Programmentwicklung Advanced II
Modulübergreifende Projektarbeit

Reflection / Serialization

Auftrag

Ihre Anwendung soll mit beliebigen Datenexportern erweitert werden können. Konkret soll Ihre Anwendung in der Lage sein, zur Laufzeit mittels Reflection Assemblies aus einem bestimmten Verzeichnis zu laden. Anschliessend sollen diese bzgl. Klassen durchsucht werden, die vorgegebene Kriterien erfüllen (z.B. Implementierung eines bestimmten Interfaces). Diese Klassen sollen danach nach Bedarf instanziiert werden.
Durch dieses einfache Plugin-System ist es möglich, Ihre Anwendung mit beliebiger Funktionalität (im konkreten Fall für Datenexport) zu erweitern, ohne dass der Code Ihrer Anwendung angepasst werden muss.

Reflection

Damit das möglich ist, müssen Sie aus der Anwendung ein entsprechendes Interface zur Verfügung stellen (siehe Listing 1).

```
/// <summary>
/// Plugin-Interface für DataExporter
/// </summary>
public interface IDataExportPlugin {
    /// <summary>
    /// Name des DataExporters
    /// </summary>
    string Name { get; }
    /// <summary>
    /// Ausführung des Export.
    /// </summary>
    /// <param name="data">Collection, die exportiert werden soll</param>
    /// <param name="destinationPath">Zielpfad des Exports
    /// (Verzeichnis und Dateiname)</param>
    void Export(IEnumerable data, string destinationPath);
}
```

Listing 1: Plugin-Interface

Nun können Drittanbieter eigene DataExporter für Ihre Anwendung implementieren und zur Verfügung stellen. Dazu muss die entsprechende DataExporter-Klasse einfach das Interface gemäss Listing 1 implementieren. Anschliessend muss die Assembly, welche die Klasse beinhaltet, im richtigen Verzeichnis abgelegt werden. So ist Ihre Anwendung in der Lage, diese mittels Reflection laden kann. Ein Assembly kann auch mehrere DataExporter beinhalten.

Damit für die Implementierung des DataExporters nicht Ihre komplette Anwendung referenziert werden muss, bietet es sich an, dass Sie das Interface `IDataExportPlugin` in einem separaten Assembly, dem Plugin SDK, zur Verfügung stellen.

Abbildung 1 zeigt den schematischen Aufbau des Plugin-Systems.

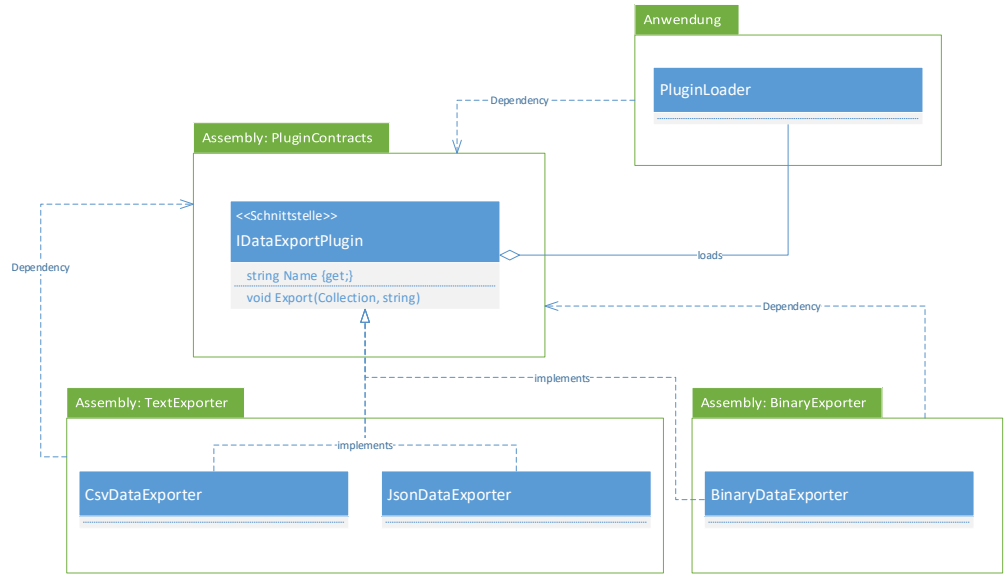

Abbildung 1: Plugin-System

Schematischer Ablauf:

1) Ihre Anwendung muss über einen `PluginLoader` verfügen. Dieser soll in der Lage sein, mittels Reflection Assemblies aus einem bestimmten Verzeichnis zu laden.
2) Der `PluginLoader` durchsucht alle geladenen Assemblies, ob Sie Klassen beinhalten, die das Interface `IDataExportPlugin` implementieren
3) Es werden alle Klassen, die das entsprechende Interface implementieren geladen und instanziiert.
4) Der PluginLoader stellt eine Liste aller erfolgreich geladenen DataExporter zur Verfügung. Diese können nun von Ihrer Anwendung verwendet werden.

Abbildung 2 zeigt den schematischen Ablauf in einem Sequenzdiagramm abgebildet.

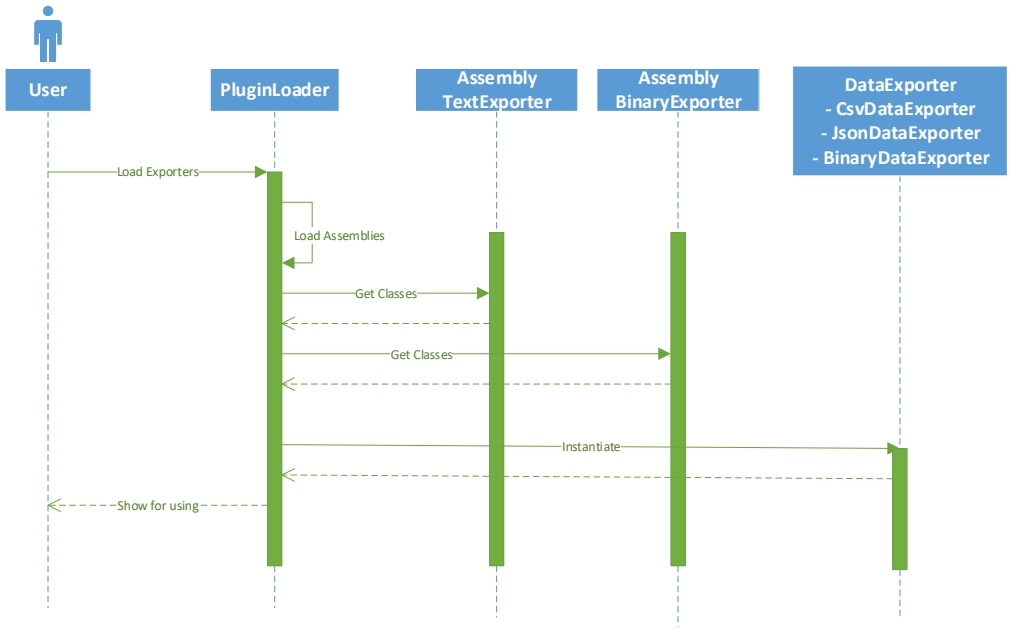

Abbildung 2: Schematischer Ablauf

Serialization	Zusätzlich zum Plugin-System sollen Sie die drei bereits erwähnten Data-Exporter implementieren: • BinaryDataExporter • JsonDataExporter • CsvDataExporter Der jeweilige DataExporter kann über die Methode `Export()` ausgeführt werden. Dieser Methode kann eine beliebige Collection sowie den Zielpfad für die Exportdatei übergeben werden (siehe auch Listing 1). Der Exporter muss anschliessend die Collection ins entsprechende Datenformat konvertieren und im Zielpfad speichern. BinaryDataExporter: Verwenden Sie für den BinaryDataExporter Binary Serialization[1] JsonDataExporter: Verwenden Sie für den JsonDataExporter Json.NET[2] von Newtonsoft. CsvDataExporter: Implementieren Sie einen eigenen CsvDataExporter. Für diese Aufgabe dürfen Sie ausschliesslich .NET-Bordmittel verwenden. D.h. Sie dürfen keine Libraries/Komponenten von Drittanbieter verwenden. Beachten Sie die für alle DataExporter zentrale Anforderung, dass es möglich sein soll, beliebige Collections zu exportieren. Ihre Aufgabe ist es, die DataExporter in Ihrer Anwendung für Kunden sowie LogEntries zur Verfügung zu stellen. Diese sollen natürlich via Plugin-System geladen werden – d.h. Ihre Anwendung darf selbstverständlich keine Referenzen (weder direkt noch indirekt) auf die drei DataExporter aufweisen.
Lieferergebnis	Als Abgabe wird der komplette Sourcecode als ZIP-Datei erwartet. Zusätzlich soll eine Kurzdokumentation erstellt werden, aus der die Handhabung Ihrer Anwendung hervorgehen. Diese Dokumentation soll alle nötigen Informationen beinhalten, damit ein Drittanbieter ohne Ihr zutun Plugins für Ihre Anwendung implementieren und zur Verfügung stellen kann.
Bewertung	Gemäss Initialdokument
Termin	Gemäss Moodle-Abgabe

[1] Binary Serialization:
https://docs.microsoft.com/en-us/dotnet/api/system.runtime.serialization.formatters.binary.binaryformatter
[2] Json.NET:
https://www.newtonsoft.com/json bzw. https://www.nuget.org/packages/Newtonsoft.Json/