Analisi Configurazioni e Modello Dati

Proposte su gestione dei dati

Sommario

[Sommario 1](#_Toc493516772)

[Obiettivo 1](#_Toc493516773)

[Premessa 1](#_Toc493516774)

[Dati di configurazione 2](#_Toc493516775)

[Dati Runtime 3](#_Toc493516776)

[Situazione Attuale 4](#_Toc493516777)

[Descrizione 4](#_Toc493516778)

[Problematiche relative alla attuale implementazione 4](#_Toc493516779)

[Architettura proposta 4](#_Toc493516780)

[Esempio di definizione di una gerarchia di oggetti 5](#_Toc493516781)

# Obiettivo

Gestione efficiente dei dati di configurazione e runtime/impostazione.

# Premessa

Un aspetto importante di ogni tipo di paradigma di programmazione è il trattamento dei dati.

Nel paradigma OOP la classica interpretazione è quella di incapsulare i dati dentro agli oggetti. Questi dati non vengono più manipolati direttamente, ma gli oggetti espongono i metodi per elaborarli.

Nei linguaggi procedurali, invece, i dati possono essere modificati liberamente da qualsiasi procedura e sta al programmatore eseguire di volta in volta le verifiche necessarie.

L’**incapsulamento** dei dati permette di cambiare più facilmente il comportamento dell’applicazione senza corrompere le funzionalità esistenti.

Come in tutte le applicazioni complesse, i dati da trattare sono essenzialmente di due tipi:

* Dati di configurazione
* Dati di runtime

I dati di configurazione sono usati per personalizzare una applicazione. Nel caso di un impianto Marini, il software Cyber500 ha bisogno, per esempio, di dover gestire un diverso numero di predosatori. Questo numero non è un dato che l’operatore deve cambiare, ma è un dato statico, che fa parte della specificità dell’impianto. Deve essere impostato in un qualche file e letto e usato allo start dell’applicazione.

## Dati di configurazione

Il file di configurazione sarà un file XML, descrittivo della struttura dell’impianto. Un qualcosa simile a questo:

<?xml version="1.0" encoding="utf-8"?>

<Impianto

xmlns="http://www.marini.fayat.com/impianto.xsd"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.marini.fayat.com/impianto.xsd file:///C:/Users/uts.MARINI/Documents/projects/new-project/oms/MariniImpianto/impianto.xsd"

id="Impianto"

name="Impianto"

type="string">

<ZonaPredosaggio id="ZonaPredosaggio" name="Zona Predosaggio">

<Predosatore id="Predosatore1" name="Predosatore 1">

<Property id="Comando" name="Comando" propertytype="Bool" bind="PLCTAG\_DO\_Predosatore1" bindtype="PLCTag" binddirection="OneWayToSource"/>

<Property id="Ritorno" name="Ritorno" propertytype="Bool" bind="PLCTAG\_DI\_RitPredosatore1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Termica" name="Termica" propertytype="Bool" bind="PLCTAG\_DI\_TermPredosatore1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Allarme" name="Allarme" propertytype="Bool" bind="PLCTAG\_DI\_AllPred1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Palpatore" name="Palpatore" propertytype="Bool" bind="PLCTAG\_DI\_PalpatorePred1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="LivelloMinimo" name="LivelloMinimo" propertytype="Bool" bind="PLCTAG\_DI\_LivMinPred1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Vibratore" name="Vibratore" propertytype="Bool" bind="PLCTAG\_DI\_VibratoreP1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Livello" name="Livello" propertytype="Int" bind="PLCTAG\_AI\_LivPredosatore1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Set" name="Set" propertytype="Int" bind="PLCTAG\_AO\_SetPredosatore1" bindtype="PLCTag" binddirection="OneWayToSource"/>

<Property id="SetLampada" name="SetLampada" propertytype="Bool" bind="PLCTAG\_DO\_LampadaP1" bindtype="PLCTag" binddirection="OneWayToSource"/>

<Property id="SetVibratore" name="SetLampada" propertytype="Bool" bind="PLCTAG\_DO\_VibratoreP1" bindtype="PLCTag" binddirection="OneWayToSource"/>

</Predosatore>

<Nastro id="Nastro1" name="Nastro 1">

<Motore id="Motore1" name="Motore 1">

<Property id="Presente" name="Presente" propertytype="Bool" bind="PLCTAG\_NM\_MOTORE1\_Presente" bindtype="PLCTag" binddirection="OneWayToSource"/>

<Property id="Comando" name="Comando" propertytype="Bool" bind="PLCTAG\_NM\_CMD\_SemiAuto\_1" bindtype="PLCTag" binddirection="OneWayToSource"/>

<Property id="Set" name="Set" propertytype="Int" bind="PLCTAG\_AO\_SetMotore01" bindtype="PLCTag" binddirection="OneWayToSource" />

<Property id="Ritorno" name="Ritorno" propertytype="Bool" bind="PLCTAG\_NM\_Ritorno\_1" bindtype="PLCTag" binddirection="OneWay" />

<Property id="Allarme" name="Allarme" propertytype="Int" bind="PLCTAG\_NM\_AllarmeMotore\_1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Blocco" name="Blocco" propertytype="Bool" bind="PLCTAG\_NM\_BloccoMotore\_1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="AccesoForzato" name="AccesoForzato" propertytype="Bool" bind="PLCTAG\_NM\_AccesoForzatoPLC\_1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="SpentoForzato" name="SpentoForzato" propertytype="Bool" bind="PLCTAG\_NM\_SpentoForzatoPLC\_1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="ValoreAmperometro" name="ValoreAmperometro" propertytype="Int" bind="PLCTAG\_NM\_MOTORE1\_Amperometri\_ValScal" bindtype="PLCTag" binddirection="OneWay"/>

</Motore>

</Nastro>

</ZonaPredosaggio>

<Property id="WatchDog" name="WatchDog" propertytype="Bool" bind="PLCTAG\_F\_WatchdogPC" bindtype="PLCTag" binddirection="TwoWay" handler="WatchDogPropertyHandler"/>

</Impianto>

Si tratta di un file XML che potrebbe avere molti livelli innestati, e del quale non si conosce a priori la quantità e la qualità di nodi figli innestati in un nodo parente.

L’incapsulamento di queste informazioni in oggetti (anch’essi innestati) può avvenire su una struttura dati predeterminata (un po’ come avviene ora), che sia progettata e abbastanza dimensionata da prevedere tutte le possibili combinazioni; oppure ci si può affidare a una costruzione di tipo ricorsivo, che permetta di adattare la struttura dati alle esigenze dei diversi impianti. Questa seconda soluzione si adatterebbe particolarmente bene al file XML, ricorsivo per sua natura.

In ambito OOP, un modo comune di usare i dati di configurazione è quello di leggerli allo start e usarli per decidere quali oggetti creare nella struttura dei dati principale. Il design pattern più utile ai nostri scopi è il **factory design** (<https://msdn.microsoft.com/it-it/library/cc185067.aspx>). Si tratta di un pattern per la costruzione di oggetti complessi per i quali non è nota a priori la struttura. In pratica viene definita una interfaccia di classe per la creazione di un oggetto base, lasciando ai tipi derivati la decisione su quale oggetto debba essere effettivamente istanziato. Nel nostro caso, la decisione del verrebbe presa a partire dal nome del nodo. Ad esempio un eventuale oggetto ZonaPredosaggio, di fronte al suo nodo:

<Nastro id="Nastro1" name="Nastro 1">

dovrebbe far costruire dal FactoryObject un oggetto Nastro, derivato da un qualche oggetto base OggettoBase.

Alrto esempio relativo agli impianti Marini: sappiamo che un impianto contiene predosatori che usano motori che usano amperometri. Un pattern di costruzione ricorsivo e un corretto file di configurazione permettono di creare il grafo degli oggetti a partire dalla semplice invocazione dell’oggetto impianto, senza doversi curare di implementare a mano l’incapsulamento degli uni negli altri. In questo modo se un giorno nascesse l’esigenza di aggiungere un predosatore, col semplice cambio del file di configurazione si otterrebbe la struttura di oggetti già pronta all’uso.

La **Dependency Injection** del pattern è implicita nel pattern stesso: il sottocomponente viene passato come parametro dei suoi costruttori. Il codice di ogni sottocomponente rimane così isolato.

Senza DI:

public SomeClass() {

myObject = Factory.getObject();

}

Con DI

public SomeClass (MyClass myObject) {

this.myObject = myObject;

}

L’altro pattern a cui far riferimento è sicuramente il **singleton**, perché il modello dei dati dovrà avere un’unica istanza attiva, fornendo un entry-point globale all’istanza stessa.

## Dati Runtime

Sono dati che l’applicazione non conosce fino al completamento della procedura di start. Ad esempio i dati inseriti dall’utente, quelli ricavati da un database oppure ottenuti da un web service. Si noti che per cambiare un dato di configurazione si deve usare un dato runtime.

Anche questi dati possono essere incapsulati in oggetti.

Come fa un consumer di questi dati ad avere l’istanza dell’oggetto che li contiene? Non avrebbe senso che se lo produca lui stesso, perché questo vorrebbe dire aver accesso ai dati, e quindi addio incapsulamento. L’unico modo è passargli un’istanza già costruita (singleton), magari come parametro di un metodo. Oppure lo stesso consumer può invocare l’istanza da un repository/factory che lo consegni o lo crei al posto suo. In questo modo il consumer non dovrebbe preoccuparsi della struttura interna del modello, ma solo dei suoi metodi.

Per i dati runtime complessi bisogna però tener conto che, se vengono implementati negli oggetti sia l’incapsulamento dei dati che i metodi che operano su di essi, questo potrebbe comportare problemi di manutenibilità del codice, in quanto ogni cambiamento potrebbe coinvolgere molti altri oggetti e interfacce.

Una soluzione potrebbe essere quella di fare oggetti che siano puri contenitori di dati incapsulati e una serie di altri oggetti che operino su di essi. Per esempio, un oggetto Documento che avesse bisogno di essere tradotto in un altro linguaggio, anziché avere un metodo interno dovrebbe appoggiarsi a un altro oggetto di tipo ITraduttore(Documento doc, Linguaggio lang). Questo approccio permette anche di avere una composizione più semplice della struttura dati, in quanto i comportamenti dei singoli oggetti sono tenuti separati, e di volta in volta che cambiano, posso intervenire su di essi in maniera semplice, in un unico punto. Inoltre, visto che il comportamento degli oggetti è mantenuto a parte, tutta la configurazione di questi comportamenti può essere fatta allo startup dell’applicazione.

In tutte queste soluzioni è comunque sempre raccomandabile che gli oggetti rimangano immutati e che ogni cambiamento produca un nuovo oggetto. Nel caso, il traduttore dovrebbe produrre un nuovo documento.

Un tipo speciale di dati runtime è lo **stato**.

Lo stato è un dato che cambia nel tempo, quindi ha un comportamento diverso dai paradigmi visti sopra. In questo caso un approccio corretto sarebbe quello di usare i getter/setter per ogni proprietà dell’oggetto e lasciare che sia un gestore esterno a occuparsi di mantenere l’istanza dei dati di stato e gestirne il comportamento a seconda delle richieste degli altri clienti in scrittura e lettura. Questo accentra anche l’accesso allo stato e migliora la comprensione in caso di debug.

# Situazione Attuale

Attualmente la struttura dei dati dell’impianto i file di configurazione principali sono:

parameters.xml

Si tratta di un file xml con un massimo di livelli innestati. In questo file vengono raccolti tutti i dati di configurazione dell’impianto, come per esempio i parametri di motori, nastri, amperometri.

Dallo stesso file viene dinamicamente creata la parte di applicazione in C# che ne permette la modifica.

Le strutture dati sono delle struct di VB6 sulle quali intervengono procedure sparse nei vari moduli. Molte di queste strutture sono richiamate in più moduli, rendendo un po’ dispersivo il codice.

# Descrizione

## Problematiche relative alla attuale implementazione

# Architettura proposta

I dati di configurazione, come già avviene, sono immagazzinati in un file XML di descrizione dell’impianto. Un file XML permette facilmente la descrizione di strutture complesse e innestate, quali sono quelle presenti negli impianti. Ad esempio un file come quello seguente descrive un impianto con un predosatoree un nastro. Nastro che a sua volta contiene un motore.

<?xml version="1.0" encoding="utf-8"?>

<Impianto

xmlns="http://www.marini.fayat.com/impianto.xsd"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.marini.fayat.com/impianto.xsd file:///C:/Users/uts.MARINI/Documents/projects/new-project/oms/MariniImpianto/impianto.xsd"

id="Impianto"

name="Impianto"

type="string">

<ZonaPredosaggio id="ZonaPredosaggio" name="Zona Predosaggio">

<Predosatore id="Predosatore1" name="Predosatore 1">

<Property id="Comando" name="Comando" propertytype="Bool" bind="PLCTAG\_DO\_Predosatore1" bindtype="PLCTag" binddirection="OneWayToSource"/>

<Property id="Ritorno" name="Ritorno" propertytype="Bool" bind="PLCTAG\_DI\_RitPredosatore1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Termica" name="Termica" propertytype="Bool" bind="PLCTAG\_DI\_TermPredosatore1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Allarme" name="Allarme" propertytype="Bool" bind="PLCTAG\_DI\_AllPred1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Palpatore" name="Palpatore" propertytype="Bool" bind="PLCTAG\_DI\_PalpatorePred1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="LivelloMinimo" name="LivelloMinimo" propertytype="Bool" bind="PLCTAG\_DI\_LivMinPred1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Vibratore" name="Vibratore" propertytype="Bool" bind="PLCTAG\_DI\_VibratoreP1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Livello" name="Livello" propertytype="Int" bind="PLCTAG\_AI\_LivPredosatore1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Set" name="Set" propertytype="Int" bind="PLCTAG\_AO\_SetPredosatore1" bindtype="PLCTag" binddirection="OneWayToSource"/>

<Property id="SetLampada" name="SetLampada" propertytype="Bool" bind="PLCTAG\_DO\_LampadaP1" bindtype="PLCTag" binddirection="OneWayToSource"/>

<Property id="SetVibratore" name="SetLampada" propertytype="Bool" bind="PLCTAG\_DO\_VibratoreP1" bindtype="PLCTag" binddirection="OneWayToSource"/>

</Predosatore>

<Nastro id="Nastro1" name="Nastro 1">

<Motore id="Motore1" name="Motore 1">

<Property id="Presente" name="Presente" propertytype="Bool" bind="PLCTAG\_NM\_MOTORE1\_Presente" bindtype="PLCTag" binddirection="OneWayToSource"/>

<Property id="Comando" name="Comando" propertytype="Bool" bind="PLCTAG\_NM\_CMD\_SemiAuto\_1" bindtype="PLCTag" binddirection="OneWayToSource"/>

<Property id="Set" name="Set" propertytype="Int" bind="PLCTAG\_AO\_SetMotore01" bindtype="PLCTag" binddirection="OneWayToSource" />

<Property id="Ritorno" name="Ritorno" propertytype="Bool" bind="PLCTAG\_NM\_Ritorno\_1" bindtype="PLCTag" binddirection="OneWay" />

<Property id="Allarme" name="Allarme" propertytype="Int" bind="PLCTAG\_NM\_AllarmeMotore\_1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="Blocco" name="Blocco" propertytype="Bool" bind="PLCTAG\_NM\_BloccoMotore\_1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="AccesoForzato" name="AccesoForzato" propertytype="Bool" bind="PLCTAG\_NM\_AccesoForzatoPLC\_1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="SpentoForzato" name="SpentoForzato" propertytype="Bool" bind="PLCTAG\_NM\_SpentoForzatoPLC\_1" bindtype="PLCTag" binddirection="OneWay"/>

<Property id="ValoreAmperometro" name="ValoreAmperometro" propertytype="Int" bind="PLCTAG\_NM\_MOTORE1\_Amperometri\_ValScal" bindtype="PLCTag" binddirection="OneWay"/>

</Motore>

</Nastro>

</ZonaPredosaggio>

<Property id="WatchDog" name="WatchDog" propertytype="Bool" bind="PLCTAG\_F\_WatchdogPC" bindtype="PLCTag" binddirection="TwoWay" handler="WatchDogPropertyHandler"/>

</Impianto>

Dal file viene ricostruita nell’applicazione la struttura dei dati di stato dell’impianto, al quale gli altri componenti potranno accedere mediante un gestore. Il gestore dell’impianto potrà in questo modo essere interrogato in vari modi (messaggistica, web, libreria,…) in modo da poter permettere varie soluzioni client.

* Object Property Subscribe / Set / Changed  
  Messaggi per la gestione di sottoscrizione, impostazione e variazione di Object properties.