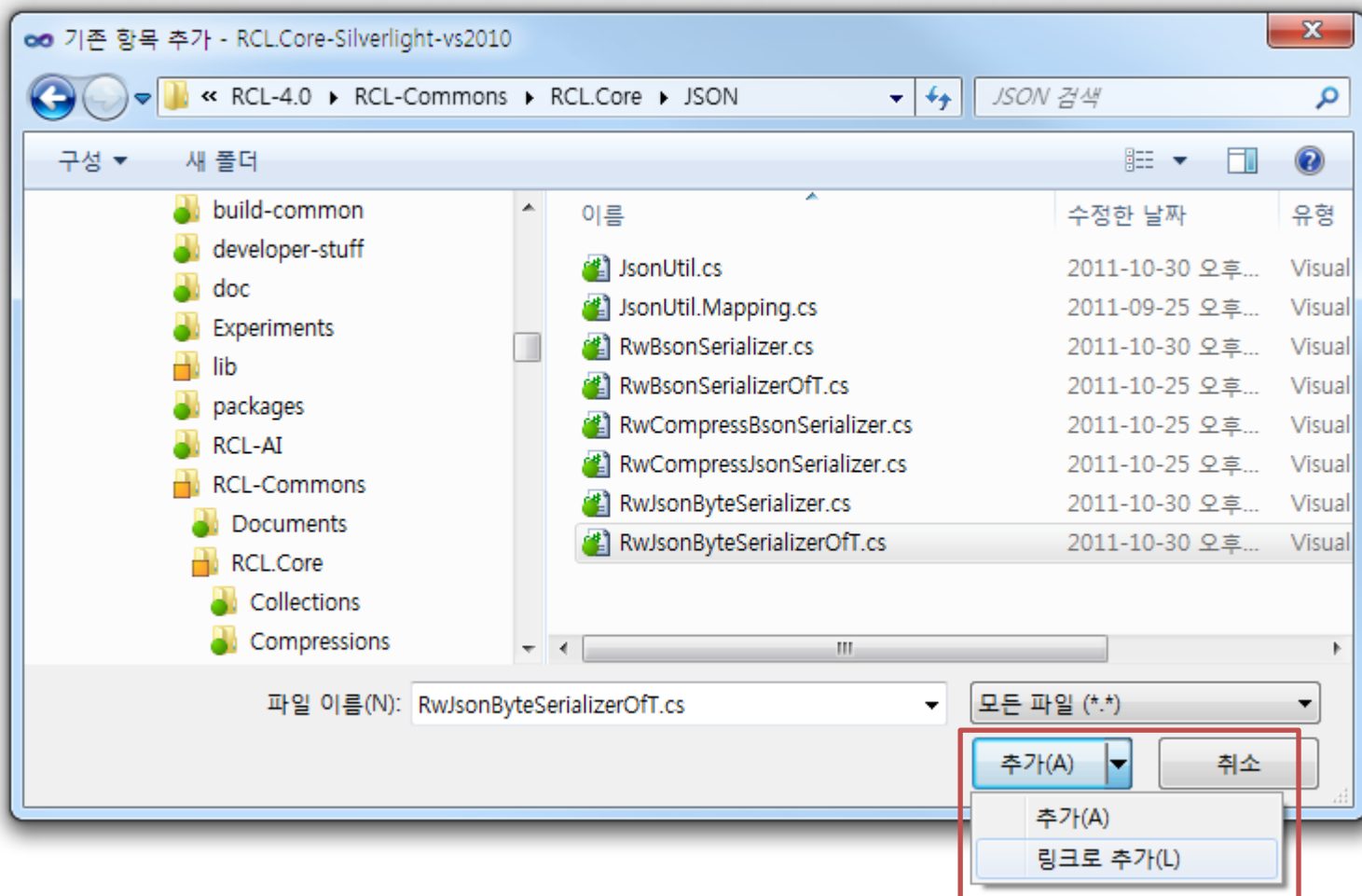
2.3 Build By NAnt

NAnt 를 이용한 빌드는 1. csc 를 이용하는 방식, 2. MSBuild 를 이용하는 방식이 있는데, RCL 에서는 2번 MSBuild 방식을 이용한다. 이 때에는 Project 파일의 참조 설정이 중요하다.

* RCL의 common.xml, common-project.xml 등도 변경되어야 합니다.

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <project name="RCL.Core.Tests"
3     default="build"
4     xmlns="http://nant.sf.net/release/0.91-alpha2/nant.xsd">
5
6     ...
14
15     <property name="root.dir" value="../../"/>
16     <include buildfile="${root.dir}/build-common/common-project.xml"/>
17
18     <target name="init" depends="common.init">
19         <property name="assembly.description" value="UnitTest of RCL Core Library for Silverlight"/>
20         <property name="assembly.is-cls-compliant" value="false" overwrite="true"/>
21         <property name="sign" value="false" overwrite="true"/>
22         <property name="build.unsafe" value="true"/>
23
24         <assemblyfileset id="project.referen" basedir="${build.dir}">...</assemblyfileset>
25
26         <resourcefileset id="project.resourc" dynamicprefix="true">...</resourcefileset>
27
28     </target>
29
30     <target name="generate-assemblyinfo" depends="init common.generate-assemblyinfo" />
31
32     <target name="build"
33         depends="generate-assemblyinfo common.compile-silverlight"
34         description="Build ${project::get-name()}">
35
36     </target>
37
38 </project>
```

2.4 Create .NET & Silverlight Code Both



링크로 추가 시에는 실제 코드는 하나이므로, 공통으로 사용하므로, .NET/Silverlight 공통으로 기능을 제공할 수 있다.

2.4 Create .NET & Silverlight Code Both

조건부 컴파일 이용

```
#if !SILVERLIGHT
    private static volatile Random myRandomGenerator = RwThread.CreateRandom();
#else
    private static volatile Random myRandomGenerator = new Random();
#endif
```

```
public static void Initialize()
{
#if !SILVERLIGHT
    Initialize(new RwContainer().Install(Configuration.FromAppConfig()));
#else
    Initialize(new RwContainer().Install(FromAssembly.This()));
#endif
}
```


2.4 Create .NET & Silverlight Code Both

```
#if SILVERLIGHT

using System.Diagnostics;
/*
 * 기존 .NET 코드와 Silverlight 코드 간의 재사용 시에 가장 문제가 되는 것이 Serializable Attribute입니다.
 * 이 SerializableAttribute를 SILVERLIGHT 컴파일 시에만 가짜로 만들어서 사용하도록 합니다.
 *
 * http://kozmic.pl/archive/2010/11/16/how-to-make-sharing-code-between-.net-and-silverlight-a.aspx
 */
namespace System
{
    ///<summary>
    /// .NET Serializable 을 흉내낸다.
    ///</summary>
    [Conditional("THIS_IS_NEVER_TRUE")]
    public class SerializableAttribute : Attribute { }
    ///<summary>
    /// .NET NonSerializableAttribute를 흉내낸다.
    ///</summary>
    [Conditional("THIS_IS_NEVER_TRUE")]
    public class NonSerializedAttribute : Attribute { }
}

#endif
```



1. Web Service 와의 통신
2. WCF Service 와의 통신
3. 일반적인 HttpHandler 와의 통신

3. SILVERLIGHT 통신

»»» 3.1 웹 서비스 통신

.NET Code - Synchronous

```
var requestBytes = ResolveRequestSerializer(productName).Serialize(requestMessage);
var responseBytes = client.Execute(requestBytes, productName);
return ResolveResponseSerializer(productName).Deserialize(responseBytes);
```

Silverlight Code - Asynchronous

```
var requestBytes = ResolveRequestSerializer(productName).Serialize(requestMessage);
var tcs = new TaskCompletionSource<byte[]>();
EventHandler<WebDataService.ExecuteCompletedEventArgs> handler = null;
handler = (sender, args) => EAPCommon.HandleCompletion(tcs, args, () => args.Result,
    () => client.ExecuteCompleted -= handler);

client.ExecuteCompleted += handler;
try
{
    client.ExecuteAsync(requestBytes, productName, tcs);
}
catch (Exception ex)
{
    if (log.IsErrorEnabled)
        log.ErrorException("웹 서비스 비동기 호출에 예외가 발생했습니다.", ex);
    client.ExecuteCompleted -= handler;
    tcs.TrySetException(ex);
}
var responseBytes = tcs.Task.Result;
return ResolveResponseSerializer(productName).Deserialize(responseBytes);
```

3.2 WCF 서비스 통신

.NET Code - Synchronous

```
var requestBytes = ResolveRequestSerializer(productName).Serialize(requestMessage);
var responseBytes = client.Execute(requestBytes, productName);
return ResolveResponseSerializer(productName).Deserialize(responseBytes);
```

Silverlight Code - Asynchronous

```
var requestBytes = ResolveRequestSerializer(productName).Serialize(requestMessage);
var tcs = new TaskCompletionSource<byte[]>();
EventHandler<WcfDataService.ExecuteCompletedEventArgs> handler = null;
handler = (sender, args) => EAPCommon.HandleCompletion(tcs, args, () => args.Result,
    () => client.ExecuteCompleted -= handler);

client.ExecuteCompleted += handler;
try
{
    client.ExecuteAsync(requestBytes, productName, tcs);
}
catch (Exception ex)
{
    if (log.IsErrorEnabled)
        log.ErrorException("WCF 서비스 비동기 호출에 예외가 발생했습니다.", ex);
    client.ExecuteCompleted -= handler;
    tcs.TrySetException(ex);
}
var responseBytes = tcs.Task.Result;
return ResolveResponseSerializer(productName).Deserialize(responseBytes);
```

▶▶▶ 3.2 WCF 통신 환경 설정

- ◆ WCF 통신은 환경설정에 의해 Protocol , 전송 방식, 전송 데이터 제한 등을 설정할 수 있다.
- ◆ Silverlight 에서는 basicHttpBinding 과 customBinding 만을 지원하여, .NET Client 와는 다르게 동작한다.
- ◆ 이에 우선은 웹 서비스와의 통신만을 사용하도록 한다.

3.3 IHttpHandler 와의 통신 (WebClient 사용)

.NET Code - Asynchronous byte[]

```
var requestBytes = ResolveRequestSerializer(productName).Serialize(requestMessage);
return
    client
        .UploadDataTask(uri, "POST", requestBytes)
        .ContinueWith(task =>
            {
                var responseBytes = task.Result;
                return ResolveResponseSerializer(productName).Deserialize(responseBytes);
            })
        .Result;
```

Silverlight Code - Asynchronous string

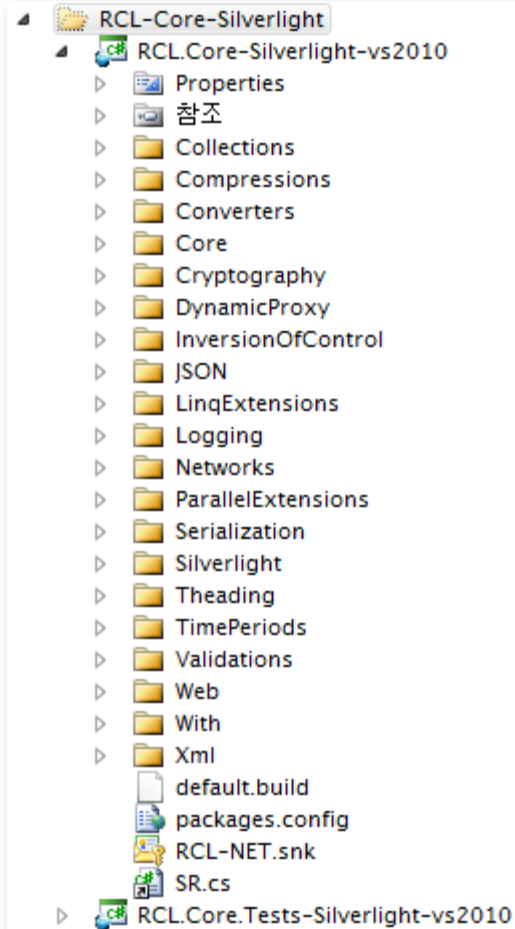
```
var requestBytes = ResolveRequestSerializer(productName).Serialize(requestMessage);
return
    client
        .UploadStringTask(uri, "POST", requestBytes.Base64Encode())
        .ContinueWith(task =>
            {
                var responseBytes = task.Result.Base64Decode();
                return ResolveResponseSerializer(productName).Deserialize(responseBytes);
            })
        .Result;
```




1. RCL.Core
2. 기타 라이브러리

3. SILVERLIGHT 용 라이브러리

3.1 RCL-Core for Silverlight



RCL.Core 의 대부분의 클래스를 지원함.

Silverlight 가 지원하지 못하는 부분에 대해서는

#if !SILVERLIGHT ... #endif 로 구분 할 수 있도록 하였음.

- ◆ Compressions
- ◆ Cryptography
- ◆ LinqExtensions
- ◆ ParallelExtensions
- ◆ TimePeriods

를 활용 할 것

3.2 Silverlight 용 라이브러리

분야	라이브러리	설명
IoC	Castle.Windsor	IoC/DI 를 Silverlight 에서 활용 할 수 있음. 단, 컴포넌트 등록을 XML이 아닌 Fluent API 를 사용 해야 함.
DB	Wintellect Sterling	Isolated File Storage를 사용하는 단순 DB http://sterling.codeplex.com/
DATA	JSON.NET	대표적인 JSON 라이브러리로서, 외부와의 데이터 통신 시에 객체를 JSON 형식으로 변환하여 통신을 수행한다.
UI	Reactive Extensions	RIA의 UI 에 대한 Reaction 관련 Library http://msdn.microsoft.com/en-us/data/gg577609
압축	Silverlight SharpZipLib	GZip, SharpBZip2 알고리즘을 제공 http://slsharpziplib.codeplex.com/
검사	FluentValidation	Fluent 방식의 Validation (Castle.Windsor 와 같 이 사용하면 좋다) http://fluentvalidation.codeplex.com/
TPL	Task Parallel Library	http://robertmclaws.com/nuget-packages/system-threading-tasks-for-silverlight

감사합니다