Adding a service provider to the Process Engine

Owner: Resultmaker Research & Development
Version: 6.0.0
Revision: 2012-12-21

Michael Aagaard Pedersen

Introduction

This document describes the overall concept of the new way of writing scripts in Process Platform 6.0.

1.1 Definitions, Acronyms and Abbreviations

- Process Engine - Part of Resultmaker Process Platform*. The runtime software for executing workflows and forms. Consists of the Workflow Engine and the Script Engine.
- Resultmaker Process Platform - The product Resultmaker Process Platform is a business process management platform. With Resultmaker Process Platform you can design and execute your processes.

2 Process Engine Service Provider

A Process Engine service provider is a dynamically loaded class that may be used from scripts. Service providers are accessed from scripts via the generic GetService<>() method, and they may be served to consumers in one of three ways:

- From the engine itself
- From the security provider configured for the engine
- From a list of service providers declared in Process Engine configuration

The point of service providers is to enable scripts to use services defined by a contract, or interface, without having to know their implementation. The GetService<>() generic method chooses an implementation for the interface given when it is called. That way, the underlying implementation of a service may change radically without the script having to be changed, as long as the interface does not change. Figure 1 shows a code sample for accessing a Service Provider.

```
var fileStoreSoap = eventArgs.GetService<FileStoreSoap>();
```

Figure 1 Code snippet for accessing a Service Provider

To add a class to the Process Engine service provider, the class must fulfill the following requirements:

1. It must implement an interface. Technically, it could also inherit from an abstract class.
2. It's most "greedy" constructor (the one that takes the most arguments) must only have arguments that may be implicitly converted from a string.
3. Failing that, a constructor fulfilling the requirements of 2) must be marked with the structuremap attribute: DefaultConstructor[1].

If a class fulfills these requirements it may be added to the Process Engine service provider by declaring it under the "configuration\Resultmaker.OC\ServiceProviders" element as a StructureMap DefaultInstance element[2], as in the following example:

```
<ServiceProviders MementoStyle="Attribute">
  <DefaultInstance
PluginType="Resultmaker.ProcessEngine.Logging.ILog,Resultmaker.Proce
ssEngine.Logging"
PluggedType="Resultmaker.ProcessEngine.Logging.Logger,Resultmaker.Pr
ocessEngine.Logging"
Scope="Singleton" />
</ServiceProviders>
```

The PluginType attribute must contain the name and assembly of the interface representing the service that the class implements.

The PluggedType attribute must contain the name and assembly of the class itself.

The Scope attribute determines lifecycle management for the service, as described in the StructureMap documentation here: http://structuremap.sourceforge.net/Scoping.htm#section2.

[1] See the StructureMap documentation for more details at: http://structuremap.sourceforge.net/UsingAttributes.htm
[2] See the structuremap documentation at:
http://structuremap.sourceforge.net/XmlConfiguration.htm

In addition, attributes, if given the same name as a constructor argument or public property may be used to provide initialization for the class as in:

```
ConnectionStringName="MyConnectionString"
```

The classes must reside in assemblies available to the Process Engine, that is, in the GAC or in the Process Engine bin folder (normally C:\Inetpub\wwwroot\OC4\bin).