Barbu Giulia-Andreea
Anul IV- IS

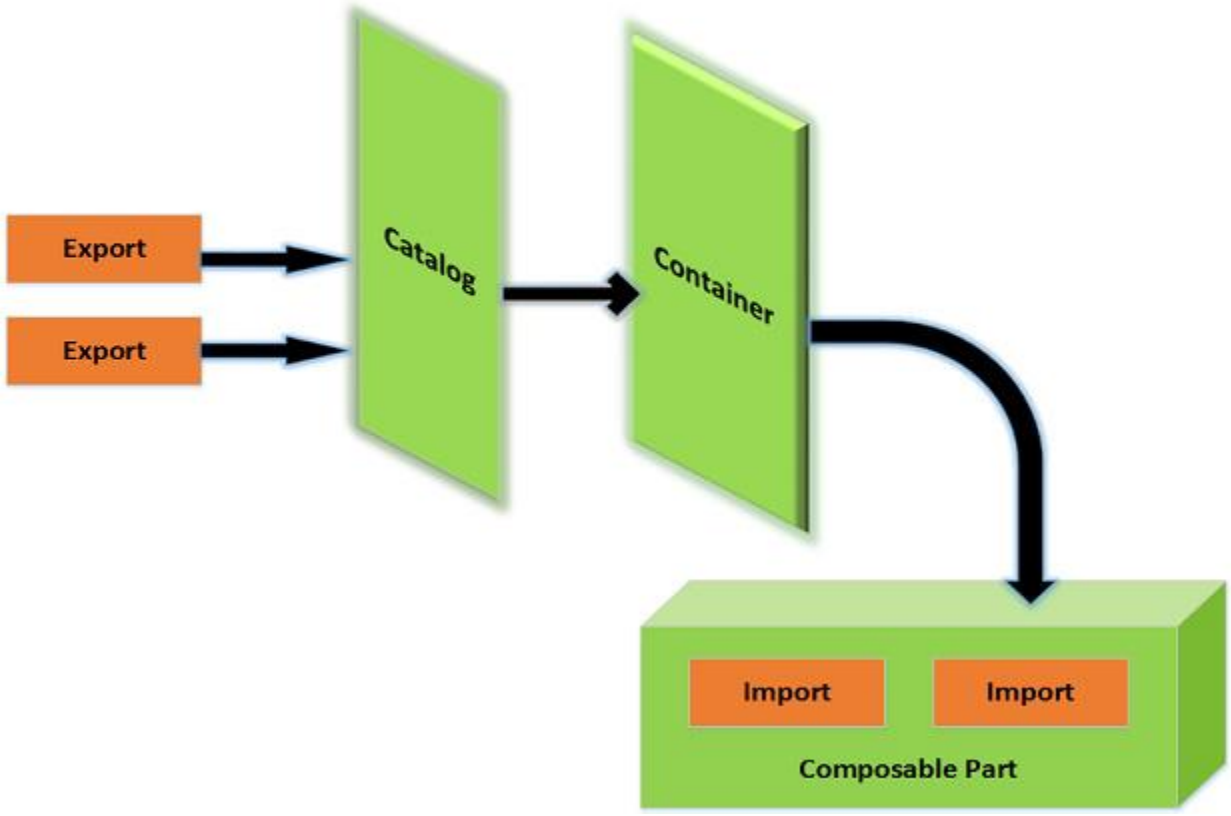
TEMA 2 PSSC - Dezvoltarea de sisteme software extensibile folosind Managed Extensibility Framework

Ce reprezintă MEF (Managed Extensibility Framework)?

Este o bibliotecă pentru crearea de aplicații extensibile. Acesta permite dezvoltatorilor să descopere și să utilizeze extensii fără o configurare necesară. De asemenea, permite dezvoltatorilor să încorporeze cu ușurință codul și să evite dependențele fragile.

Elementele constitutive ale MEF-ului sunt:

- Import
- Export
- Compose



Barbu Giulia-Andreea
Anul IV- IS

Unde este MEF disponibil?

MEF este o parte integrantă a .NET Framework 4 și este disponibil oriunde este utilizat .NET Framework. Putem utiliza MEF în aplicațiile client, indiferent dacă utilizează Windows Forms, WPF sau orice altă tehnolgie sau în aplicațiile server care utilizează ASP.NET.

Terminologiile utilizate:

1. **PARTEA**: o parte este un obiect(de exemplu: o clasă, o metodă sau o proprietate) care poate fi importat sau exportat în aplicație.
2. **CATALOG**: un obiect care ajută la descoperirea pieselor compacte disponibile dintr-un ansamblu sau un director.
3. **CONTRACT**: piesele de import si export trebuie să comunice între ele prin intermediul unui contract(de exemplu: o interfață sau un tip de date predefinit, cum ar fi șirul).
4. **IMPORT ATTRIBUTE**: definește necesitatea unei părți
5. **IMPORTMANY ATTRIBUTE**: este similar cu Import Attribute, dar acceptă multiple atribute de export
6. **EXPORT ATTRIBUTE**: atributul de import face referire la necesități, iar cel de export se referă la îndeplinirea acestora
7. **COMPOSE**: este zona în care părțile exportate vor fi asamblate cu cele importate.

Problema extinderii

Imaginați-vă că sunteți arhitectul unei aplicații mari care trebuie să ofere suport pentru extensibilitate. Cererea dvs. trebuie să includă un număr mare de componente mai mici și este responsabilă de crearea și difuzarea acestora.

Cea mai simplă abordare a problemei este **includerea componentelor** ca și cod sursă în aplicație și **apelarea acestora direct din cod**.

Acest lucru are o serie de dezavantaje evidente:

- nu putem adăuga componente noi fără a modifica codul sursă, o restricție care poate fi acceptată, de exemplu, în cazul unei aplicații Web, dar nu poate fi utilizată într-o aplicație client.
- la fel de problematic este si faptul că este posibil să nu avem acces la codul sursă al componentelor, deoarece acestea ar putea fi dezvoltate de 3 părți și, din același motiv, nu avem drept de acces.

O abordare puțin mai sofisticată ar fi **furnizarea unei interfețe** pentru a permite decuplarea între aplicație și componentele sale. În acest model, este posibil să oferim o interfață pe care o componentă o poate implementa și un API pentru a permite interacțiunea cu aplicația. Acest lucru rezolvă problema solicitării accesului la codul sursă, dar are totuși dificultăți proprii.

Deoarece aplicația nu dispune de capacitate de descoperire a componentelor pe cont propriu, trebuie să se spună în continuare în mod explicit, **ce componente sunt disponibile** și ar trebui încărcate. Acest lucru este de obicei realizat **prin înregistrarea** explicită a componentelor disponibile **într-un fișier de configurare**. Aceasta înseamnă că asigurarea faptului că componentele sunt corecte devine o problemă de întreținere, mai ales dacă este vorba de utilizatorul final și nu dezvoltatorul care se așteaptă să facă actualizarea.

În plus, **componentele sunt incapabile sa comunice una cu alta**, cu exceptia canalelor definite ale aplicației. Dacă arhitectul aplicației nu a anticipat necesitatea unei comunicări particulare, este de obicei imposibil.

În cele din urmă, dezvoltatorii de componente trebuie să accepte o dependență puternică de ceea ce ansamblul conține : interfața unei componente. Acest lucru face dificilă utilizarea unei componente în mai multe aplicații și poate crea probleme și atunci când este creat un cadru de testare pentru componente.

Care sunt beneficiile MEF?

O componentă MEF, numită parte, specifică atât dependențele sale (cunoscute sub numele de importuri), cât și capacitățile (cunoscute sub numele de exporturi) pe care le pune la dispoziție.

Când se creează o parte, motorul compoziției MEF satisface importurile sale cu ceea ce este disponibil din alte părți.

Pe lângă exporturile furnizate, o parte poate specifica importurile sale, care vor fi completate de alte părți. Acest lucru face **comunicarea între părți** nu numai **disponibilă**, dar **ușoară** și permite **factorizarea** codului. De exemplu, serviciile comune mai multor componente pot fi incluse într-o parte separată și ușor modificate sau înlocuite.

Concluzie

- Conceptele de bază ale MEF sunt: părțile, cataloagele și containerul
- Părțile și containerul reprezintă elementele de bază ale unei aplicații MEF.
- O parte este orice obiect care importă sau exportă o valoare.
- Un catalog oferă o colecție de piese dintr-o anumită sursă.
- Containerul de compoziție utilizează componentele furnizate de un catalog pentru a efectua compoziția, legarea importurilor de exporturi.
- Importurile și exporturile reprezintă modul în care componentele comunică. Cu import, componenta specifică necesitatea unei anumite valori sau a unui obiect, iar cu un export specifică disponibilitatea unei valori. Fiecare import este însoțit de o listă a exporturilor prin contract.