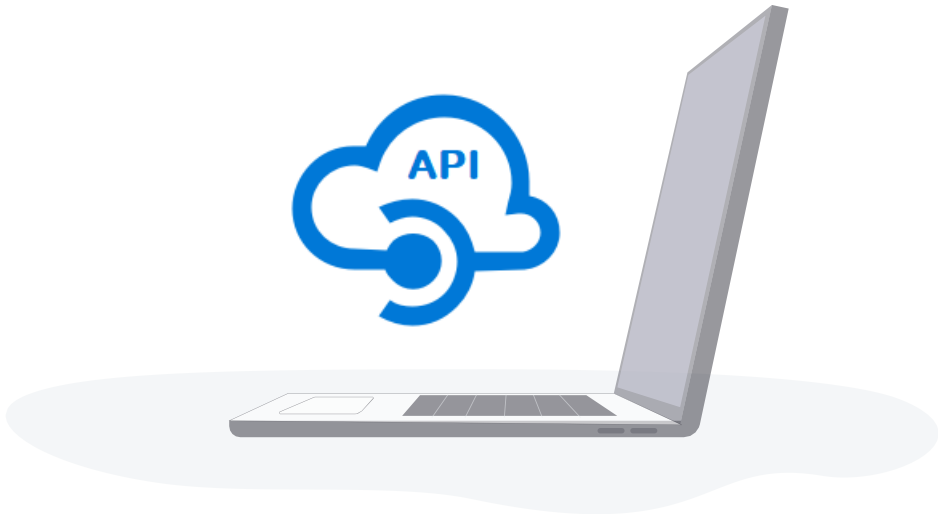
Harmony IT Solution

Web Application Programming Interface (API)

Tahaluf Training Center 2022

Harmony IT Solution

1. Overview of Data Transfer Object (DTOs)
2. Overview of Data Filtering
3. Create a Filter using DTO



Harmony IT Solution

Overview of Data Transfer Object (DTOs)

Data Transfer Objects or DTOs are objects that carry data between processes used to reduce the number of functions calls.

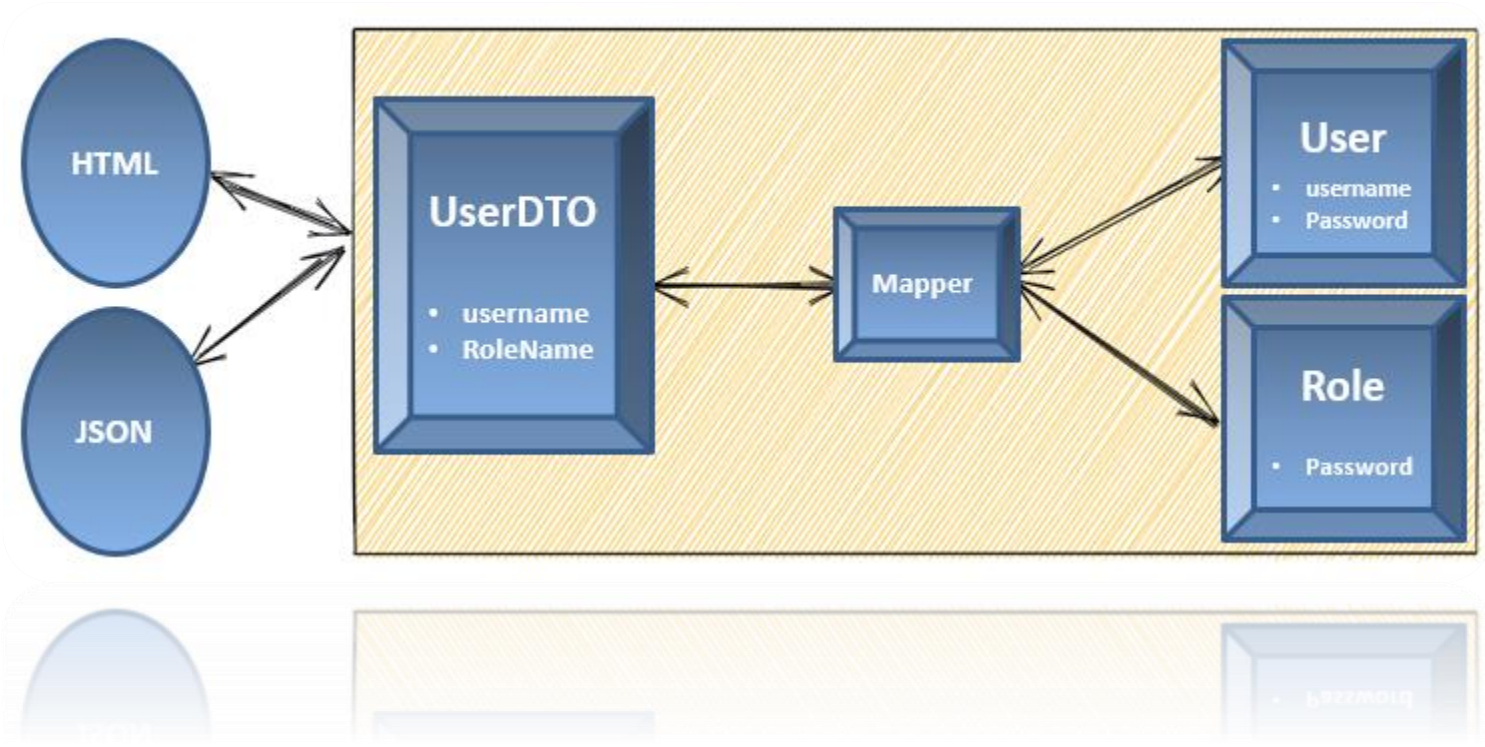
The pattern was first introduced by Martin Fowler EAA book. It is used to reduce the network overhead in such remote operations and encapsulate of the serialization's logic.

How to Use DTOs?

DTOs are created as POJOs. They **are flat data structures that contain no business or data logic.** They contain only accessors, storage, and methods related to parsing or serialization.

The data is mapped from the data access models to the DTOs, through a mapper part in the presentation layer.

The image illustrates the interaction between the components:



When to Use DTOs?

DTOs used in systems with remote calls, because they help to reduce the number of them and when the domain or data access model is composed of many various objects, and the presentation model needs all data at once or even reduces roundtrip between server and client.

When to Use DTOs?

DTOs used to build different views from our domain or data access models and allow to create other representations of the same domain, but optimizing them to the clients' needs without affecting our domain design.

Harmony IT Solution

Overview of Data Filtering

Data filtering can refer to a wide range of solutions or strategies for refining data sets.

The data sets are refined into simply what a user needs, without including other data that can be irrelevant, repetitive, or even sensitive.

Harmony IT Solution

Different types of data filters can be used to amend query, reports, results, or other kinds of information results.

Create a Filter using DTO

In stdcourse_Package package specification Add a SearchStudentAndCourse Stored Procedure:

PROCEDURE SearchStudentAndCourse(cName in varchar , sName in varchar , DateFrom in date , DateTo in date);

In stdcourse_Package package body Add a SearchStudentAndCourse Stored Procedure:

```
PROCEDURE SearchStudentAndCourse(cName in varchar , sName in varchar ,
DateFrom in date , DateTo in date)
As
Get_Cur SYS_REFCURSOR;
Begin
open Get_Cur for
select s.firstname , s.LastName , c.CourseName , sc.markofstd
from Student s
inner join stdCourse sc
```

In stdcourse_Package package body Add a SearchStudentAndCourse Stored Procedure:

```
on s.studentid = sc.stdid
inner join course c
on c.courseid = sc.courseid
where (upper(s.firstname) like '%'||upper(sName) ||'%') -- null
And ( upper( c.coursename) like '%' || upper( cname) || '%') -- S
And (DateFrom is null or DateTo is null or  sc.DateofRegister between DateFrom
and DateTo);
dbms_sql.return_result(Get_Cur);
End SearchStudentAndCourse;
```

Create a DTOs

Right Click on TahalufLearn.Core => Add New Folder => DTO.

Right Click on DTO => Add Class => Search.

Search DTO Code:

```
public class Search
        {
        public string Firstname { get; set; }
        public string Lastname { get; set; }
        public decimal? Markofstd { get; set; }
        public string Coursename { get; set; }
        public DateTime? DateFrom { get; set; }
        public DateTime? DateTo { get; set; }

        }
```

```
public class Search
    {
    1 reference
    public string Firstname { get; set; }
    0 references
    public string Lastname { get; set; }
    0 references
    public decimal? Markofstd { get; set; }
    1 reference
    public string Coursename { get; set; }
    1 reference
    public DateTime? DateFrom { get; set; }
    1 reference
    public DateTime? DateTo { get; set; }

    }
```

In TahalufLearn.Core => Repository => IStudentCourseRepository add the following abstract method:

```
List<Search> SearcheStudenCourse(Search search);
```

In TahalufLearn.Infra => Repository => StudentCourseRepository add the following method:

```
 public List<Search> SearcheStudenCourse(Search search)
            {
            var p = new DynamicParameters();
            p.Add("sName", search.Firstname, dbType: DbType.String,
direction: ParameterDirection.Input);
            p.Add("DateFrom", search.DateFrom, dbType:
DbType.DateTime, direction: ParameterDirection.Input);
            p.Add("DateTo", search.DateTo, dbType: DbType.DateTime,
direction: ParameterDirection.Input);
            p.Add("cName", search.Coursename, dbType:
DbType.String, direction: ParameterDirection.Input);
            var result =
dBContext.Connection.Query<Search>("stdcourse_Package.SearchStudent
AndCourse", p, commandType: CommandType.StoredProcedure);
            return result.ToList();
            }
```

In TahalufLearn.Core => Service => IStudentCourseService add the following abstract methods:

```
List<Search> SearcheStudenCourse(Search search);
```

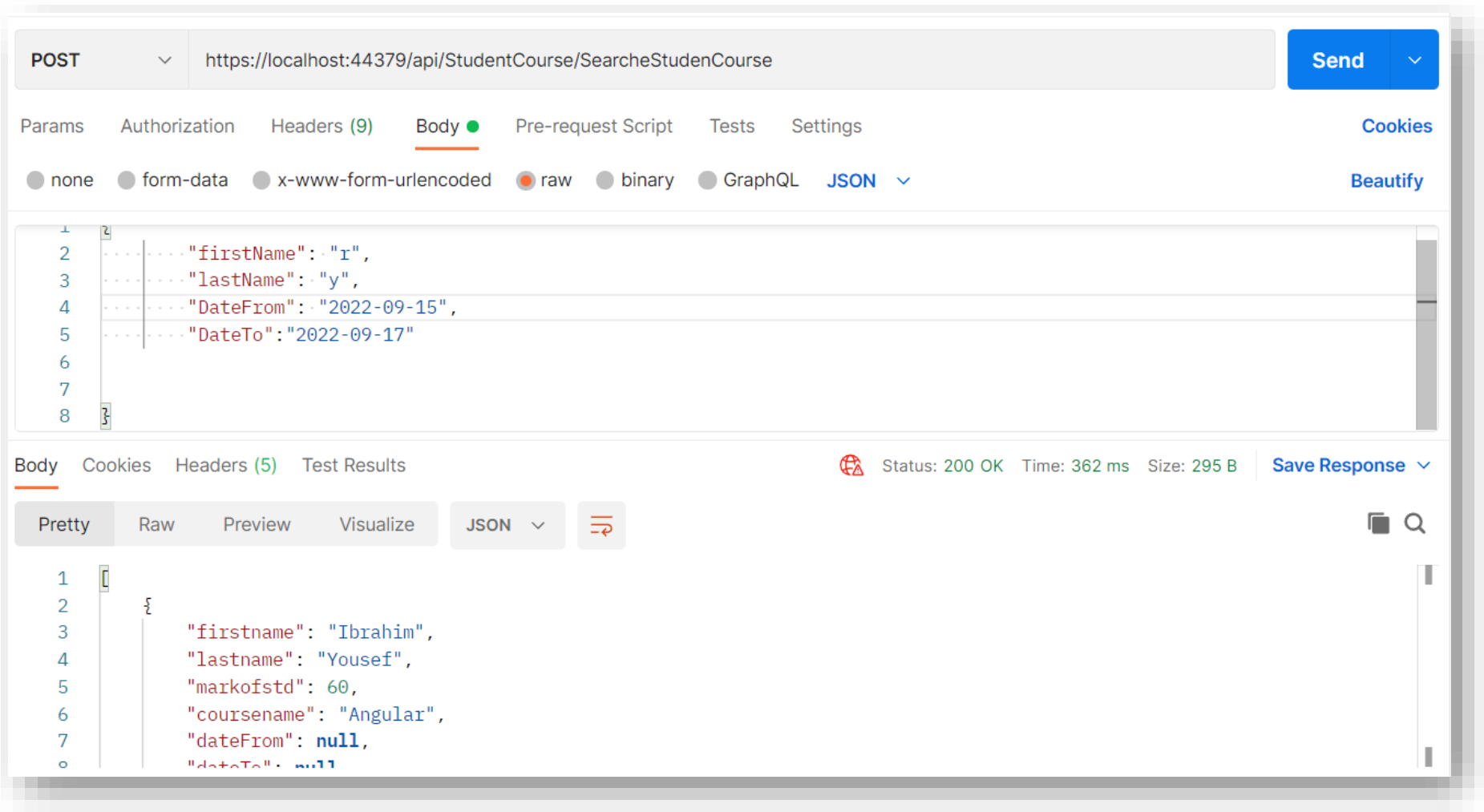
In TahalufLearn.Infra => Service => StudentCourseService add the following method:

```
public List<Search> SearcheStudenCourse(Search search)
            {
           return _studentCourseRepository.SearcheStudenCourse(search);
            }
```

In TahalufLearn.API => Controler => StudentCourseController add the following method:

```
[HttpPost]
        [Route("SearcheStudenCourse")]
        public List<Search> SearcheStudenCourse(Search search)
            {
            return
_studentCourseService.SearcheStudenCourse(search);
            }
```

Harmony IT Solution



Exercise

Create a function to retrieve the total number of students in each course.

In stdcourse_Package package specification Add a TotalStudentInEachCourse Stored Procedure:

PROCEDURE TotalStudentInEachCourse

In stdcourse_Package package body Add a TotalStudentInEachCourse Stored Procedure:

```
PROCEDURE TotalStudentInEachCourse
As
C_all SYS_REFCURSOR ;
Begin
open C_all for
select c.coursename , count(s.studentid) as StudentCount
from stdcourse sc
full outer join student s
```

In stdcourse_Package package body Add a TotalStudentInEachCourse Stored Procedure:

```
on s.studentid = sc.stdid
full outer join course c
on c.courseid = sc.courseid
group by c.coursename ;
Dbms_sql.return_result(c_all);
End TotalStudentInEachCourse ;
```

Create a DTOs

Right Click on DTO => Add Class => TotalStudents.

```
public class TotalStudents
        {
        public string CourseName { get; set; }
        public decimal? StudentCount { get; set; }
        }
```

In TahalufLearn.Core => Repository => IStudentCourseRepository add the following abstract method:

```
public List<TotalStudents> TotalStudentInEachCourse();
```

In TahalufLearn.Infra => Repository => StudentCourseRepository add the following method:

```
public List<TotalStudents> TotalStudentInEachCourse()
            {
            var result =
dBContext.Connection.Query<TotalStudents>("stdcourse_Package.TotalS
tudentInEachCourse", commandType: CommandType.StoredProcedure);
            return result.ToList();
            }
```

In TahalufLearn.Core => Service => IStudentCourseService add the following abstract methods:

```
public List<TotalStudents> TotalStudentInEachCourse();
```

In TahalufLearn.Infra => Service => StudentCourseService add the following method:

```
public List<TotalStudents> TotalStudentInEachCourse()
            {
            return
_studentCourseRepository.TotalStudentInEachCourse();
            }
```

In TahalufLearn.API => Controler => StudentCourseController add the following method:

```
[HttpGet]
        [Route("TotalStudentInEachCourse")]
        public List<TotalStudents> TotalStudentInEachCourse()
            {
            return
_studentCourseService.TotalStudentInEachCourse();
            }
```

Harmony IT Solution