SPRING.NET

O Spring.Net é um framework desenvolvido tendo base a plataforma Java, portanto ele é implementado em sistemas que possuem as JVM (Java Virtual Machine), para entendermos o Spring.Net é necessário entender alguns conceitos importantes, como o que é um framework e da onde surgiu sua necessidade, e entendermos um pouco sobre programação orientada ao objeto (POO) e Programação Orientada ao Aspecto (POA).

Spring.Net é um conjunto de instruções comuns que tem como objetivo fornecer soluções para diversos problemas comuns em aplicações, ou seja, quando programamos em alguma linguagem, acabamos criando classes, métodos, entre outros que partilham de um problema, que muitas vezes demandam de muito tempo do desenvolvedor e criam um acoplamento muito grande entre si, o Spring.Net cria soluções inteligentes para esse problema, o framework engloba as instruções para esse problema comum que facilitam a vida do desenvolvedor.*(**Wikipédia: Framework**, 2014)*

O Spring.Net deve ser flexível e poder ser extensível, permitindo que novos recursos sejam implementados a ele, garantindo uma maior quantidade de instruções para prover as soluções necessárias para as mais diversas aplicações.*(**Universidade Federal de Campina Grande: O que é Frameworks**)*

O conceito de paradigma de orientação ao objeto parte do princípio de fazermos uma análise cognitiva de como uma classe se comportaria caso existisse, ou seja, quais seus atributos, seu comportamento, com quais outros objetos ele pode/deve interagir, esse conceito é amplamente utilizado em diversas linguagens de programação, tais como: Java, C++, C#, VB.Net, entre outros.*(**Wikipédia: Orientação a Objetos**, 2014)*

Apesar da programação OO ter trago muitos conceitos que facilitam o desenvolvedor, entender e modelar o comportamento de cada classe ainda é algo difícil de ser feita, por isso surgiu diversas frameworks, como o Spring.Net, fazendo uso da OO, ele traz consigo o conceito de POA (Programação orientada ao Aspecto), esse conceito tem como objetivo fazer uma análise dos aspectos de uma classe.

Quando estamos desenvolvendo alguma aplicação é comum criarmos classes que possuem tarefas comuns entre si, por exemplo, podemos ter diversas classes que precisem acessar as informações de uma determinado arquivo, então teríamos que criar linhas de código para executar essa tarefa, porém o Spring.Net é utilizado justamente para analisar esse aspecto da nossa aplicação e encapsular esse comando nele mesmo, ou seja, ao invés de termos diversas classes fazendo a mesma coisa, podemos fazer com que as classes façam referência ao framework, que por sua vez fica encarregado de executar essa rotina para todas as classes. *(**Wikipédia: Programação Orientada a Aspecto**, 2013)*

É interessante analisar que o Spring.Net faz uso dos padrões do Inversão de Controle (IoF) e Injeção de Dependência.

Inversão de controle é onde a chamada de métodos é delegado a um container, ou seja, ele determina qual método será chamado dependente do que está sendo solicitado, na programação tradicional quem determina essa chamada é o próprio programador no momento do desenvolvimento.*(**Wikipédia: Inversão de Controle**, 2013)*

Injeção de dependência é o nome dado para quando o container fica responsável por declarar o nível de acoplamento entre os diversos módulos

da aplicação, na programão padrão o próprio desenvolvedor determina qual módulo depende de qual e qual o grau do acoplamento, a injeção de dependência é uma técnica muito utilizado quando não se quer ter um acoplamento muito grande e desnecessário entre os diferentes módulos da aplicação.*(**Wikipédia: Injeção de dependência**, 2013)*

O Spring.Net tem como base de sua arquitetura POJO's e Interfaces:

POJO são classes simples em Java, ela possui basicamente seus getters e setters e não dependem de interfaces e de outras frameworks.*(**GUJ: O que é POJO, afinal?**, 2002)(**Wikipédia: Plain Old Java Objects**, 2013)*

O conceito de interface parte do princípio que diferentes classes poderem executar uma mesma operação de formas diferentes, o framework obriga a classe a executar determinada tarefa, porém a forma como será feito varia para cada classe. *(**GUJ: Interfaces,**2002)(**Caelun: Interfaces**, 20-?.)(**Wikipédia: Interface (Ciência da Computação)**, 2013)*

A framework possui diversos módulos que facilitam o desenvolvimento da aplicação, alguns deles são:

Spring.Core: É utilizada na gerência do tipo de objeto, através desse módulo a configuração da injeção de dependência fica mais fácil de ser feito.*(**Spring Framework: Transaction management**, 20-?)*

Spring.AOP: Esse módulo permite dizer o grau de acoplamento entre as classes, esse módulo tem como objetivo centralizar os métodos em comum da aplicação.*(**Spring Framework: Aspect Oriented Programming with Spring.NET**, 20-?.)*

Spring.Data: Facilita a persistência de dados, graças a esse módulo o Spring.Net consegue trabalhar de forma confiável com diversos tecnologias, além de dar suporte a interfaces DAO (Data Acess Object), que consistem em fazer com que partes diferentes da aplicação não enxerguem a outra, não interferindo no acesso ao banco de dados, diferentes partes da aplicação podem acessar o banco de dados, uma pode estar atualizando, outra visualizando, ou deletando, por exemplo.*(**Wikipédia: Injeção de dependência**) (**Spring Framework: DAO support**, 20-?.)(**Wikipédia: Objeto de Acesso a Dados)***

Spring.Data.Nhibernate: Este módulo tem como objetivo a integração como o Nhibernate através do gerenciamento feito usando o módulo Spring.Data, graças a ele a integração é mais fácil de ser feita.*(**Spring Framework: DAO support**, 20-?.)*

Spring.Testing.Nunit: Módulo utilizado para rodar testes de integridade utilizando a ferramenta Nunit.

Spring.Testing.Microsoft: Faz teste na integração assim como o Spring.Testing.Nunit, porém ele é utilizado na ferramenta MSTEST, ferramenta de testa da microsoft.*(**Spring Framework: Testing**)*

Spring.Scheduling.Quartz: Integra o sistema com a ferramenta agendador de tarefas do Quartz.Net, ele ajuda na criação de rotinas pré-agendadas, que são executadas mediante uma determinada condição, estabelecidas pelo desenvolvedor.

*(**Spring Framework: Scheduling and Thread Pooling**, 20-?.)*