

Persistens med ”*repository pattern*”



Upphovsrätt för detta verk

Detta verk är framtaget i anslutning till kursen ASP.NET MVC vid Linnéuniversitetet.

Du får använda detta verk så här:

Allt innehåll i detta verk av Mats Look, förutom Linnéuniversitetets logotyp och symbol, är licensierad under:



Creative Commons Erkännande-IckeKommersiell-DelaLika 2.5 Sverige licens.

<http://creativecommons.org/licenses/by-nc-sa/2.5/se/>

Det betyder att du i icke-kommersiella syften får:

- kopiera hela eller delar av innehållet
- sprida hela eller delar av innehållet
- visa hela eller delar av innehållet offentligt och digitalt
- konvertera innehållet till annat format
- du får även göra om innehållet

Om du förändrar innehållet så ta inte med Linnéuniversitetets logotyp och symbol i din nya version!

Vid all användning måste du ange källan: "Linnéuniversitetet – ASP.NET MVC" och en länk till <https://coursepress.lnu.se/kurs/aspnet-mvc> och till Creative Common-licensen här ovan.

Persistent födelsedata i en XML-fil

- ✓ Data ska läsas från en XML-fil med namnet `birthdates.xml`. XML-noderna, med namnet `birthdate`, ska översättas till `Birthday`-objekt, sorteras på antal dagar kvar till nästa födelsedag och sedan presenteras.
- ✓ Nya `Birthday`-objekt ska kunna skapas och sparas i XML-filen.

```
<?xml version="1.0" encoding="utf-8"?>
<birthdates>
  <birthdate>
    <name>Ellen Nu</name>
    <date>2010-01-01T00:00:00</date>
  </birthdate>
  <birthdate>
    <name>Nisse Hult</name>
    <date>1967-02-11T00:00:00</date>
  </birthdate>
  <birthdate>
    <name>Carl Gustaf</name>
    <date>1946-04-30T00:00:00</date>
  </birthdate>
</birthdates>
```

Nästa födelsedag!

- Ellen Nu kommer att fylla 4 år på en onsdag om 47 dagar.
- Nisse Hult kommer att fylla 47 år på en tisdag om 88 dagar.
- Carl Gustaf kommer att fylla 68 år på en onsdag om 166 dagar.

Lägg till födelsedag

Hur läsa data från en XML-fil?

- ✓ *LINQ to XML* är en teknik för att skapa, skriva och läsa XML-data.
- ✓ När en XML-fil har laddats in i ett `XDocument`-objekt är det enkelt att ställa frågor.

```
version="1.0" encoding="utf-8"
<birthdates>
  <birthdate>
    <name>Ellen Nu</name>
    <date>2010-01-01T00:00:00</date>
  </birthdate>
  <birthdate>
    <name>Nisse Hult</name>
    <date>1967-02-11T00:00:00</date>
  </birthdate>
  <birthdate>
    <name>Carl Gustaf</name>
    <date>1946-04-30T00:00:00</date>
  </birthdate>
</birthdates>
```

```
public ActionResult Index()
{
    var path = Server.MapPath("~/App_Data/Birthdates.xml");
    var doc = XDocument.Load(path);
    var model = (from birthdate in doc.Descendants("birthdate")
                 select new Birthday
                 {
                     Name = birthdate.Element("name").Value,
                     Birthdate = DateTime.Parse(birthdate.Element("date").Value)
                 }).OrderBy(b => b.DaysUntilNextBirthday).ToList();

    return View("Index", model);
}
```

Hur skriva data till en XML-fil?

- ✓ För att lägga till en ny nod i XML-trädet laddas hela XML-filen, en ny nod skapas som läggs till rotnoden och XML-trädet sparas.

```
<?xml version="1.0" encoding="utf-8"?>
<birthdates>
  <birthdate>
    <name>Ellen Nu</name>
    <date>2010-01-01T00:00:00</date>
  </birthdate>
  <birthdate>
    <name>Nisse Hult</name>
    <date>1967-02-11T00:00:00</date>
  </birthdate>
  <birthdate>
    <name>Carl Gustaf</name>
    <date>1946-04-30T00:00:00</date>
  </birthdate>
</birthdates>
```

```
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Create(Birthday birthday)
{
    if (ModelState.IsValid)
    {
        var path = Server.MapPath("~/App_Data/Birthdates.xml");
        var doc = XDocument.Load(path);

        var element =
            new XElement("birthdate",
                new XElement("name", birthday.Name),
                new XElement("date", birthday.Birthdate));

        doc.Root.Add(element);
        doc.Save(path);

        return RedirectToAction("Index");
    }

    return View("Create", birthday);
}
```

Vad måste kontrollern veta?

✓ Retoriska(?) frågor...

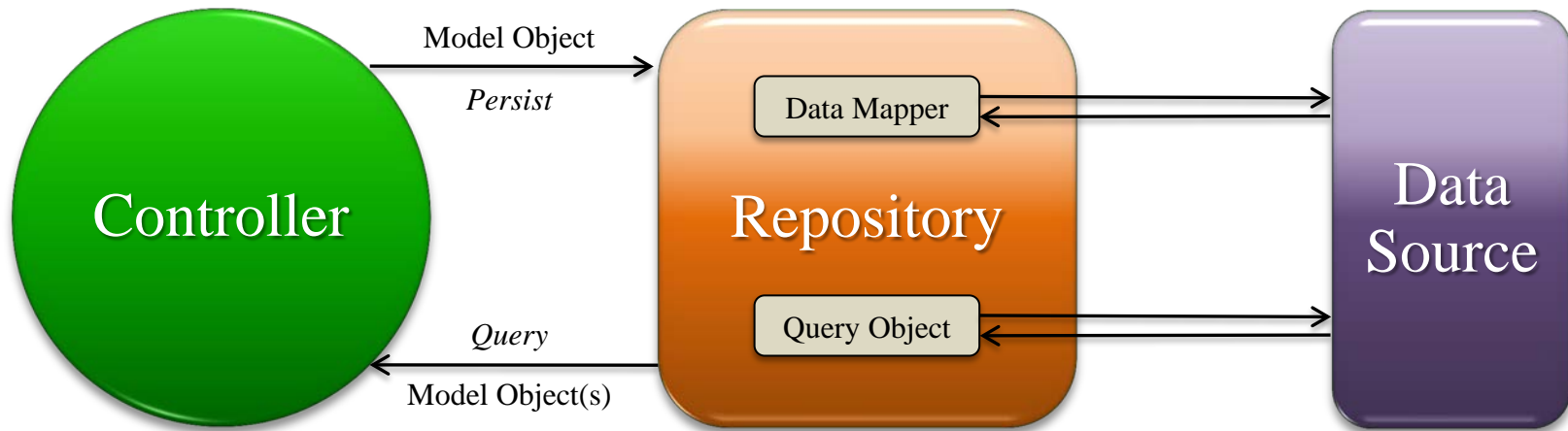
- Om sättet att lagra persistent data ändras är det då lämpligt att kontrollern måste känna till hur lagringen av `Birthday`-objekts data sker?
- Är det en fördel att kod, som har med hantering av persistent data att göra, kan komma att behöva dupliceras av en eller flera controller?
- Underlättar det att skriva tester mot controllrar då controllrar tar hand om alla detaljer beträffande hanteringen av persistent data?
- Förenklas underhållet av applikationen då...? Är det enklare att...? Kommer...?

✓ ...och svaret på samtliga frågor är: **NEJ!**

- ✓ Genom att låta kontrollern kommunicera med ett centrallager för data ("*repository*"), en klass för hantering av persistent data, istället för direkt med XML-filen kan koden som har med hantering av persistent data lyftas ut från kontrollern.

”Repository pattern”

- ✓ Ett centrallager (”*repository*”) finns kod för hantering av persistent data gentemot datakällan.



”Repository”-klass

- ✓ Klassen XmlRepository kapslar in all hantering av persistent data mot XML-filen.
- ✓ Via de publika metoderna GetBirthdays, InsertBirthday och Save kan en controller hämta, lägga till och spara Birthday-objekt.

```
public class XmlRepository
{
    private static readonly string PhysicalPath;
    private XDocument _document;
    private XDocument Document
    {
        get
        {
            return _document ?? (_document = XDocument.Load(PhysicalPath));
        }
    }

    static XmlRepository()
    {
        PhysicalPath = HttpContext.Current.Server.MapPath("~/App_Data/Birthdates.xml");
    }

    public List<Birthday> GetBirthdays()
    {
        return (from birthdate in Document.Descendants("birthdate")
                select new Birthday
                {
                    Name = birthdate.Element("name").Value,
                    Birthdate = DateTime.Parse(birthdate.Element("date").Value)
                }).OrderBy(b => b.DaysUntilNextBirthday).ToList();
    }

    public void InsertBirthday(Birthday birthday)
    {
        Document.Root.Add(
            new XElement("birthdate",
                new XElement("name", birthday.Name),
                new XElement("date", birthday.Birthdate)));
    }

    public void Save()
    {
        Document.Save(PhysicalPath);
    }
}
```


Den nya controllern

- ✓ Då all funktionalitet för hantering av persistent data är placerad i klassen `XmlRepository` behöver inte controllern längre ha någon kännedom om hur Birthday-objekten lagras.
- ✓ Det enda controllern behöver göra är att instansiera ett `XmlRepository`-objekt och använda de publika medlemmarna.

```
public class BirthdayController : Controller
{
    // private XmlRepository _repository = new XmlRepository();
    // GET: /Birthday/
    public ActionResult Index()
    {
        return View("Index", _repository.GetBirthdays());
    }
    // GET: /Birthday/Create
    public ActionResult Create()
    {
        return View("Create");
    }
    // POST: /Birthday/Create
    [HttpPost]
    [ValidateAntiForgeryToken]
    public ActionResult Create(Birthday birthday)
    {
        if (ModelState.IsValid)
        {
            _repository.InsertBirthday(birthday);
            _repository.Save();
            return RedirectToAction("Index");
        }
        return View("Create", birthday);
    }
}
```

Mer information

- ✓ .NET Language-Integrated Query for XML Data
 - <http://msdn.microsoft.com/en-us/library/bb308960.aspx>
- ✓ LINQ to XML
 - [http://msdn.microsoft.com/en-us/library/bb387098\(v=vs.110\).aspx](http://msdn.microsoft.com/en-us/library/bb387098(v=vs.110).aspx)
- ✓ Using LINQ to XML (and how to build a custom RSS Feed Reader with it)
 - <http://weblogs.asp.net/scottgu/archive/2007/08/07/using-linq-to-xml-and-how-to-build-a-custom-rss-feed-reader-with-it.aspx>
- ✓ Introduction to LINQ, Part 2: LINQ to XML
 - <http://www.codeguru.com/csharp/csharp/net30/article.php/c13715>
- ✓ Using LINQ to XML to query XML data
 - <http://dotnet.dzone.com/articles/using-linq-xml-query-xml-data>