

FLUX – Extension Points Overview

Datum 2011/02/09
Autor **Florian Luethi**

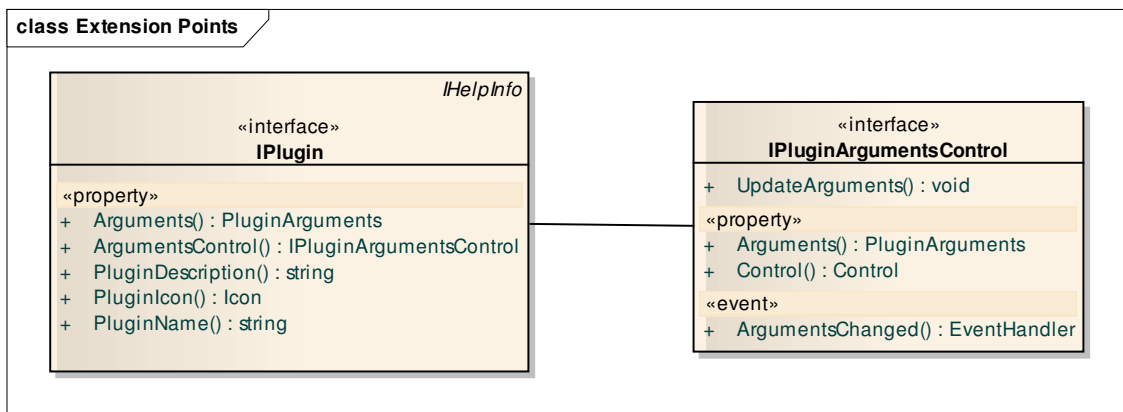


1 Einführung

Dieses Dokument hat zum Ziel, kurz eine Übersicht über die von FLUX angebotenen Extension Points zu zeigen. Der beste Weg zur Erweiterung besteht normalerweise darin, eines oder mehrere der weiter unten aufgeführten Interfaces zu Implementieren und am gegebenen Ort zu konfigurieren. Diese Implementationen sind straight-forward; ausserdem existiert ein FLUX SDK, welches unter anderem Tools zur automatisierten Erstellung von Fluxlet-Unit Tests oder Ressourcen enthält.

In den Klassendiagrammen sind nur jene Aspekte ersichtlich, welche für das generelle Verständnis wesentlich sind.

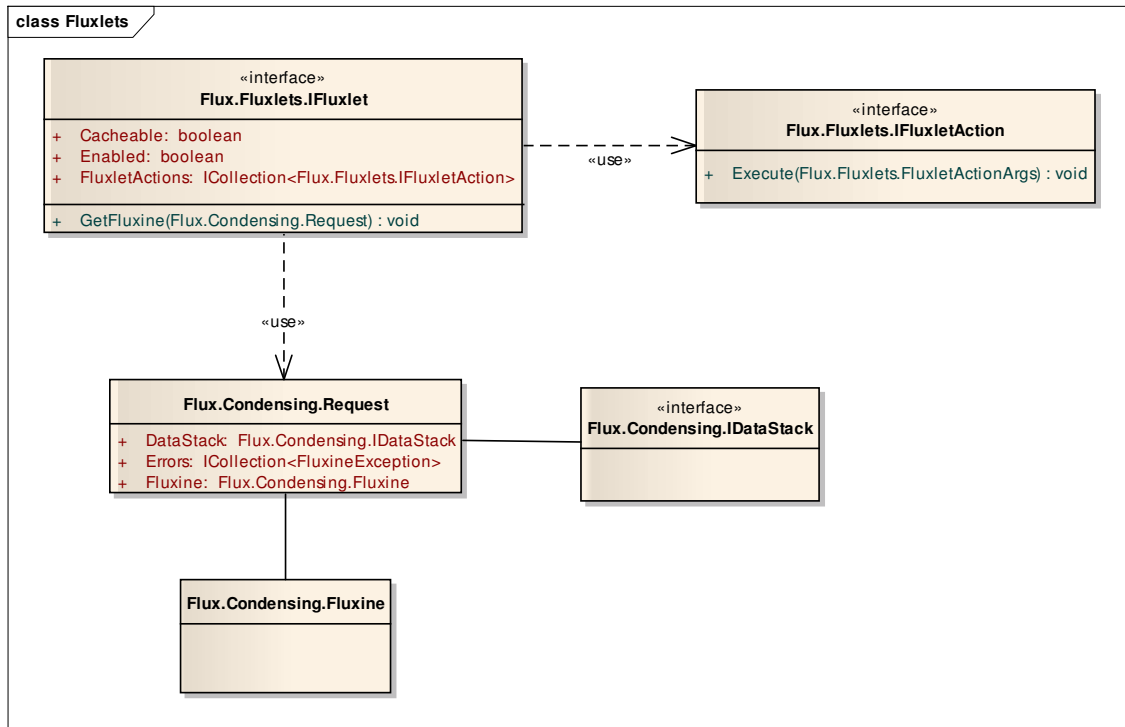
2 IPlugin



Alle Extension Point-Interfaces sind vom Interface Flux.Plugging.IPlugin abgeleitet. Dadurch wird jede Implementation vom Plugin-System erkannt und kann sich im Client graphisch repräsentieren und ein Control zur Konfiguration zur Verfügung stellen.



3 Fluxlets



Der wichtigste Extension Point sind die Fluxlets. Jedes Fluxlet implementiert das Interface Flux.Fluxlets.IFluxlet.

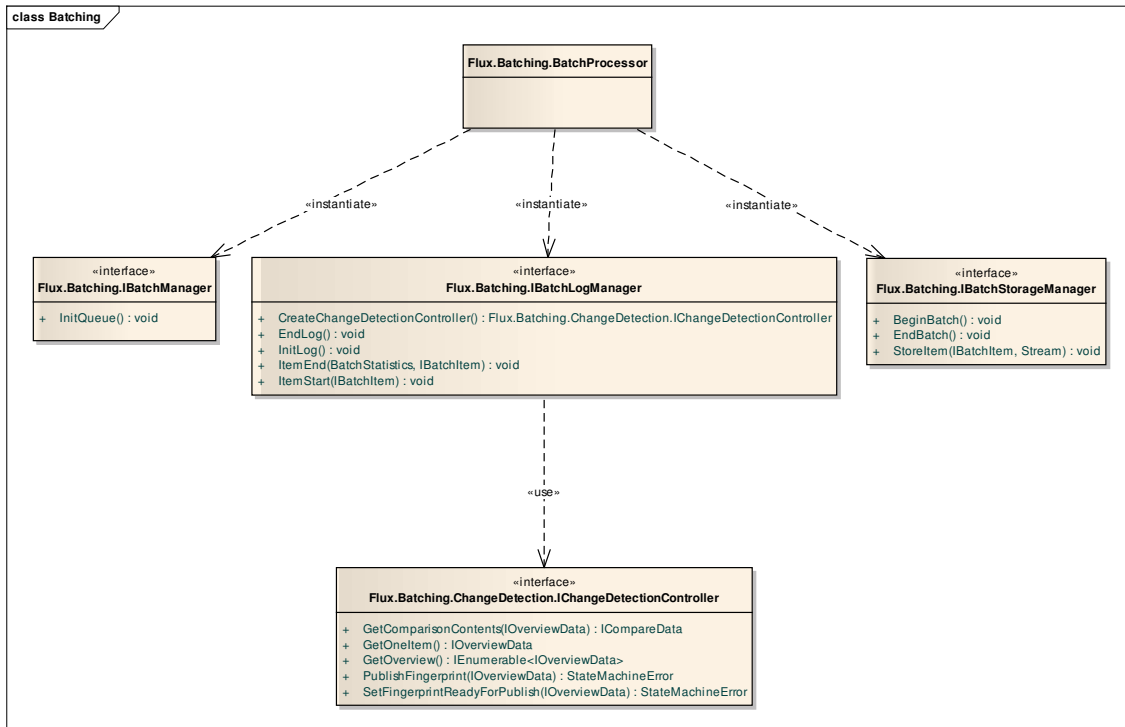
Die wichtigste Methode dieses Interfaces ist GetFluxine(). Ihr wird eine Instanz der Klasse Flux.Condensing.Request übergeben, worin dann die Manipulation des Fluxines (XML) vorgenommen werden kann. Ausserdem können allfällige Fehler in das Errors-Attribut eingefüllt werden. Diese Fehler werden dann vom Framework sinnvoll weiterprozessiert (z.B. wird dadurch verhindert dass ein Dokument durch die Massenproduktion abgelegt wird).

Im Attribut FluxletActions können Fluxlet-spezifische Aktionen definiert werden (z.B. Anzeige der Rohdaten für Enterprise Connectivity Fluxlets).

Für die Anwendungsbereiche Enterprise Connectivity Fluxlet und Validator-Fluxlet existieren Basisklassen, welche weitere anwendungsspezifische Basisinfrastruktur zur Verfügung stellen.



4 Batching



4.1 Flux.Batching.IBatchManager

Die Aufgabe dieses Interfaces ist das Füllen der Queue – d.h. die Batch Manager entscheiden welche Dokumente genau in einem Batch Run produziert werden. Normalerweise geschieht dies basierend auf Daten aus dem Stammdaten-Quellsystem. Dies ist ein weiterer wichtiger Extension Point.

4.2 Flux.Batching.IBatchLogManager

Die Aufgabe dieses Interfaces ist das Logging über den Batch Run. Während eines Runs wird die Methode ItemStart() vor der Produktion jedes Dokuments und die Methode ItemEnd() nach der Produktion jedes Dokuments aufgerufen. Damit können beispielsweise Produktionsstati zurück ins Stammdaten-Quellsystem geschrieben oder eine Dashboard-Website befüllt werden. Die FLUX-Standardimplementierung befriedigt damit auch die ganze inkrementelle Produktion (so können beispielsweise alle bereits fehlerfrei produzierten Dokumente für den aktuellen Run ausgenommen werden), oder auch die Change Detection.

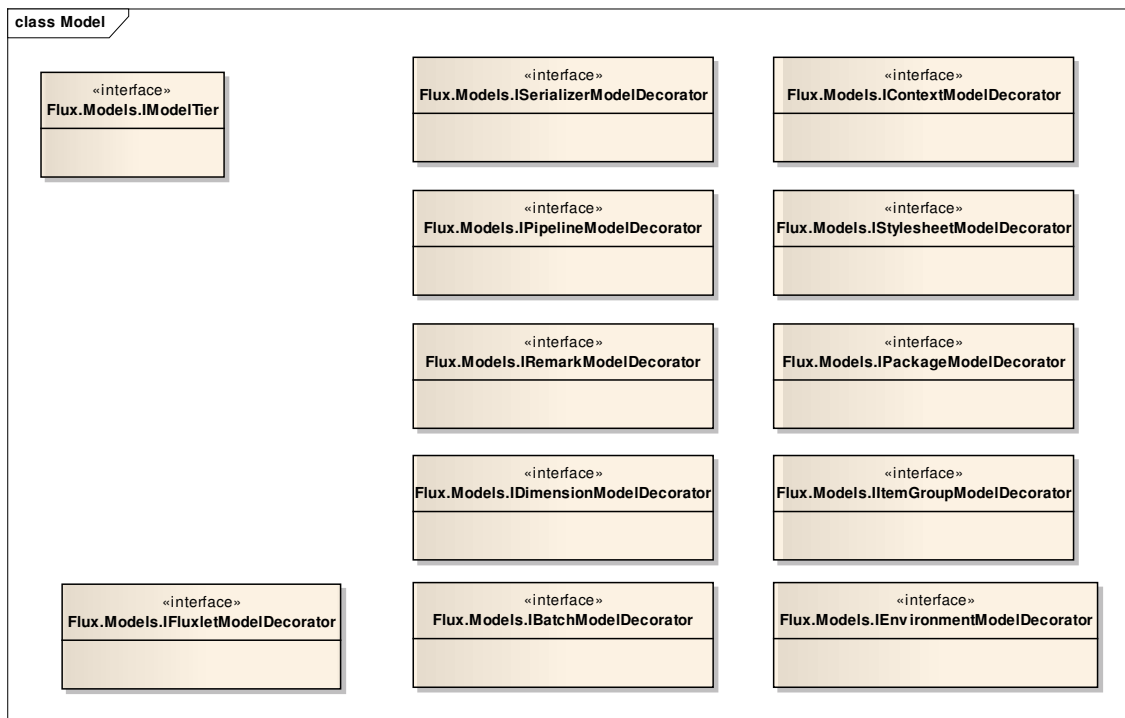


4.3 Flux.Batching.IBatchStorageManager

Die Aufgabe dieses Interfaces ist die Persistierung der produzierten Dokumente, zum Beispiel auf einem File-Share, in einer Datenbank, automatisches Verschicken per Email, automatisches Publizieren auf Sharepoint, FTP etcetera. Ausserdem wird die Produktion von Booklets über einen speziellen BatchStorageManager gelöst, in dem dieser das produzierte nicht-gerenderte Fluxine entgegennimmt, aggregiert und am Schluss den konfigurierten Renderer nochmals über das Ganze anwendet.

5 Model

Auch die Ablage der Konfigurationsdaten wird über einen Extension Point behandelt:



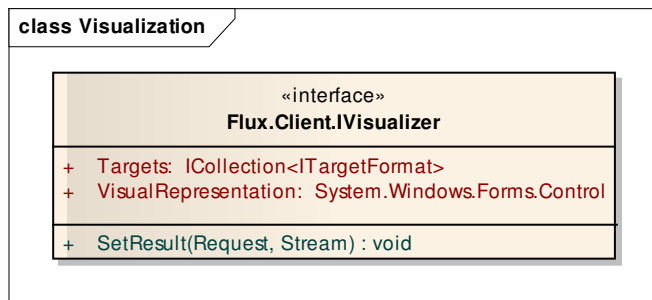
Durch die Implementation dieser Interfaces ist es möglich, die darunterliegende Datenhaltung zu ersetzen.

FLUX-Standardimplementationen sind Microsoft SQL Server, XML-Files sowie Roast. Dabei handelt es sich um einen IModelTier welcher einen beliebigen anderen IModelTier komplett lesen und daraus eine .Net –Assembly machen kann. Darin befindet sich dann für jedes Konfigurationselement ein hartkodierter Entscheidungsbaum, was extrem performant ist, allerdings Modifikationen nur durch ein erneutes Auslesen des Original-ModelTiers erlaubt. Eine sinnvolle Anwendung von Roast ist beispielsweise eine mit FLUX-Content betriebene Website.



6 Visualisierung

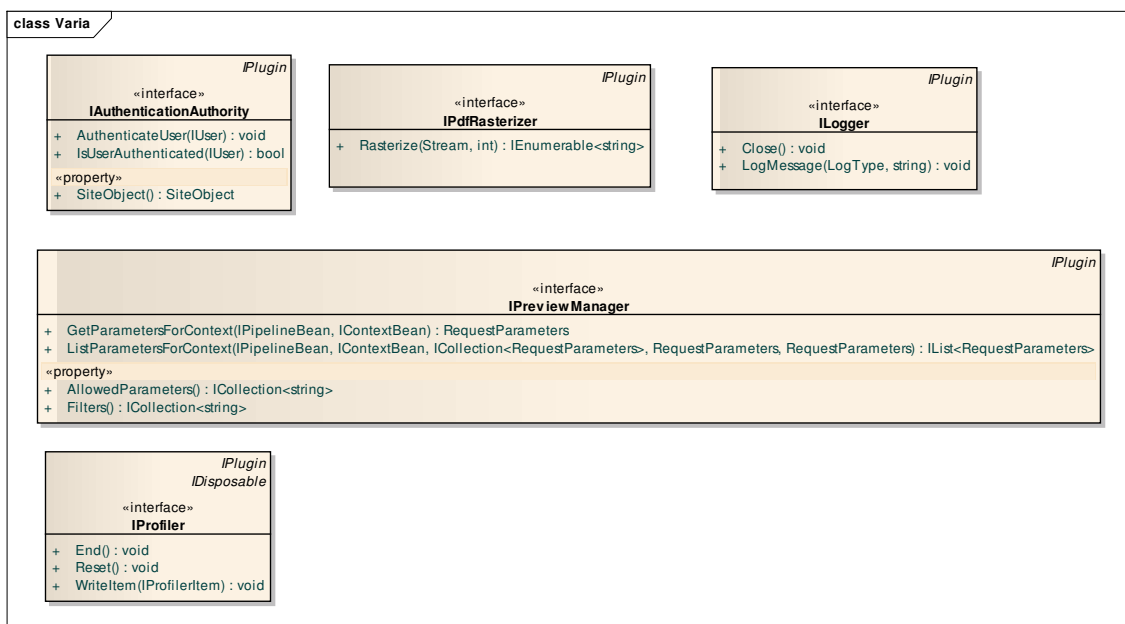
Da die Renderer in FLUX ebenfalls Fluxlets sind und dadurch auch Extension Points, muss auch der Client Extension Points zur Visualisierung des von den Renderern gelieferten Contents zur Verfügung stellen:



FLUX-Standardimplementierungen sind ein eingebetteter Acrobat Reader für PDFs, der Quasi-WYSIWYG-Editor sowie ein generischer XML-Viewer für das ungerenderte Fluxine.

7 Varia

Des weiteren verfügt FLUX über einige für die Integration in bestehende Landschaften wichtige Extension Points:





7.1 IAuthenticationAuthority

Dieser Extension Point bietet die Möglichkeit, zwischen dem Client beziehungsweise der Processing-Engine und der Datenbank (FLUX-Konfigurationsdatenbank wie auch Stammdaten-Systeme) eine Benutzererkennung und/oder rollenbasierte Authorisierung einzuschalten. Dies ist für allem für IT-Landscapes mit solcherart zentralisierten Services wünschenswert.

7.2 IPdfRasterizer

Der Context Tree des Clients kann die ganze Konfiguration analysieren und Beispiel-Dokumente für jeden Kontext mit integriertem Hervorheben der Unterschiede herstellen. Diese Funktionalität benötigt ein Programm das PDFs in Bitmaps verwandeln kann. Die FLUX-Standardimplementation dazu beherrscht das Aufrufen eines externen Tools (z.B. Ghostscript).

7.3 ILogger

Dieser Extension Point bietet die Möglichkeit, das von FLUX generierte Log (Programm-Fehler, Warnungen etc.) in externe Systeme weiterzuleiten. Die FLUX-Standardimplementation unterstützt Logging in Textfiles.

7.4 IPreviewManager

Dieser Extension Point bietet die Möglichkeit, den Context Tree in der möglichst sinnvollen Auswahl der Beispiel-Dokumente zu unterstützen. Da nur das Stammdaten-Quellsystem den Zusammenhang zwischen produzierten Dokumenten und Stammdaten (welche eben für die Population der Kontextkriterien ausschlaggebend sind) kennt, kann hier ein PreviewManager implementiert werden welcher genau diese Übersetzung vornimmt.

7.5 IProfiler

Dieser Extension Point bietet die Möglichkeit, Ausführung und Ausführungsdauer von FLUX-internen Operationen zu profilieren und abzulegen. Die FLUX-Standardimplementation unterstützt Profiling und Aggregation in Textfiles, welche dann von einem Tool aus dem FLUX SDK wieder gelesen und dargestellt werden können.