Hibernation of Advanced Railroad Trains (ArrT)

Schritt Eins – Sich entspannen und Ordnung machen.....

Unbound Objects (UBOs)

Dieser Hibernation Report ist ein "Snapshot", er wird nicht mehr upgedatet werden.

1 Anwendungsfälle

SrrTrains verfolgt das Ziel, Züge in beliebiger Art und Weise aus Schienenfahrzeugen zusammenzusetzen.

Das bedeutet eben nicht, dass Züge in einer Art Konfigurationsfile vordefiniert werden müssen, sondern es bedeutet eine freie Zusammenstellung, bei der zur Laufzeit immer wieder folgende Szenarien durchgeführt werden können:

- Kreieren eines "one-vehicle-train" (ein Schienenfahrzeug wird auf die Schienen gesetzt)
- Löschen eines einzelnen Fahrzeuges aus dem Zugsverband
- Zusammenfügen zweier Züge zu einem einzigen
- Auftrennen eines Zuges in zwei Züge

Weiters wollen wir das Handovern von Schienenfahrzeugen vorbereiten. Das heisst, dass jedes Fahrzeug innerhalb eines Zuges theoretisch jederzeit einem anderen Modul zugeordnet sein kann.

2 Der Ansatz

Aus den Anwendungsfällen können wir ableiten, dass jedes Fahrzeug jederzeit einem Modul zugeordnet ("assigned") ist. Sobald das Modul in einer Szeneninstanz geladen ist und das Fahrzeug diesem Modul "assigned" ist, wird das Fahrzeug sichtbar.

Der Zug hingegen ist keinem Modul zugeordnet, er ist ein "astrales Objekt". Der Zug an sich ist ja auch nicht sichtbar, nur die einzelnen Fahrzeuge sind sichtbar. Wenn also ein Zug über eine Modulgrenze fährt, und eines der beiden Module ist geladen, das andere ungeladen, dann wird es so sein, dass nur ein Teil der Fahrzeuge sichtbar ist.

Dieser Ansatz entspricht auch dem Credo, dass ein Modell niemals ein Teil eines Modells sein kann, dass es aber (unsichtbare) Container-Objekte geben kann, die mehr als ein Modell enthalten.

Mit diesem Ansatz haben wir also zwei Klassen von UBOs, also zwei Universal Object Classes (UOCs):

- **RailVehicles** (sichtbare Modelle von Schienenfahrzeugen – Lokomotiven, Waggons, ...)
- **Trains** (unsichtbare Container für Schienenfahrzeuge)

Um die RailVehicles mit den Trains zu verknüpfen, werden wir eine dritte Klasse (UOC) von astralen Objekten benötigen:

- **RailVehicleTrainRelation**

3 Implementierung

Bereits jetzt, nach dem Step 0033.10, gilt, dass jede SSC Extension, die eine oder mehrere UOCs realisieren möchte, für jede UOC einmal den **Prototypen SscDispatcher** instanziieren muss.

Zur Zeit werden UOCs nur im Zusammenhang mit UOC Parametern verwendet, jetzt im Step 0033.11 haben wir vor, diese auch zu verwenden, um Klassen von ungebundenen Objekten (UBOs) zu definieren.

Da nicht jede UOC auch ungebundene Objekte erzeugen muss, haben wir vor diese Funktion – die Verwaltung von ungebundenen Objekten – in einen eigenen Prototypen auszulagern, den **neuen Prototypen SscUboLoader**.

Jedem SSC Dispatcher kann optional ein UBO Loader zugeordnet werden, wenn die UOC auch UBOs kreieren soll.

Dafür wird der SSC Dispatcher ein SFNode Feld ***uboLoader*** bekommen, das defaultmäßig den Wert NULL hat. Wenn der User (also der Programmierer der SSC Extension) hier einen UBO Loader angibt, wird dieser verwendet um eine Klasse von UBOs zu verwalten.

3.1 Interfaces

Der UBO Loader ergänzt die Interfaces iiControl und eiDisp.

Das heisst, SSC Base wird mehr Funktionalität anbieten, um zusätzlich zum SSC Dispatcher auch den UBO Loader zu triggern bzw. von ihm getriggert zu werden (iiControl) und der UBO Loader ergänzt die Funktionalität des SSC Dispatchers, die von jeder SSC Extension genützt werden kann (eiDisp).

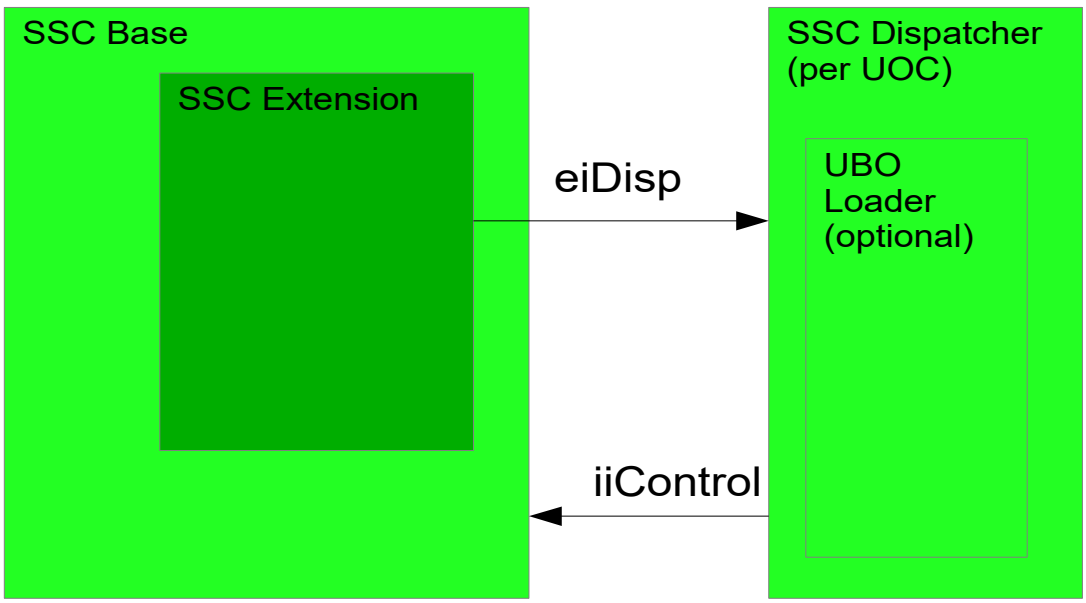

Abbildung 1: How to use the UBO Loader by an SSC Extension

Die Idee besteht darin, dass die **grundlegenden Szenarien**, also z.B.

- Kreieren eines UBO
- Zuordnen eines UBO zu einem Modul ("Assignen")
- Löschen eines UBO

über das iiControl Interface vom SSC Base erreichbar sind, damit diese grundlegenden Szenarien

- von jedem MIDAS Objekt über eiControl,
- vom Frame über uiControl

erreichbar sind.

Es soll aber auch möglich sein, dass der Progarmmierer der SSC Extension eigene, kompliziertere Szenarien aus den grundlegenden Szenarien zusammensetzt.

So wird zum Beispiel das "Kreieren eines one-vehicle-train" (OVT) aus folgenden Schritten bestehen

- Kreieren eines RailVehicle
- Kreieren eines Train
- Kreieren einer RailVehicleTrainRelation

3.2 Der "Existence State"

Der UBO Loader wird einen Network Sensor enthalten, um den globalen "Existence State" zu führen. Das ist der State, der die Existenz aller UBOs einer UOC darstellt.

Er besteht aus drei Arrays, die alle dieselbe Länge haben, nämlich die Maximalanzahl von UBOs dieser UOC

- uboObjIds.....objIds aller UBOs dieser UOC. Wenn ein Index frei ist, dann enthält dieses Element den Leerstring
- uboModuleIxs.....moduleIxs der Module, denen die UBOs dieser UOC zugeordnet sind. Wenn ein UBO keinem Modul zugeordnet ("assigned") ist, dann enthält dieses Element den Wert -1
- uboTypes.....objectTypes aller UBOs dieser UOC. Die möglichen Objekt Types sind in der "Dynamic Element Description" vordefiniert