

O/RM(Object Relation Mapping) in MVC

Conf.dr. Cristian KEVORCHIAN

Facultatea de Matematică și Informatică

ck@fmi.unibuc.ro

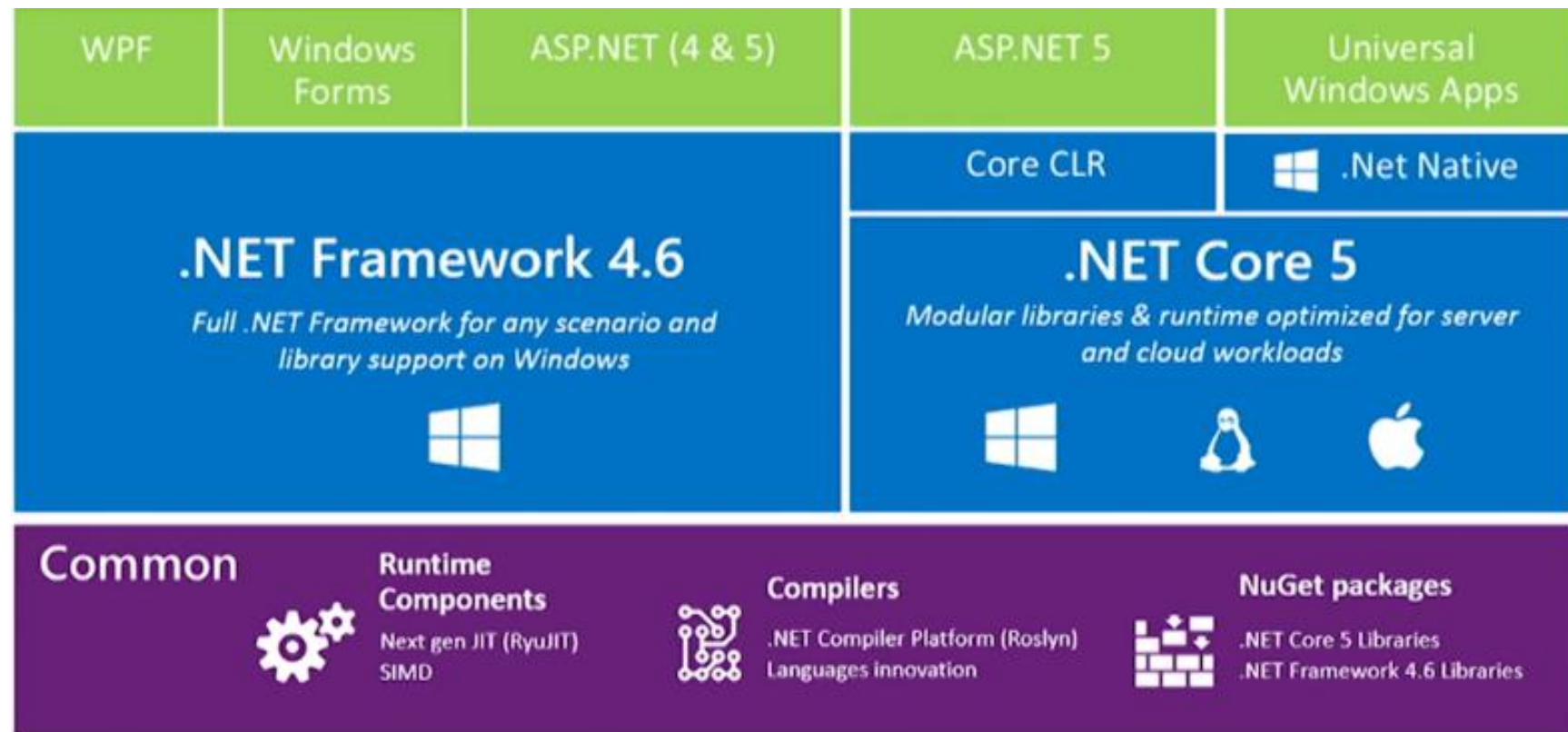
O/RM

O/RM este o tehnică de programare pentru conversia tipurilor de date ale SGBD incompatibile cu tipurile de date implementate de limbajele de programare orientate pe obiecte. Practic este create o "bază de date virtuală a obiectelor" care poate fi utilizată de limbajul de programare OO.

Exemple: EntityFramework, EF Core, Hibernate(implementare de JPA-ORM Hibenate, Nhibernate, Java ORM Apache Cayenne

O/RM pentru noSQL – EF Core, Hibernate OGM(Object Grid Mapper)-JPA

.NET Framework vs. .NET Core



- .NET Framework 4.6 este un framework general pentru platforme .NET, dar nu oferă biblioteci și funcții de runtime optimizate pentru lucrul multi-platăformă (Windows, Linux și Mac) și cloud
- .NET Core, pe de altă parte, este o componentă a platformei .NET Framework 4.6, care este în principal optimizată pentru implementări multi-platăformă și cloud
- .NET Native este folosit în principal pentru dezvoltarea de aplicații universale care vor fi aplicații desktop operaționale pe diverse dispozitive și platforme, nativ (nu pentru aplicațiile web)

Entity- Framework

- Utilizarea Entity Framework Code First drept model de date.
- EF Code First permite generarea de tabele dintr-o baza de date prin
- intermediul obiectelor POCO(Plain Old CLR Objects)
- EF permite utilizarea LINQ(Language Integrated Query) în
- operarea cu entități si expresii Lambda
- ADO.NET Framework este un ORM(Object Relation Mapping)
- integrat Entity Framework

EF este orientat entitate

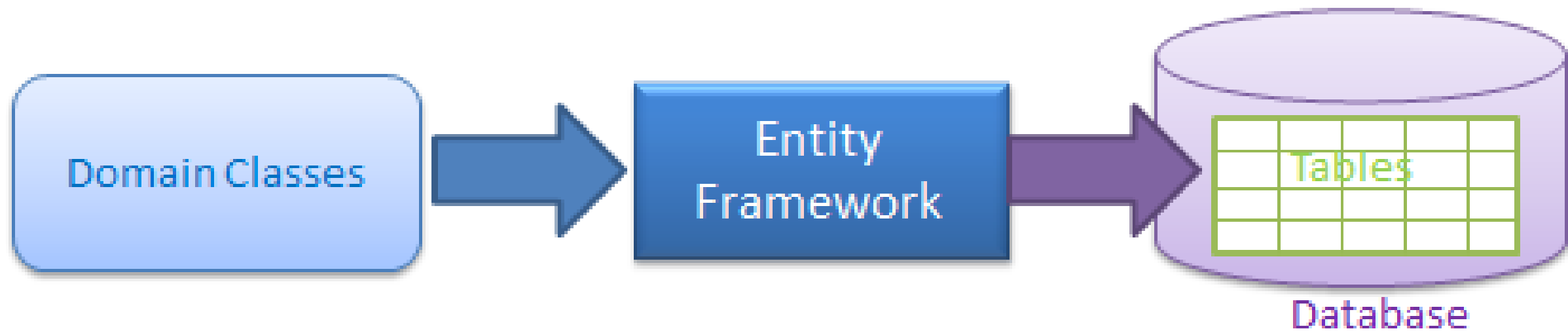
- Când utilizăm EF operăm cu un model entitate în locul modelului bazei de date relaționale.
- Această abstracție ne permite o mai bună orchestrare a entităților și relațiilor dintre acestea ce definesc logica de business
- "Data context" asociat EF pentru a defini interogările în contextul familiei de entități și a relațiilor dintre acestea.
- Când una din operațiile CRUD este invocată, EF va genera secvența SQL asociată operației.

EF Code-First

- EF a introdus abordarea bazata pe Code - First începând cu versiunea 4.1 a EF.
- DDD(Domain-Driven Design) reprezintă o formă de abordare a proiectării soluțiilor software de mare complexitate. Reprezinta un set de bune practici în rezolvarea problemelor complexe.
- Concept extrem de util în DDD(Domain Driven Design) .
- Dezvoltatorii se pot concentra pe design în sensul proiectării de clase ca alternativa la proiectarea metabazei.
- API-urile pentru Code-First conduc la crearea programată a bazei de date.

DDD(Domain Driven Design)

- Colecție de principii și pattern-uri care ajută programatorii să dezvolte sisteme de obiecte în dialect OO de o manieră foarte elegantă.
- Corect aplicate pot conduce la abstractizarea pieselor software dezvoltate într-un context "domain models".
- **Modelele încapsulează logică de business**
- Minimizează distanța între realitatea de business și codul dezvoltat.
- Modelul lui Platon



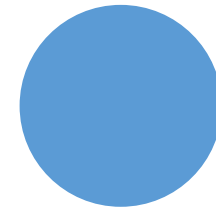
Schema de Abordarea pentru Code-First

- Convenția este un set de reguli utilizat în configurarea automată a modelului conceptual bazat pe definirea claselor din domeniu, prin intermediul namespace-ului:

System.Data.Entity.ModelConfiguration.Conventions

- Convenții
 1. Type Discovery
 2. Primary Key
 3. Relationship
 4. Foreign Key
 5. Complex Type
-

Conventii pentru Code-First



Tipuri de date C#	Corespondentul Tipului de date in SQL Server
int	int
string	nvarchar(Max)
decimal	decimal(18,2)
float	real
byte[]	varbinary(Max)
datetime	datetime
bool	bit
byte	tinyint
short	smallint
long	bigint
double	float
char	No mapping
sbyte	No mapping (throws exception)
object	No mapping

Maparea tipuri de date C# - SQL Server

Conventii de mapare

Sursa: <http://www.entityframeworktutorial.net/code-first/code-first-conventions.aspx>

```
public class Grade
{
    public int GradeId { get; set; }
    public string GradeName { get; set; }
    public string Section { get; set; }

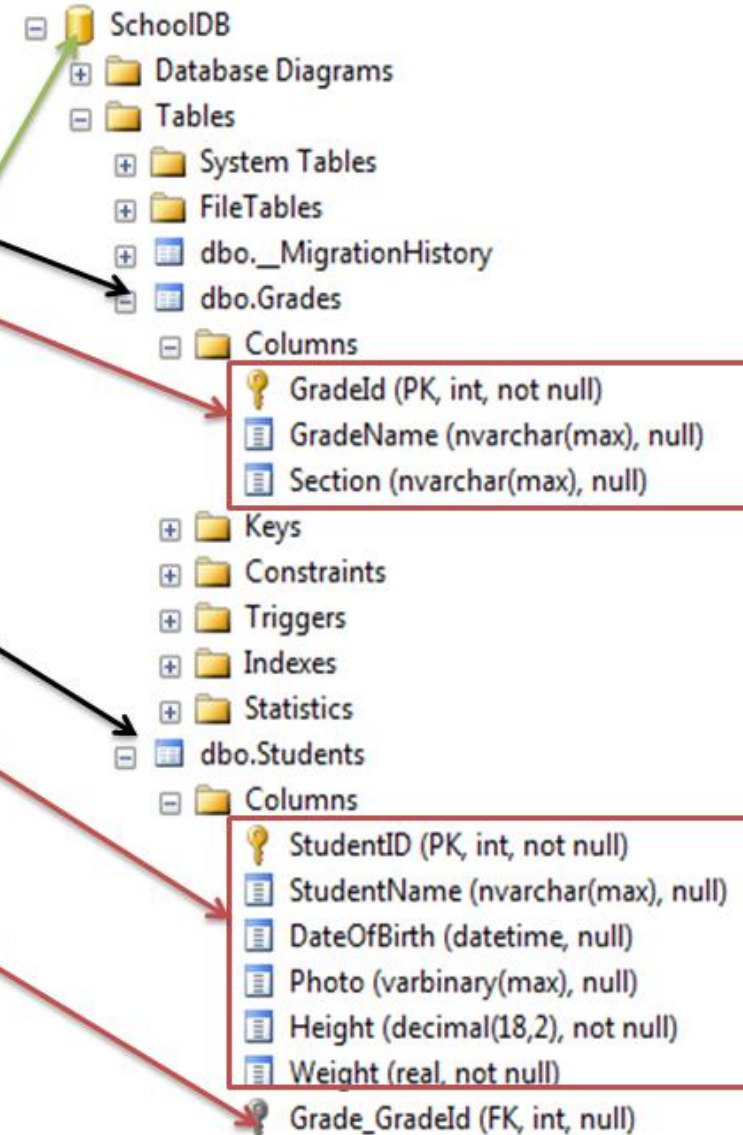
    public ICollection<Student> Students { get; set; }
}
```

```
public class Student
{
    public int StudentID { get; set; }
    public string StudentName { get; set; }
    public DateTime? DateOfBirth { get; set; }
    public byte[] Photo { get; set; }
    public decimal Height { get; set; }
    public float Weight { get; set; }

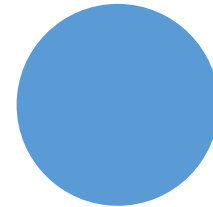
    public Grade Grade { get; set; }
}
```

```
public class SchoolContext : DbContext
{
    public SchoolContext() : base("SchoolDB")
    {
    }

    public DbSet<Student> Students { get; set; }
    public DbSet<Grade> Grades { get; set; }
}
```

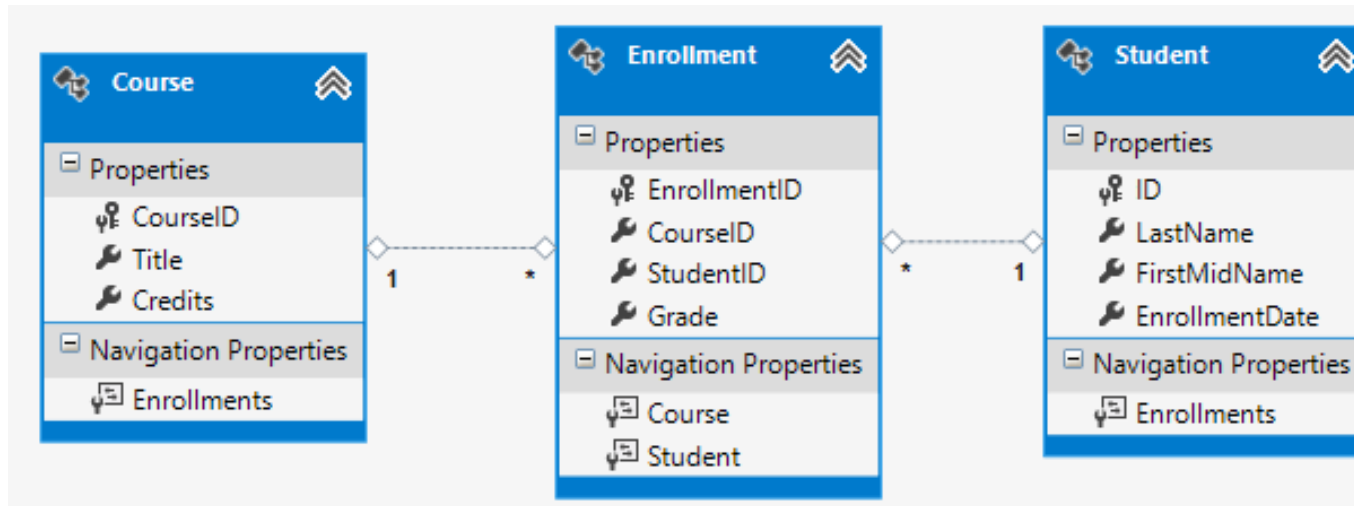


**Code-First construiește modelul conceptual
pe baza claselor din domeniu prin
utilizarea convențiilor**



Crearea Modelului de Date

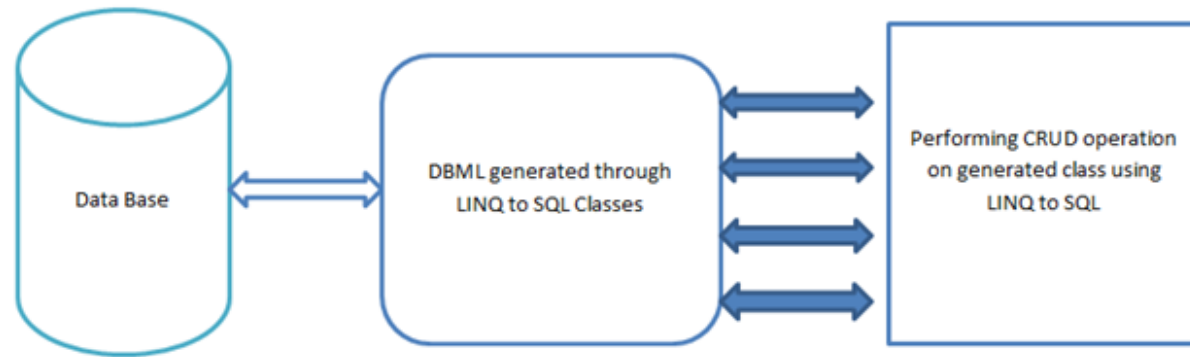
- Crearea claselor asociate modelului ER.



- Crearea Contextului asociat Bazei de Date
 - Principala clasă care coordonează funcționalitățile Entity Framework asociate unui model de date este **clasa context** a bazei de date derivată din:


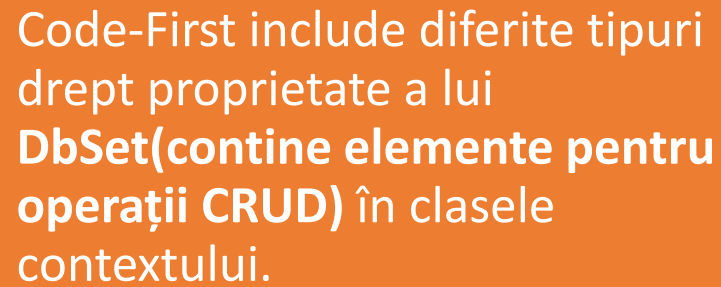
System.Data.Entity.DbContext

Exemplu LINQ to SQL(ORM)


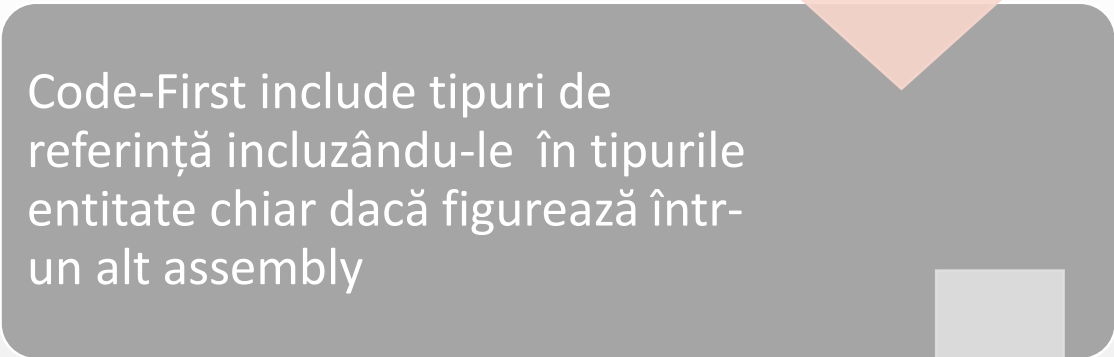


```
public void InterogareSimpla()
{
    DataClasses1DataContext dc = new DataClasses1DataContext();
    var q =
        from a in dc.GetTable<Order>()
        select a;
    dataGridView1.DataSource = q;
}
```

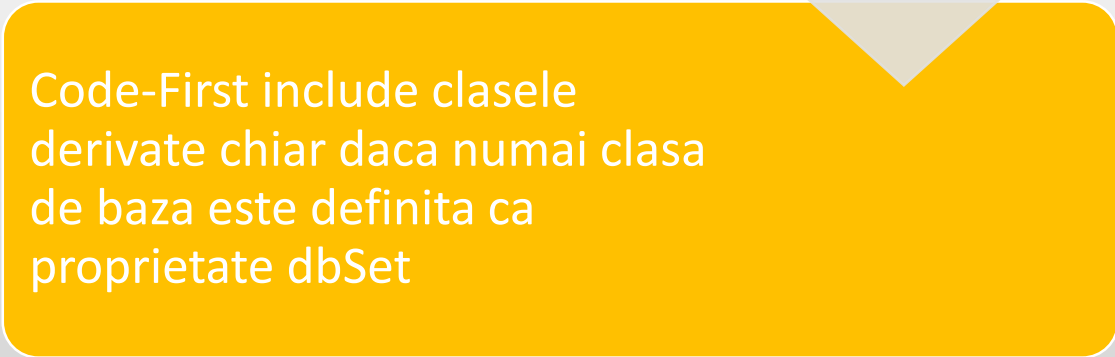
Code-First include diferite tipuri drept proprietate a lui **DbSet(contine elemente pentru operații CRUD)** în clasele contextului.



Code-First include tipuri de referință incluzându-le în tipurile entitate chiar dacă figurează într-un alt assembly



Code-First include clasele derivate chiar daca numai clasa de baza este definita ca proprietate dbSet



”Type Discovery”

Convențiile Primary Key și Relationship

- Code-First crează automat cheie primară dacă numele de proprietate este Id sau <nume clasă>Id(nu este case sensitive).
- Code-First induce o relatie între două entități utilizând proprietatea de navigare.
- Proprietatea de navigare poate fi un tip de referinta simpla sau un tip colectie.

Convențiile "Foreign Key" și "Complex Type"

- Code-First inserează automat un foreign key acolo unde întâlnește o proprietate de navigare.
- Code-First crează un Complex Type pentru clasele care nu includ proprietăți cheie sau primary key.

DataBase First

Entity Framework permite obținerea unui model de date cu clase și proprietăți asociate tabelelor și coloanelor unei baze de date existente.

Informațiile despre structura bazei de date (schema), modelul conceptual și asocierea între ele sunt furnizate prin intermediul unui fișier XML (.edmx).

Entity Framework Designer oferă o interfață grafică pentru afișarea și editarea fișierului .edmx.

Code First in Java

- Soluția recomandată este Spring Data JPA(Java Persistence API) este un ORM pentru managementul datelor expuse relațional.
 - Entități
 - Inferență de entități
 - Managementul entităților
 - Interogarea entităților
- Stiva de tehnologii
 - JDK 1.8 sau ulterioare
 - Gradle 2.3+ sau Maven 3.0+
- Putem importa cod în IDE-ul nostru:
 - [Spring Tool Suite \(STS\)](#)
 - [IntelliJ IDEA](#)

Instrumente de implementare

Dezvoltarea de aplicații web cu ajutorul Spring MVC presupune crearea mai multor nivele logice.

- DAO (Data Access Object Repository). Realizează comunicarea cu baza de date. Implică scrierea unui volum însemnat de cod.
- Spring Data automatizează această activitate.
- Spring Data JPA, Spring MVC, MySQL și Maven
- Hibernate este utilizat drept JPA.

Etapele de realizare implică utilizarea:

- Crearea unui proiect dinamic Maven în Eclipse.
- Spring MVC application
- Spring+Hibernate

vsChart.com, februarie 2018

	<u>ASP.NET MVC</u>	<u>Spring</u>
	ASP.NET MVC	Spring Framework
Category	<ul style="list-style-type: none">•<u>Web application framework</u>•<u>Framework</u>	<ul style="list-style-type: none">•<u>Web application framework</u>
Preference	51% votes	49% votes

Q&A