1) RunWithElevatedPrivileges?
2) Why can't we use RunWithElevatedPrivileges in event handlers?
3) Impersonation Improvements in SharePoint 2010 Event Receivers?
4) Best recommended practice use of it?
5) Best recommended practice to use of it in Event Receivers?
6) Best recommended practice to use of it in Feature Receivers?
7) RunWithElevatedPrivileges in visual studio workflows:
8) Is RunWithElevatedPrivileges allowed in sandbox solution?
9) By using which credentials the RunWithElevatedPrivileges will run?
10) Difference between SPSecurity.CodeToRunElevated and SPSecurity. RunWithElevatedPrivileges?

RunWithElevatedPrivileges?

As per the msdn: Executes the specified method with Full Control rights even if the user does not otherwise have Full Control.

Whenever we use SPSecurity.RunWithElevatedPrivileges (), it will execute the code under the context of Application Pool identity, so you must ensure that the App Pool account is a member of a site collection group with sufficient perms to add/edit/delete or whatever your code is trying to do. If not, the code will quietly break without popping an exception.

Method: Microsoft.SharePoint. SPSecurity.RunWithElevatedPrivileges

The code will not run within the elevated privilege if the object accessed was not created within the SPSecurity.RunWithElevatedPrivileges block. The main reason for doing so is to ensure that all the objects are in the context of the App Pool's identity.

Impersonation Improvements in SharePoint 2010 Event Receivers?

Instead of using RunWithElevatedPrivileges, In SharePoint 2010, there are new properties namely OriginatingUserToken, UserDisplayName and UserLoginName which help the developers to revert back to the original user who triggered the event very easily.

I will update once get the clear idea on what use of the new properties OriginatingUserToken, UserDisplayName and UserLoginName introduced in SharePoint 2010.

RunWithElevatedPrivileges in visual studio workflows:

No need to use any elevated privileges when working with workflows because it runs under SharePoint System Account by default (the App Pool account).

Is RunWithElevatedPrivileges allowed in sandbox solution?

You cannot use SPSecurity.RunWithElevatedPrivileges method in case of Sandboxed solution. The main reason is Sandbox solutions execute in User Code service with limited privileges.

Best recommended practice use of it:

Can't use **SPContext** inside the code being RunWithElevatedPrivileges. We may get "access denied" error if instead write the SPWeb site = SPContext.Current.Web inside the RunWithElevatedPrivileges, because your web was created in the context of the current user.

Best recommended practice is: Take the current context outside the SPSecurity.RunWithElevatedPrivileges block and then create a new instance of SPSite and SPWeb inside the block which will run under application pool identity.

Simply to say is: The objects you're working with need to be recreated within your RunWithElevatedPrivileges code block.

Best recommended practice #1:

```
private void Test()
{
   Guid webID = SPContext.Current.Web.ID;

   Guid siteID = SPContext.Current.Site.ID;

   SPSecurity.RunWithElevatedPrivileges(delegate()
   {
      using (SPSite site = new SPSite(siteID))
      {
         using (SPWeb web = site.OpenWeb(webID))
         {
            // Code Using the SPWeb Object goes here
         }
      }
   });
}
```

Best recommended practice #2:

```
private void Test()
{
   SPSite site = SPContext.Current.Site;
   SPWeb web = SPContext.Current.Web;
```

```
        SPSecurity.RunWithElevatedPrivileges(delegate()
        {
           using (SPSite CurrentSite = new SPSite(site.ID))
           {
              using (SPWeb CurrentWeb = CurrentSite.OpenWeb(web.ID))
              {
                 // Code Using the SPWeb Object goes here
              }
           }
        });
     }
```

"RunWithElevatedPrivileges" in Feature Receivers:

```
[Guid("b321499d-9b43-410e-8a8f-779ffb81d738")]
  public class Feature1EventReceiver : SPFeatureReceiver
  {
     public override void FeatureActivated(SPFeatureReceiverProperties properties)
     {
        try
        {
           using (SPSite spSite = properties.Feature.Parent as SPSite)
           {
              using (SPWeb spWeb = spSite.OpenWeb())
              {
                 SPSecurity.RunWithElevatedPrivileges(delegate()
                 {
                    //code here
                 });
              }
           }
        }
        catch (Exception ex)
        {
        }
     }
```

"RunWithElevatedPrivileges" in Event Receivers:

Better to use the ID properties of the properties object, to get new instances of SPSite, SPWeb and SPListItem.

If you need to run actions with elevated privileges on SPSite and SPWeb object use new SPSite(properties.SiteId); and site.OpenWeb(properties.RelativeWebUrl) instead of properties.web and web.site;

Best Recommended Practice #1

```
namespace MyCustomDlgFramework.EventReceiver
{
    /// <summary>
    /// List Item Events
    /// </summary>
    public class EventReceiver : SPItemEventReceiver
    {
       /// <summary>
       /// An item is being added.
       /// </summary>
       public override void ItemAdding(SPItemEventProperties properties)
       {
           SPSecurity.RunWithElevatedPrivileges(delegate()
           {
               using (SPSite site = new SPSite(properties.SiteId))
               {
                   using (SPWeb web = site.OpenWeb(properties.RelativeWebUrl))
                   {
                       //code here
                   }
               }
           });

       }

    }
}
```

Best recommended practice #2:

```
namespace MyCustomDlgFramework.MyEventReceiver
{
    /// <summary>
    /// List Item Events
    /// </summary>
    public class MyEventReceiver : SPItemEventReceiver
```

```
   {
     /// <summary>
     /// An item is being added.
     /// </summary>
     public override void ItemAdding(SPItemEventProperties properties)
     {
          using (SPSite site = new SPSite(properties.WebUrl))
        {
           using (SPWeb web = site.OpenWeb())
           {
              SPSecurity.RunWithElevatedPrivileges(delegate()
              {
                 //Code here
              });
           }
      }
   }
  }
}
```

Best recommended practice #3:

```
namespace MyCustomDlgFramework.EventReceiver
{
   /// <summary>
   /// List Item Events
   /// </summary>
   public class EventReceiver : SPItemEventReceiver
   {
     /// <summary>
     /// An item is being added.
     /// </summary>
     public override void ItemAdding(SPItemEventProperties properties)
     {
        SPSecurity.RunWithElevatedPrivileges(delegate()
        {
           using (SPWeb web = properties.OpenWeb())
           {
              //Code here
           }

        });
     }

   }
}
```

Chances of getting “Access Dined” error:

```
private void Test()
{
   SPSecurity.RunWithElevatedPrivileges(delegate()
   {
      SPWeb currentWeb = SPContext.Current.Web;
      SPList spList = currentWeb.Lists["MyList"];
   });
}
```

```
private void Test()
{
   SPSecurity.RunWithElevatedPrivileges(delegate()
   {
      using (SPSite currentSite = new SPSite(SPContext.Current.Site.Url))
      {
         using (SPWeb currentWeb = currentSite.OpenWeb())
         {
            // Access granted as System account!!
         }
      }
   });
}
```

```
private void Test()
{
   SPSecurity.RunWithElevatedPrivileges(delegate()
   {
      SPSite site = SPContext.Current.Site;
      SPWeb web = SPContext.Current.Web;
      web.AllowUnsafeUpdates = true;

      SPList list = web.Lists["MyList"];
      SPListItem item = list.GetItemById(1);
      item["MyField"] = "SharePoint";
      item.Update();
```

```
            web.AllowUnsafeUpdates = false;
        });
    }
```

Reason: SPContext.Current.Site and SPContext.Current.Web runs the List Item update code in the context of the currently logged in user and not in the context of the App Pool identity.