Postupak u kojem se ponovo pokreću testovi kako bi se provjerilo da novim provjerama nije narušen prethodno ispravni i testirani kod naziva se

a reevaluacija

b evaluacijsko testiranja

c reverzno testiranje

d aspektno testiranje

e regresijsko testiranje

Što ispisuje sljedeći programski odsječak?

```
var data = GetData();
Console.WriteLine(data.First());
...
IEnumerable<string> GetData() {
  Console.WriteLine(1);
  yield return "A";
  Console.WriteLine(2);
  yield return "B";
  Console.WriteLine(3);
  yield return "C";
}
```

a
```
1
A
```

b
```
1
2
3
```

c
```
1
2
3
A
```

d
```
1
2
3
C
```

e
```
A
1
2
3
```

Ako je razred *Result* zadan s

```
public class Result {
    [JsonPropertyName("action")]
    public string ActionName { get; set; }
    [JsonPropertyName("status")]
    public bool ResultStatus { get; set; }
```

a u *Startup.cs* datoteci se nalazi

```
services...
        .AddJsonOptions(configure => configure.JsonSerializerOptions.PropertyNamingPolicy = null);
```

tada se objekt tipa *Result* s određenim vrijednostima za *ActionName* i *ResultStatus* automatski pretvora u JSON oblika:

a
```
{"action":"Run", "status" : true}
```

b
```
{"result@7ea1246f" : {"ActionName":"Run", "ResultStatus" : true}}
```

c
```
{"ActionName":"Run", "ResultStatus" : true}
```

d
```
{"actionName":"Run", "resultStatus" : true}
```

e
```
{"Action":"Run", "Status" : true}
```

Želimo li opisati moguće odgovore metode WebApi servisa koja treba vratiti 1 podatak (npr. Artikl) na osnovi vrijednosti primarnog ključa artikla, koji od sljedećih odsječaka je ispravan.

a

```
[ProducesResponseType ((int)HttpStatusCode.Ok )]
public async Task<IActionResult> Get(string sifArtikla )
```

b

```
[ProducesResponseType (typeof (IActionResult), (int)HttpStatusCode.OK )]
[ProducesResponseType ((int)HttpStatusCode.NotFound )]
public async Task<IActionResult> Get(string sifArtikla )
```

c

```
[ProducesResponseType (typeof (Artikl), (int )HttpStatusCode.OK )]
[ProducesResponseType ((int)HttpStatusCode.NotFound )]
public async Task<IActionResult> Get(string sifArtikla )
```

d

```
[ProducesResponseType ((int)HttpStatusCode.NotFound )]
public async Task<IActionResult> Get(string sifArtikla )
```

e

```
[ProducesResponseType ((int)HttpStatusCode.Ok )]
[ProducesResponseType ((int)HttpStatusCode.NotFound )]
public async Task<IActionResult> Get(string sifArtikla )
```

Pretpostavimo da **integracijskim testiranjem** želimo provjeriti preusmjerava li upravljač *Drzava* s akcije *Index* na *Create* za situaciju koja je podešena u testu (npr. prazna baza podataka) te uz pretpostavljene parametre usmjeravanja. Kojim od programskih odsječaka to možemo obaviti?

a

```
var controller = new DrzavaController(...
...
var client = factory.CreateClient(new WebApplicationFactoryClientOptions {
                            AllowAutoRedirect = false
                        });
var response = await client.GetAsync(controller.Url);
Assert.Equal(HttpStatusCode.Redirect, response.StatusCode);
Assert.Equal(nameof(Controller.Create), response.Headers.Location);
```

b

```
string url = "/Drzava/Index";
var client = factory.CreateClient(new WebApplicationFactoryClientOptions {
                            AllowAutoRedirect = false
                        });
var response = await client.GetAsync(url);
Assert.Equal(HttpStatusCode.Redirect, response.StatusCode);
Assert.Equal("/Drzava/Create", response.Headers.Location);
```

c

Navedeno se ne može provjeriti integracijskim testiranjem

d

```
var controller = new DrzavaController(...
...
var result = controller.Index();
...
var redirectToActionResult =  Assert.IsType<RedirectToActionResult>(result);
Assert.Equal("Create", redirectToActionResult.ActionName);
```

e

```
var controller = new DrzavaController(...
...
var result = controller.Index();
...
var actionResult =  Assert.IsType<IActionResult>(result);
```

Neka se na adresama `https://server/app/mjesto` i `https://server/app/drzava` nalaze web-servisi za rad s mjestima te neka za manipulaciju podacima koriste GET, POST, PUT i DELETE metode.

Prilikom dohvata svim mjesta ili država web-servis vraća popis mjesta i država bez dodatnih informacija.

Prilikom dodavanja novog podatka korisniku se vraća status 201 uz pohranjeni podatak i adresu koja jednoznačno određuje dodani podatak.

Koji nivo Richardsonovog modela zrelosti zadovoljava ovaj web-servis?

a PATCH

b 1

c GET + POST

d 3

e 2

Razred kojim se pogreške u web-servisima opisuju u skladu s RFC-om 7807 zove se

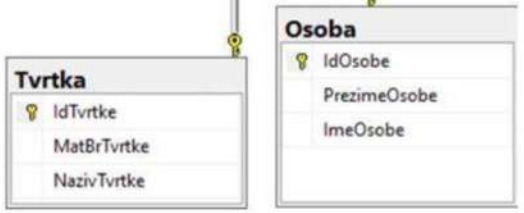
a ProblemDetails

b NotFound

c BadRequest

d ActionResult

e ExceptionFilter



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Što će se dogoditi izvršavanje programskog koda opisanog sljedećom skicom?

Napomena: *ctx* je konkretni EF kontekst, a vrijednosti s desne strane navedenih izraza su ispravne, ali izostavljene zbog jednostavnosti. IdPartnera je tipa *identity* i sve trenutne vrijednosti u bazi podataka za IdPartnera su 1 ili više.

```
Osoba o = new Osoba();
o.PrezimeOsobe = ...
o.ImeOsobe = ...
ctx.Add(o);
ctx.SaveChanges();
```

a Izvršit će se dva INSERT upita, prvo INSERT u tablicu Osoba kako bi se dobio PK, koji će onda biti korišten prilikom INSERTa u tablicu Partner kao PK, pri čemu će ostale vrijednosti u tablici Partner biti null.

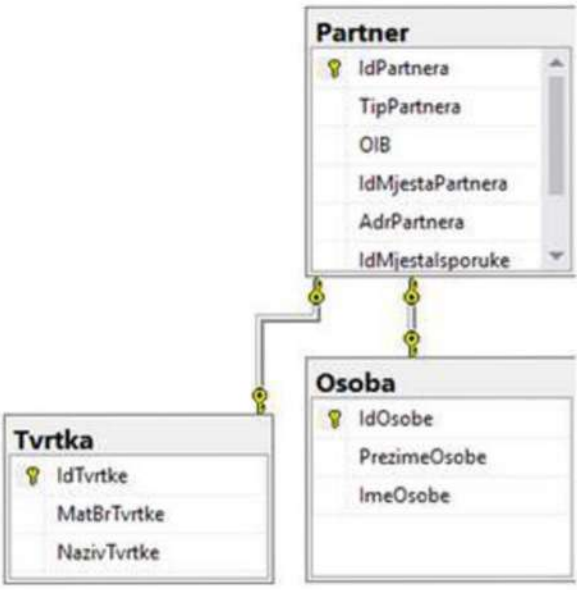
b Izvršavanjem neće doći do iznimke, ali se neće dogoditi niti jedan INSERT upit.

c Izvršit će se dva INSERT upita, prvo INSERT u tablicu Partner koristeći null za sve vrijednosti osim primarnog ključa koji će biti automatski generiran, a zatim INSERT i tablicu Osoba s vrijednosti PK koja odgovara generiranom PK prilikom unosa u tablicu Partner.

d Izvršit će se samo jedan INSERT upit i to u tablicu Osoba.

e Izvršavanje izaziva iznimku jer se osoba ne može dodati bez dodavanja postojećeg partnera.

Slika predstavlja primjer specijalizacije i generalizacije napravljen po principu



a table per hierarchy

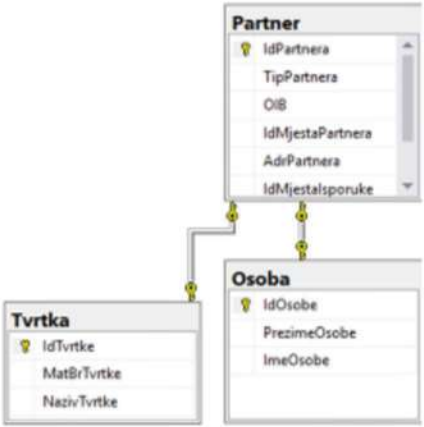
b table per foreign key

c table per concrete class

d table per primary key

e table per type

Neka se u relacijskoj bazi podataka nalaze tablice s odnosima kao na sljedećoj slici, odnosno u skladu s primjerima na predavanjima.



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Za dodavanje novog partnera, koji može biti osoba ili tvrtka, koristi se prezentacijski model *PartnerViewModel*. Na koji od navedenih atributa treba dodati validacijski atribut `[Required]`?

```
public class PartnerViewModel {
   public int IdPartnera { get; set; }
   public string TipPartnera { get; set; }
   public string PrezimeOsobe { get; set; }
   public string ImeOsobe { get; set; }
   public string MatBrTvrtke { get; set; }
   public string NazivTvrtke { get; set; }
   public string Oib { get; set; }
   public string AdrPartnera { get; set; }
   public int? IdMjestaPartnera { get; set; }
   public string NazMjestaPartnera { get; set; }
```

a `PrezimeOsobe`

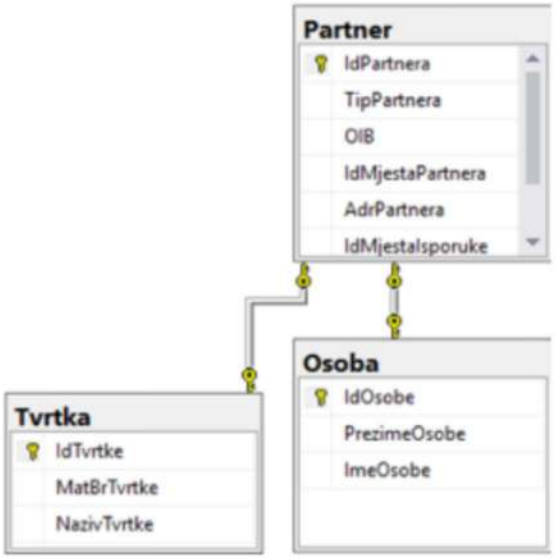
b `IdPartnera`

c `MatBrTvrtke`

d `ImeOsobe`

e `OIB`

Neka se u relacijskoj bazi podataka nalaze tablice s odnosima kao na sljedećoj slici, odnosno u skladu s primjerima na predavanjima.



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Što će se dogoditi izvršavanje programskog koda opisanog sljedećom skicom?

Napomena: *ctx* je konkretni EF kontekst, a vrijednosti s desne strane navedenih izraza su ispravne, ali izostavljene zbog jednostavnosti. IdPartnera je tipa *identity* i sve trenutne vrijednosti u bazi podataka za IdPartnera su 1 ili više.

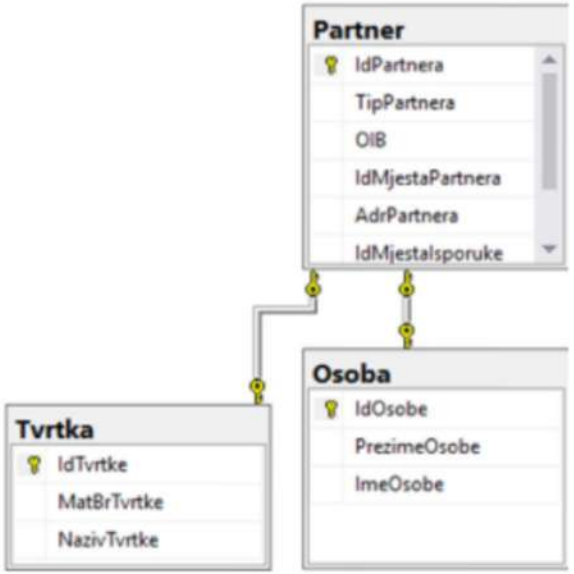
```
Partner p = new Partner();
p.AdrIsporuke = ...;
p.AdrPartnera = ...;
p.IdMjestaIsporuke = ...;
p.IdMjestaPartnera = ...;
p.Oib = ...;
ctx.Add(p);
ctx.SaveChanges();
```

a Izvršit će se samo jedan INSERT upit i to u tablicu Partner.

b Izvršavanjem neće doći do iznimke, ali se neće dogoditi niti jedan INSERT upit.

c Upit se ne može izvršiti (tj. izaziva iznimku prilikom izvršavanja), jer za Partnera nije postavljena osoba ili tvrtka.

Neka se u relacijskoj bazi podataka nalaze tablice s odnosima kao na sljedećoj slici, odnosno u skladu s primjerima na predavanjima.



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Što će se dogoditi izvršavanje programskog koda opisanog sljedećom skicom?

Napomena: *ctx* je konkretni EF kontekst, a vrijednosti s desne strane navedenih izraza su ispravne, ali izostavljene zbog jednostavnosti. IdPartnera je tipa *identity* i sve trenutne vrijednosti u bazi podataka za IdPartnera su 1 ili više.

```
Partner p = ctx.Find(...
p.OIB = ...
p.Osoba = new Osoba { ImeOsobe = ..., PrezimeOsobe = ... };
ctx.SaveChanges();
```

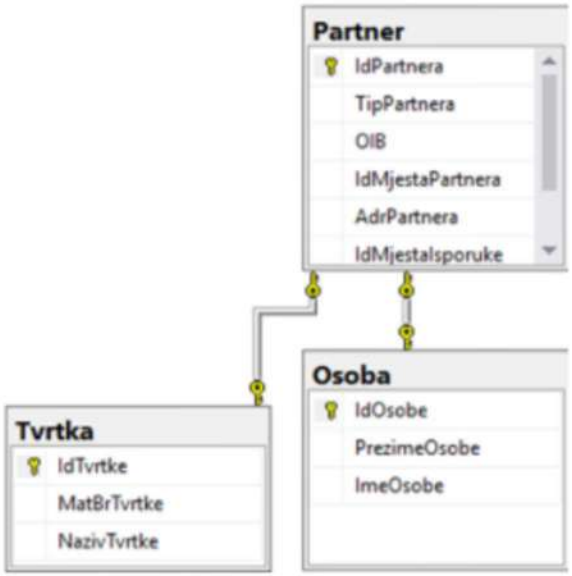
a Izvršit će se jedan UPDATE upit i jedan DELETE upit. UPDATE mijenja OIB partnera, a DELETE briše osoba koja je prethodno imala IdOsobe koji je odgovarao IdPartnera. Veza Osobe i Partnera je uklonjena zbog new p.Osoba = , a ta osoba nije dodana u kontekst.

b Pokušat će se izvršiti INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner, ali će se dogoditi iznimka jer se za IdOsobe koristi vrijednost 0.

c Izvršit će se INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner na način da se promijeni OIB i IdPartnera u skladu s dobivenom vrijednosti primarnog ključa u tablici Osoba.

d Izvršit će se samo jedan UPDATE upit te će se promijeniti samo OIB dohvaćenog partnera.

Neka se u relacijskoj bazi podataka nalaze tablice s odnosima kao na sljedećoj slici, odnosno u skladu s primjerima na predavanjima.



Temeljem navedenih odnosa, EF Core je stvorio model s 3 istoimena entiteta. Što će se dogoditi izvršavanje programskog koda opisanog sljedećom skicom?

Napomena: *ctx* je konkretni EF kontekst, a vrijednosti s desne strane navedenih izraza su ispravne, ali izostavljene zbog jednostavnosti. IdPartnera je tipa *identity* i sve trenutne vrijednosti u bazi podataka za IdPartnera su 1 ili više.

```
Partner p = ctx.Find(...
p.OIB = ...
p.Osoba = nev Osoba { IdOsobe = p.IdPartnera, ImeOsobe = ..., PrezimeOsobe = ... };
ctx.Entry(partner.Osoba).State = EntityState.Modified;
ctx.SaveChanges();
```

a Izvršit će se samo jedan UPDATE upit te će se promijeniti samo OIB dohvaćenog partnera.

b Izvršit će se INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner na način da se promijeni OIB. Iznimka se neće dogoditi ako već nije postojala osoba s istom vrijednosti primarnog ključa.

c Izvršit će se 2 UPDATE upita: jedan za promjenu Osobe koja ima IdOsobe jednak IdPartnera partnera kojem mijenjamo OIB.

d Pokušat će se izvršiti INSERT upit u tablicu Osoba i UPDATE upit nad tablicom Partner, ali će se dogoditi iznimka jer se za IdOsobe koristi vrijednost postojećeg partnera.

Koju od navedenih kombinacija treba upotrijebiti u EF kontekstu da bi mogli izvršiti naredbu

```
var list = ctx.ViewPribor.ToList();
```

ako je u EF modelu definiran razred ViewPribor koji svojim oblikom prati rezultat pogleda vw_Pribor iz baze podataka?

a

```
public virtual DbSet<ViewPribor> ViewPribor { get; set; }
...
partial void OnModelCreatingPartial(ModelBuilder modelBuilder)  {
      modelBuilder.Entity<ViewPribor>(entity => {
        entity.HasNoKey();
        entity.ToView("vw_Pribor");
      });
```

b

```
public virtual DbSet<vw_Pribor> ViewPribor { get; set; }
...
partial void OnModelCreatingPartial(ModelBuilder modelBuilder)  {
      modelBuilder.Entity<vw_Pribor>(entity => {
        entity.HasNoKey();
      });
```

c

```
public virtual DbSet<vw_Pribor> ViewPribor { get; set; }
...
partial void OnModelCreatingPartial(ModelBuilder modelBuilder)  {
      modelBuilder.Entity<ViewPribor>(entity => {
        entity.HasNoKey();
        entity.ToView("vw_Pribor");
      });
```

d

```
public virtual DbSet<ViewPribor> vw_Pribor { get; set; }
...
partial void OnModelCreatingPartial(ModelBuilder modelBuilder)  {
      modelBuilder.Entity<ViewPribor>(entity => {
```

Koju od navedenih kombinacija treba upotrijebiti u EF kontekstu da bi mogli izvršiti naredbu

```
var list = ctx.vw_Pribor.ToList();
```

ako je u EF modelu definiran razred ViewPribor koji svojim oblikom prati rezultat pogleda vw_P

a

```
public virtual DbSet<vw_Pribor> ViewPribor { get; set; }
...
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {
      modelBuilder.Entity<ViewPribor>(entity => {
        entity.HasNoKey();
        entity.ToView("vw_Pribor");
      });
```

b

```
public virtual DbSet<ViewPribor> vw_Pribor { get; set; }
...
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {
      modelBuilder.Entity<ViewPribor>(entity => {
        entity.HasNoKey();
      });
```

c

```
public virtual DbSet<ViewPribor> vw_Pribor { get; set; }
...
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {
      modelBuilder.Entity<ViewPribor>(entity => {
        entity.HasNoKey();
        entity.ToView("ViewPribor");
      });
```

d

```
public virtual DbSet<vw_Pribor> ViewPribor { get; set; }
...
partial void OnModelCreatingPartial(ModelBuilder modelBuilder) {
      modelBuilder.Entity<vw_Pribor>(entity => {
```

Ako je razred *Result* zadan s

```
public class Result {
    [JsonPropertyName("action")]
    public string ActionName { get; set; }
    [JsonPropertyName("status")]
    public bool ResultStatus { get; set; }
```

a u *Startup.cs* datoteci se nalazi

```
services...
        .AddJsonOptions(configure => configure.JsonSerializerOptions.PropertyNamingPolicy = JsonNamingPolicy.CamelCase);
```

tada se objekt tipa *Result* s određenim vrijednostima za *ActionName* i *ResultStatus* automatski pretvora u JSON oblika:

a `{"action":"Run", "status" : true}`

b `{"actionName":"Run", "resultStatus" : true}`

c `{"result@7ea1246f" : {"ActionName":"Run", "ResultStatus" : true}}`

d `{"Action":"Run", "Status" : true}`

e `{"ActionName":"Run", "ResultStatus" : true}`

Ako je razred *Result* zadan s

```
public class Result {
    public string ActionName { get; set; }
    public bool ResultStatus { get; set; }
```

a u *Startup.cs* nije ekspclicitno ništa navedeno oko serijalizacijskih opcija

tada se objekt tipa *Result* s određenim vrijednostima za *ActionName* i *ResultStatus* automatski pretvora u JSON oblika:

a `{"ActionName":"Run", "ResultStatus" : true}`

b `{"Name":"Run", "Status" : true}`

c `{"result@7ea1246f" : {"ActionName":"Run", "ResultStatus" : true}}`

d `{"actionName":"Run", "resultStatus" : true}`

e `{"name":"Run", "status" : true}`

U pogledu ili glavnoj stranici potrebno je definirati varijablu koja sadrži putanju do aplikacije tako da se ta vrijendost može koristiti u skriptnim datoteka.

Npr. neka se akcija *Index* i upravljač *Home* nalazi na adresi `https://rppp.fer.hr:4443/42/Apps/20200316.2/Home/Index`. Tada je putanja `/42/Apps/20200316.2/`.

Koja od sljedećih naredbi u varijablu root posprema putanju aplikacije u odnosu na server?

a

```
<script>
root = '@Application.Root';
</script>
```

b

```
<script>
root = '@System.IO.Path.Get("~")';
</script>
```

c

```
<script>
root = '@Url.Content("~/")';
</script>
```

d

```
<script>
root = 'asp-action="~"';
</script>
```

e

```
<script>
root = '~';
</script>
```

Želimo li model iz EF modela, proširiti svojstvom koje nema svoj ekvivalent u tablici u bazi podataka, upotrijebit ćemo atribut?

a Partial

b Computed

c NotFromSql

d BindNever

e NotMapped

Označite načine kako se u bazi podataka mogu implementirati specijalizacija i generalizacija.

Napomena: Pitanje ima više točnih odgovora!

a table per primary key

b table per type

c table per hierarchy

d table per foreign key

e table per concrete class

Ako forma izgleda nalik sljedećoj

```
<form action="Edit" method="post">
    <input name="X" value="A"/>
    <input name="X" value="B"/>
    <input name="X" value="C"/>
</form>
```

tada, za prihvat vrijednosti, akcija na upravljaču treba biti definirana na sljedeći način

a `public IActionResult Edit(X[] name) { ... }`

b `public IActionResult Edit(string[] X) { ... }`

c `public IActionResult Edit(string A, string B,string C) { ... }`

d `public IActionResult Edit([From(Name="X")] string[] values) { ... }`

e `public IActionResult Edit(X A, X B, X C) { ... }`

Ako je akcija na upravljaču definirana na sljedeći način

```
public IActionResult Test(Stavka[] stavke) { ... }
```

tada unutar forme moraju biti elementi nalik:

a

```
<input type="hidden" name="stavke.Index" value="knjiga" />
<input type="text" name="stavke[knjiga].IdStavke" ... />

<input type="hidden" name="stavke.Index" value="5" />
<input type="text" name="stavke[5].IdStavke" ... />
```

b

```
<input type="text" name="stavke.IdStavke[0]" ... />
<input type="text" name="stavke.IdStavke[1]" ... />
<input type="text" name="stavke.IdStavke[2]" ... />
```

c

```
<input type="hidden" name="stavke.Index" value="0-2" />
<input type="text" name="stavke[Index].IdStavke" ... />
<input type="text" name="stavke[Index].IdStavke" ... />
<input type="text" name="stavke[Index].IdStavke" ... />
```

d

```
<input type="text" name="stavke.IdStavke" ... />
<input type="text" name="stavke.IdStavke" ... />
<input type="text" name="stavke.IdStavke" ... />
```

e

```
<input type="text" name="stavke[0].IdStavke" ... />
<input type="text" name="stavke[2].IdStavke" ... />
<input type="text" name="stavke[4].IdStavke" ... />
```

Koja od navedenih formi omogućava prihvat datoteke?

a `<form asp-action="Edit" post="multipart/form-data">`

b `<form asp-action="Create" method="post">`

c `<form asp-action="Create" method="get" enctype="multipart/form-data">`

d `<form asp-action="Create" method="multipart/post">`

e `<form asp-action="Edit" method="post" enctype="multipart/form-data">`

Akcija GetImage na nekom upravljaču unutar MVC aplikacije treba vratiti sliku (byte[]) ili status 404 a

a `public await Task<IActionResult> GetImage(int id) { ... }`

b `public async Task<IActionResult> GetImage(int id) { ... }`

c `public async Task<byte[]> GetImage(int id) { ... }`

d `public async IActionResult GetImage(int id) { ... }`

e `public async byte[] GetImage(int id) { ... }`

Ako nekom svojstvu postavimo validacijski atribut `Remote`, npr.

```
public class MyModel {
  [Remote(action:"CheckA", controller: "C", ErrorMessage = "Vrijednost nije ispravna")]
  public int A { get; set; }
```

tada u upravljaču C mora biti definiran postupak CheckA sljedećeg oblika

a public async Task<string> CheckA(int A) { ... }

b public async Task<bool> CheckA(int A) { ... }

c public async Task<string> CheckA(MyModel model) { ... }

d public async Task<IActionResult> CheckA(MyModel model) { ... }

e public async Task<ActionResult> CheckA(int A) { ... }

Proces kojim program može pregledavati i modificirati vlastitu strukturu tijekom izvođenje zove se

a perspektiva

b refleksija

c introspektiva

d simetrija

e retrospektiva

Koji od navedenih programskih odsječaka **ne omogućava** dobivanje informacija o tipu MojRazred

a
```
Type t = Type.GetType("NazivProjekta.MojRazred");
```

b
```
Type t = typeof(MojRazred)
```

c
```
Type t = MojRazred.Invoke(GetType());
```

d
```
Type t = Type.GetType("NazivProjekta.MojRazred, NazivAsemblija");
```

e
```
Type t = new MojRazred().GetType();
```

Ako je referenca *obj* tipa *object* referenca na objekt tipa *Robot* iz **nereferenciranog** asemblija pri čemu je Robot definiran na sljedeći način

```
public class Robot{
  public Robot(name) { ...
  public void Move(string direction, int moves) { ...
}
```

robota možemo poslati 2 koraka na sjeverozapad sljedećim programskim odsječkom

a
```
type.GetMethod("Move").Invoke("NW", 2);
```

b
```
obj.Invoke(type.GetMethod("Move"), "NW", 2);
```

c
```
((Robot) obj).Move("NW", 2);
```

d
```
obj.Move("NW", 2);
```

e
```
type.GetMethod("Move").Invoke(obj, new object[] { "NW", 2});
```

Ako objekt *type* tipa *Type* sadrži informaciju o razredu *Robot* iz **nereferenciranog** asemblija pri čemu je Robot definiran na sljedeći način

```
public class Robot{
  public Robot(name) { ...
  public void Move(string direction, int moves) { ...
}
```

novi robot pod nazivom R1P3 možemo dinamički instancirati na sljedeći način:

a
```
object robot = Activator.CreateInstance("Robot", "R1P3");
```

b
```
Robot robot = Activator.CreateInstance(Robot::new("R1P3"));
```

c
```
Robot robot = Activator.CreateInstance(type, "R1P3");
```

d
```
object robot = Activator.CreateInstance(type, "R1P3");
```

e
```
Robot robot = new Roboto("R1P3");
```

Želimo li definirati vlastiti atribut koji se može primijeniti **samo na svojstva**, pa primjerice možemo imati sljedeći odsječak:

```
public class Test{
  [Hide(When="Web")]
  public int X { get; set; }
}
```

napisat ćemo:

a

```
[AttributeUsage(AttributeTargets.Property)]
public class HideAttribute : Attribute {
    public string When { get; set; }
}
```

b

```
[AttributeUsage(AttributeTargets.Property)]
public class HideWhen : Attribute {
    public string Web { get; set; }
}
```

c

```
[AttributeUsage(AttributeTargets.Property)]
public class Hide : Attribute {
    public string When(string web, int x) {  }
}
```

d

```
public class Hide : PropertyAttribute {
    public string When { get; set; }
}
```

Što ispisuje sljedeći programski odsječak?

```
foreach(string s in GetData()) {
  Console.WriteLine(s);
}
...
IEnumerable<string> GetData() {
  Console.WriteLine("GetData");
  yield return "A";
  yield return "B";
  yield return "C";
}
```

a

```
GetData
```

b

```
GetData
A
GetData
B
GetData
C
```

c

```
GetData
A
```

d

```
GetData
A
B
C
```

Što ispisuje sljedeći programski odsječak?

```
var data = GetData();
Console.WriteLine(data.Last());
...
IEnumerable<string> GetData() {
  Console.WriteLine(1);
  yield return "A";
  Console.WriteLine(2);
  yield return "B";
  Console.WriteLine(3);
  yield return "C";
}
```

a

```
3
C
```

b

```
3
2
1
C
```

c

```
1
2
3
C
```

HATEOAS je skraćenica od

a Hypermedia Automated Extension of a Service

b Hypermedia Automated Engine of a Service

c Hypermedia and the reverse SOA

d Hypermedia as the Envelope of Application State

e Hypermedia as the Engine of Application State

REST servisi pozivaju se sljedećim HTTP zahtjevima

a GET, PUT, POST, DELETE

b LOAD, INSERT, UPDATE, DELETE

c GET, ADD, UPDATE, DELETE

d CREATE, READ, UPDATE, DELETE

e MOVE, SET, PEEK, POP

Ako WebApi (REST) servis uspješno ažurira postojeći podatak korisniku će se vratiti statusna poruka čiji je statusni kod

a 201 CREATED

b 202 ACCEPTED

c 302 FOUND (REDIRECT)

d 204 NO CONTENT

e 205 RESET CONTENT

Ako WebApi (REST) servis uspješno obriše podatak korisniku će se vratiti statusna poruka čiji je statusni kod

a 410 GONE

b 404 NOT FOUND

c 205 RESET CONTENT

d 204 NO CONTENT

e 301 MOVED PERMANENTLY

Ako WebApi (REST) servis uspješno doda novi podatak korisniku će se vratiti statusna poruka čiji je statusni kod

a 201 CREATED

b 302 FOUND (REDIRECT)

c 200 OK

d 304 NOT MODIFIED

e 204 NO CONTENT

WSDL označava:

a Mehanizam za povezivanje udaljenih postupaka.

b XML shemu za opis web servisa.

c Atribut kojim se označava postupak web servisa.

d Protokol za razmjenu informacija u distribuiranim, heterogenim okruženjima

e Postupak lociranja web servisa.

Želimo li da WebApi servis automatski izvrši provjeru primljenog modela te vrati status 400 u slučaju validacijskih pogrešaka, upravljaču ćemo dodati atribut

a BadRequestFilter

b ControllerBase

c ExceptionFilter

d ApiController

e TypeFilter

Izbacite uljeza

Novi/drugi naziv za Swagger je

a. OpenAPI

b. RESTful

c. JSON

d. OData

e. REST API

Koji nivo Richardsonovog modela zrelosti zadovoljavaju web-servisi koji koriste SOAP protocol?

a 1

b 0

c 3

d GET + POST

e 2

Ako je u WebAPI upravljaču definirano sljedeće:

```
public class MjestoController : ControllerBase {
    ...
    [HttpPost]
    public IActionResult Create (string s, int i) {
    ...
```

argumenti će se

a rekonstruirati iz parova ključ=vrijednost u FormData

b rekonstruirati iz tijela zahtjeva

c rekonstruirati iz query stringa

d rekonstruirati iz route

e dobiti korištenjem *dependency injectiona*

Ako je REST servis firma oblikovan prema standardu OData tada se prva 3 artikla poredana po cijeni silazno mogu dohvatiti na adresi

a `http://.../firma/artikl?$top=3&$orderby=CijArtikla desc`

b `http://.../firma/3/artikl?$$by=CijArtiklaDesc`

c `http://.../firma/artikl?$top=3&$filter=CijArtikla desc`

d `http://.../firma/artikl(3)/By/CijArtkla`

e `http://.../firma/artikl/3?orderby=CijArtkla&desc=true`

Ako je REST servis firma oblikovan prema standardu OData tada se artikl sa šifrom 8 nalazi na sljedećoj adresi

a `http://.../firma/artikl?filterByPK(8)`

b `http://.../firma/artikl?SifArtikla=8`

c `http://.../firma/artikl/8`

d `http://.../firma/artikl?filter=8`

e `http://.../firma/artikl(8)`

Razredi koji sadrže samo podatke te se koriste za prijenos podataka između različitih slojeva ili preko mreže nazivaju se

a Network Lying Objects

b Data Transfer Objects

c Network Transfer Objects

d Data Access Objects

e Data Access Layers

Command Query Separation pomaže riješiti probleme

a ORM-a

b debelog klijenta

c CRUD aplikacija

d tankog klijenta

e *constructor overinjectiona*

Ako je upit (po principu *Command Query Separation*) definiran na sljedeći način

```
public interface IQuery <TResult> { }
public class GetLongNamesQuery : IQuery<List<string>> {
    public int LongerThan { get; set; }
}
```

a sučelje za rukovatelje upita definirano s

```
public interface IQueryHandler<TQuery, TResult> where TQuery : IQuery<TResult> {
    Task<TResult> Handle (TQuery query);
}
```

tada rukovatelja upita za dohvat imena dužih od određenog broja znakova možemo opisati sučeljem:

a
```
public interface IGetLongNamesQueryHandler : IQueryHandler<IQuery<List<string>>, List<string>> { }
```

b
```
public interface IGetLongNamesQueryHandler : IQueryHandler<int, List<string>> { }
```

c
```
public interface IGetLongNamesQueryHandler : IQueryHandler<GetLongNamesQuery, int> { }
```

d
```
public interface IGetLongNamesQueryHandler : IQueryHandler<GetLongNamesQuery, List<string>> { }
```

e
```
public interface IGetLongNamesQueryHandler : IQueryHandler<List<string>, int> { }
```

Ako su zadani sljedeća razredi i sučelja (u svrhu ostvarenja principa *Command Query Separation*)

```
public interface IQuery <TResult> { }
public interface IQueryHandler<TQuery, TResult> where TQuery : IQuery<TResult> {
    Task<TResult> Handle (TQuery query);
}

public class ContainsCount : IQuery<int> {
    public string[] Names { get; set; }
}

public interface IContainsCountQueryHandler : IQueryHandler<ContainsCount, int> { }
```

koja od sljedećih naredbi je ispravna?

Napomena: referenca `IContainsCountQueryHandler handler` je ispravno definira te joj je pridružen konkretni rukovatelj upitom.

a
```
int q = await handler.Handle(new ContainsCount { Names = new [] { "a", "b" } });
```

b
```
IQuery<int> q = await handler.Handle(new ContainsCount { Names = new [] { "a", "b" } });
```

c
```
int q = await handler.Handle(new [] { "a", "b" });
```

d
```
string[] q = await handler.Handle(33);
```

e
```
string[] q = await handler.Handle(new ContainsCount(33) };
```

Ako je razred *Result* zadan s

```
public class Result {
    [JsonPropertyName("action")]
    public string ActionName { get; set; }
    [JsonPropertyName("status")]
    public bool ResultStatus { get; set; }
```

a u *Startup.cs* datoteci se nalazi

```
services...
        .AddJsonOptions(configure => configure.JsonSerializerOptions.PropertyNamingPolicy = JsonNamingPolicy.CamelCase);
```

tada se objekt tipa *Result* s određenim vrijednostima za *ActionName* i *ResultStatus* automatski pretvora u JSON oblika:

a

```
{"actionName":"Run", "resultStatus" : true}
```

b

```
{"result@7ea1246f" : {"ActionName":"Run", "ResultStatus" : true}}
```

c

```
{"ActionName":"Run", "ResultStatus" : true}
```

d

```
{"action":"Run", "status" : true}
```

e

```
{"Action":"Run", "Status" : true}
```

Pretpostavimo da **jediničnim testom** želimo provjeriti preusmjerava li upravljač *Drzava* s akcije *Index* na *Create* za situaciju koja je podešena u testu (npr. prazna baza podataka). Kojim od programskih odsječaka to možemo obaviti?

a Navedeno se ne može provjeriti jediničnim testiranjem

b

```
var controller = new DrzavaController(...
...
var result = controller.Index();
...
var redirectToActionResult =  Assert.IsType<RedirectToActionResult>(result);
Assert.Equal("Create", redirectToActionResult.ActionName);
```

c

```
string url = "/Drzava/Index";
var client = factory.CreateClient(new WebApplicationFactoryClientOptions {
                            AllowAutoRedirect = false
                        });
var response = await client.GetAsync(url);
Assert.Equal(HttpStatusCode.Redirect, response.StatusCode);
Assert.Equal("/Drzava/Create", response.Headers.Location);
```

d

```
var controller = new DrzavaController(...
...
var client = factory.CreateClient(new WebApplicationFactoryClientOptions {
                            AllowAutoRedirect = false
                        });
var response = await client.GetAsync(controller.Url);
Assert.Equal(HttpStatusCode.Redirect, response.StatusCode);
Assert.Equal(nameof(DrzavaController.Create), response.Headers.Location);
```

Što treba napisati u zadnjem retku sljedećeg programskog odsječka kako bi `c` bio cijeli broj s vrijednosti 10.

```
var mockOptions = new Mock<IOptionsSnapshot<AppSettings>>();
var appSettings = new AppSettings {
    AutoCompleteCount = 10
 };
mockOptions.SetupGet(options => options.Value).Returns(appSettings);
int c = [odaberite odgovor];
```

a `mockOptions.Object.Get("Value").AutoCompleteCount`

b `mockOptions.Value.AutoCompleteCount`

c `mockOptions.Get("Value").AutoCompleteCount`

d `mockOptions.Object.Value.AutoCompleteCount`

e `mockOptions.AutoCompleteCount`

Jedan od problema Web API servisa je da podaci koji se vraćaju ne sadrže dovoljno podataka, pa klijent mora raditi niz dodatnih upita kako bi dohvatio vezani podatak. Primjerice, može se dogoditi da popis gradova ne sadrži nazive države, već samo njihove kratice, pa je potrebno naknadno dohvatiti i nazive država.

Kako se naziva takav problem?

a overflow

b requests explosion

c subprojection

d underfetching

e inadequate flow

Što od navedenog **nije** jedno od 4 osnovna pravila servisno orijentirane arhitekture

a Ugovor, a ne implementacija

b Brzina i mala količina podataka

c Semantika, a ne samo sintaksa

d Neovisnost servisa

e Jasno određene granice

Ako je razred *Result* zadan s

```
public class Result {
    public string ActionName { get; set; }
    public bool ResultStatus { get; set; }
```

a u *Startup.cs* datoteci se nalazi

```
services...
        .AddJsonOptions(configure => configure.JsonSerializerOptions.PropertyNamingPolicy = null);
```

tada se objekt tipa *Result* s određenim vrijednostima za *ActionName* i *ResultStatus* automatski pretvora u JSON oblika:

a
```
{"action":"Run", "status" : true}
```

b
```
{"actionName":"Run", "resultStatus" : true}
```

c
```
{"Action":"Run", "Status" : true}
```

d
```
{"ActionName":"Run", "ResultStatus" : true}
```

e
```
{"result@7ea1246f" : {"ActionName":"Run", "ResultStatus" : true}}
```

Neuhvaćena iznimka u upravljaču/web servisu uzrokuje status

a 404 Not Found

b 418 I'm a teapot

c 205 Reset Content

d 400 Bad Request

e 500 Internal Server Error

Ako je u WebAPI upravljaču definirano sljedeće:

```
public class MjestoController : ControllerBase {
    ...
    [HttpPost]
    public IActionResult Create (MjestoViewModel model) {
    ...
```

model će se

a. rekonstruirati iz parova ključ=vrijednost u FormData

b. rekonstruirati iz query stringa

c. rekonstruirati iz route

d. rekonstruirati iz tijela zahtjeva

e. dobiti korištenjem *dependency injectiona*

Izbacite uljeza

a FromHeader

b FromForm

c FromApi

d FromRoute

e FromBody

Što će biti poruka nakon izvršavanja sljedećeg programskog odsječka?

```
List<string> list = null;
list.Should().NotBeNull().And.BeOfType<List<int>>().Which.Count().Should().Be(5);
```

a `NullReferenceException : Object reference not set to an instance of an object`

b `Expected type to be System.Collections.Generic.List``1[System.Int32 ...], but found System.Collections.Generic.List``1[System.String...`

c `Expected list to be 5, but found 0 (difference of -5)`

d `Expected list not to be <null>.`

e `RuntimeException: List<string> does not contain a definition for 'Should'`

Generički jezik (neovisan o konkretnom programskom jeziku) kojim se opisuju sučelja i strukture podataka (i preslikavanja u konkretne jezike) naziva se

a Domain Specific Language

b Service Listing Definition Language

c Service Specification Language

d Open Service Specification Language

e Interface Definition Language

Sljedeći odsječak

```
variable.Should().
        .NotBeNull()
        .And.BeOfType<List<int>>()
        .Which.Count().Should().Be(value)
```

će biti u

a testovima

b korisničkoj dokumentaciji

c Entitiy Framework modelu

d programskoj dokumentaciji

e opisu web-servisa ako se koristi Swagger

Akcija GetImage na nekom upravljaču unutar MVC aplikacije vraća sadržaj slike s `return File(...` ili null ako slika ne postoji. Ako se za dohvat slike koristi await i asinkrona metoda, tada postupak GetImage može biti sljedećeg oblika

a `public async IActionResult GetImage(id) { ... }`

b `public async byte[] GetImage(id) { ... }`

c `public async Task<FileContentResult> GetImage(id) { ... }`

d `public async Task<IActionResult> GetImage(id) { ... }`

e `public async Task<byte[]> GetImage(id) { ... }`

Za koju od sljedećih skraćenica se može reći da je distribuirani sustav u kojem sudjeluje više autonomnih servisa međusobno šaljući poruke preko granica određenih procesom, mrežom, ...

a WSDL

b CIA

c WCF

d POA

e SOA

Testovi se slažu tako da slijede obrazac

a Init - Assert - Pass

b Setup - Assert - Clean

c Arrange - Act - Assert

d Write - Fail - Pass

e Seek - Act - Destroy

Ako forma izgleda nalik sljedećoj

```
<form action="Edit" method="post">
    <input name="broj" value="1"/>
    <input name="broj" value="2"/>
    <input name="broj" value="3"/>
</form>
```

tada za prihvat vrijednosti 1, 2 i 3 akcija na upravljaču treba biti definirana na sljedeći način

a `public IActionResult Edit(int broj) { ... }`

b `public IActionResult Edit(string[] broj) { ... }`

c `public IActionResult Edit(string broj1, string broj2, string broj3) { ... }`

d `public IActionResult Edit(int broj1, int broj2, int broj3) { ... }`

e `public IActionResult Edit(string broj) { ... }`

Što je od navedenog **neispravno** za postupak WebAPI servisa izvedenog iz *ControllerBase*.

Napomena: *Tools* i *Categories* su vlastiti razredi.

Pitanje ima više točnih odgovora, tj. više izraza je neispravno.

a

```
[HttpPost]
public IActionResult Assign([FromQuery] Tools tools, [FromQuery] Categories categories)
```

b

```
[HttpPost]
public IActionResult Assign(Tools tools, Categories categories)
```

c

```
[HttpPost]
public IActionResult Assign([FromBody] Tools tools, [FromBody] Categories categories)
```

d

```
[HttpPost]
public IActionResult Assign([FromBody] Tools tools, [FromQuery] Categories categories)
```

e

```
[HttpPost]
public IActionResult Assign(Tools tools, [FromServices] Categories categories)
```