IronSP Documentation

# Installation

If you already installed a previous version, you should first do an uninstall.

1. Install the “IronSharePoint.wsp” solutions package  
   PS> Add-SPSolution “C:\IronSP\IronSharePoint.wsp”  
   PS> Install-SPSolution “IronSharePoint.wsp” –GACDeployment -WebApplication http://yourwebapp
2. Install the “IronSharePoint.IronRuby10.wsp” solutions package  
   PS> Add-SPSolution “C:\IronSP\IronSharePoint.IronRuby10.wsp”  
   PS> Install-SPSolution “IronSharePoint.IronRuby10.wsp” –GACDeployment

# Uninstall

1. Uninstall “IronSharePoint.wsp”   
   PS> Uninstall-SPSolution “IronSharePoint.wsp”   
   PS> Remove-SPSolution “IronSharePoint.wsp”
2. Uninstall “IronSharePoint.IronRuby10.wsp”   
   PS> Uninstall-SPSolution “IronSharePoint.IronRuby10.wsp”   
   PS> Remove-SPSolution “IronSharePoint.IronRuby10.wsp”

# Configuration

1. Activate the farm feature “IronSP Farm” in the central administration.
2. Activate the web application feature “IronSP WebApp” on every web application you want to use IronSP in the central administration.
3. Create an IronSP Hive Site Collection  
   PS>$hiveSite = new-SPSite “http://yourwebapp/sites/IronHive” -Template STS#1 –OwnerAlias “contoso\administrator”
4. Register the site collection as a trusted IronSP Hive. The registration will automatically activate the hidden “IronSP Hive Site” feature on the site.  
   PS> . “C:\IronSP\IronHiveRegistry.ps1”  
   PS> $ihr = Get-IronHiveRegistry  
   PS> $ihr.AddTrustedHive($hiveSite.Id)  
   PS> $ihr.Update($true)
5. To use the IronSP features on a site collection, you first have to associate the corresponding site collection, web application, subscription or the whole farm to an IronSP Hive Site Collection.   
   PS> $target = Get-SPWebApplication “http://intranet”  
   PS> #or map tenant to an IronSP Hive# $target = (Get-SPSite “http://intranet/sites/anytenatsite”).SiteSubscription  
   $ihr.AddHiveMapping($hiveSite, $target)  
   $ihr.Update($true)
6. Activate the “IronSP Site” Site Collection feature on any site collection you want to use it (Site Collection must be within a mapping scope (see 5.)

# IronSP Hive

The IronSP Site Collection stores all the sources for an IronSP Runtime. An IronSP Hive is always associated with one IronSP Runtime. The path to the Hive Library is “~site/\_catalogs/IronHive”. It is recommended that you organize your scripts within folders and that the folders correspond to namespaces or modules. Currently only Ruby 1.0 scripts with file extension \*.rb are supported. When you use the IronScript ContentType, the script gets executed each time you save it.

# IronSP Controls

## IronPart

IronPart is a WebPart that can use of dynamic code.

1. Activate the “IronSP Site” Site Collection feature
2. Add IronPart to a page.
3. Edit IronPart
   1. Enter the “ScriptName” (e.g. MyControls/TestControl.rb)
   2. Enter the “ScriptClass” (e.g. MyControls.TestControl)  
      Note: Classes have to be registered in the script via “ironRuntime.AddDynamicType(‘MyControls.TestControl’,MyControls::TestControl)”

Example Script:

module MyControls

class TestControl < IronControl

def Render(writer)

writer.Write(“Hello IronSP!”)

end

end  
end

# Register HttpHandler

You can register a dynamic HttpHandler. The handler is mapped to the url “\_iron/\*” Add the following code the “application.rb” .

class HttpHandler

include System::Web::IHttpHandler

def IsReusable

false

end

def ProcessRequest(context)

context.Response.Write("Hello Dynamic Handler")

end

end

$RUNTIME.RegisterDynamicType("HttpHandler",HttpHandler)

$RUNTIME.HttpHandlerClass = "HttpHandler"

# IronHive Events

You can register to the following IronHive events: ItemAdded, ItemUpdated, ItemDeleted, ItemFileMoved, ItemCheckedIn, ItemAdding, ItemUpdating, ItemDeleting, ItemFileMoving, ItemCheckingIn, ItemCheckingOut.

You should register the event handler within the application.rb.

Example Code:

# sender is SPItemEventReceiver; props is SPItemEventProperties

$RUNTIME.IronHive.Events do |sender, props|

if props.Event == "ItemUpdated"

sender.EventFiringEnabled = false

item = props.EventProperties.ListItem

item["Title"] = "Iron Hive Event receiver"

item.SystemUpdate()

sender.EventFiringEnabled = true

end

end

# Iron Event Receivers

You can register list item events on any list with the following code:

include Microsoft::SharePoint

class TestEventReceiver < SPItemEventReceiver

def ItemUpdated(props)

self.EventFiringEnabled = false

item = props.ListItem

item["Title"] = "Iron Hive Event receiver"

item.SystemUpdate()

self.EventFiringEnabled = true

end

end

$RUNTIME.RegisterDynamicType("TestEventReceiver", TestEventReceiver )

list = SPSite.new("http://intranet/sites/IronSharePoint").RootWeb.Lists["Announcements"]

# register will only register events that has not been registered before  
IronSharePoint::EventReceivers::IronItemEventReceiver.Register(list, SPEventReceiverType.ItemUpdated, SPEventReceiverSynchronization.Default, 1000, "TestEventReceiver")

# IronSharePoint::EventReceivers::IronItemEventReceiver .GetAllRegistered(list)

# IronSharePoint::EventReceivers::IronItemEventReceiver. Unregister(list, SPEventReceiverType.ItemUpdated, “TestEventReceiver”)

# IronSharePoint::EventReceivers::IronItemEventReceiver. IsRegistered(list, SPEventReceiverType.ItemUpdated, “TestEventReceiver”)

# Use a Local Development Hive

Use a debug build and set the environment variable “IRONSPDEVHIVE” to your local hive path.

PS> [Environment]::SetEnvironmentVariable("IRONSPDEVHIVE", "C:\Sourc es\LocalHive", "Machine”