

# Spring .NET

**Pittsburgh .NET User Group meeting  
March 2008**

Lou Biancaniello  
Jorge Balderas

summa



## About Summa

- Since 1996, Summa has been providing high-impact consulting services for companies of all sizes, including Global 2000 companies
- We specialize in helping companies evaluate and implement IT modernization strategies to better meet their business objectives
- IBM Premier Business Partner, BEA Select Level Partner, Microsoft Certified Partner, Oracle Partner Member, Member of Rockwell Automation's Solution Provider Program

summa

Summa 2008

2

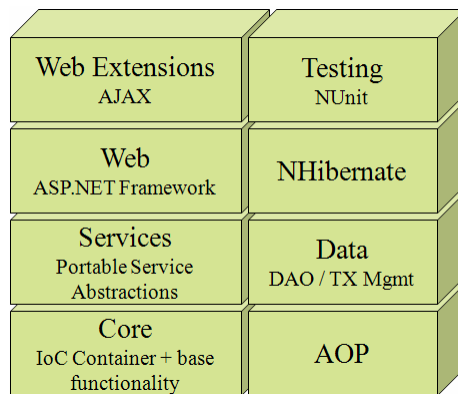
## Agenda

- ***Intro to Spring .NET***
- Dependency Injection
- Aspect Oriented Programming
- Transaction Management
- Testing
- Data Access
- Web
- Q&A

## Intro to Spring .NET

- Open source framework based on Spring for Java
- Provides core services to apply [primarily] IOC and AOP principles to .NET development
- Most services found in the Java version are available in Spring .NET including some specific to the .NET framework:
  - Support for ASP .NET templates, master pages, bidirectional data binding, and localization
  - Support for nHibernate, iBatis.net, ADO .NET
  - Support for COM+ interop, Windows Services, & .NET Remoting
- Current Release: Spring 1.1
  - 8 assemblies/24K LOC

## Spring .NET Framework modules



## Agenda

- Intro to Spring .NET
- **Dependency Injection**
- Aspect Oriented Programming
- Transaction Management
- Testing
- Data Access
- Web
- Q&A

## Dependency Injection

- Dependency Injection (DI) is a specific implementation of the Inversion of Control (IoC) pattern
- DI is the process of supplying an external dependency to a software component through the use of external configuration files
- The benefits are a loosely coupled system that allows for the exact implementation of a service to be defined outside the consuming Object

## Dependency Injection

- Conventionally, if an Object needs to gain access to a particular service, it must either create its own reference, or use a service locator to request a reference
- In DI, an Object need only create a property that holds a representation of the service necessary, and provide a public mechanism for setting that property

## Dependency Injection Example

- Typical Object Creation

```
CustomerDao customerDao = new CustomerDao();
```

- Object Creation with Spring.Net

```
<object id="customerDao" type="Summa.Demo.Spring.DataAccess.CustomerDao,  
    Summa.Demo.Spring.DataAccess">  
  <property name="AdoTemplate" ref="adoTemplate"/>  
</object>
```

- Inject via Factory...

```
IApplicationContext ctx = new  
ApplicationXmlContext("assembly://Summa.Demo.Spring.DataAccess/Summa.Demo.Sprin  
g.DataAccess.CustomerDao");  
ICustomerDao customerDao = ctx["customerDao"] as ICustomerDao;
```

- ...or simply define an Interface with a public setter.

summa

Summa 2008

9

## Agenda

- Intro to Spring .NET
- Dependency Injection
- **Aspect Oriented Programming**
- Transaction Management
- Testing
- Data Access
- Web
- Q&A

summa

Summa 2008

10

## Aspect Oriented Programming (AOP)

- A complement to Object Oriented Programming by providing a mechanism to perform *cross cutting* operations
- Examples of uses for AOP include but are not limited to:
  - Logging
  - Transactions
  - Security
  - Others

## AOP Concepts

- Aspect
  - A modularized concern that cross cuts multiple objects
- Join point
  - A point in the execution of a program where an aspect can be “embedded”. In Spring AOP, this is always tied to a method
- Advice
  - Action taken by an aspect at a specific join point
- Pointcut
  - Defines at what join points advice are applied
- Weaving
  - Linking aspects with other objects to create advised objects
- Introduction
  - This allows new methods or attributes to be added to an existing class

## AOP Advice types

- Before
  - The advice operation is performed before the targeted method (pointcut) is executed
- After Returning
  - The advice operation is performed after the targeted method returns control
- After Throws
  - The advice operation is performed after an Exception is thrown
- Around
  - This wraps a method call within an advice operation. This type allows for the advice to cancel the call to the method

## AOP in Spring .NET

- Creating an Aspect

```
public class NotImplementedAdvice : IBeforeAdvice
{
    public void Before()
    {
        MessageBox.Show("This function has not yet been enabled", "Error",
            MessageBoxButtons.Ok );
    }
}
```

# AOP in Spring .NET

- Define the Advisor

```
<object id="notImplementedAdvice"  
type="Summa.Demo.Spring.DIAop.NotImplementedAdvice, Summa.Demo.Spring.DIAop" />
```

- Define the Pointcut

```
<object id="customerDaoPointcutAdvisor"  
type="Spring.Aop.Support.RegularExpressionMethodPointcutAdvisor, Spring.Aop">  
  <property name="advice">  
    <ref object="notImplementedAdvice" />  
  </property>  
  <property name="pattern" value=".*GetAllCustomers.*" />  
</object>
```

summa

Summa 2008

15

# AOP in Spring .NET

- Define the Proxy

```
<object id="customerDao" type="Spring.Aop.Support.ProxyFactoryObject,  
Spring.Aop">  
  <property name="proxyInterfaces"  
    value="Summa.Demo.Spring.DataAccess.ICustomerDao" />  
  <property name="target" ref="customerDaoTarget" />  
  <property name="interceptorNames">  
    <list>  
      <value>customerDaoPointcutAdvisor</value>  
    </list>  
  </property>  
  <property name="pattern" value=".*GetAllCustomers.*" />  
</object>
```

summa

Summa 2008

16

## AOP in Spring .NET

- Spring .NET includes Aspects for commonly used AOP functions.
  - Caching
  - Exception Handling
  - Logging
  - Retry
  - Transactions

## Agenda

- Intro to Spring .NET
- Dependency Injection
- Aspect Oriented Programming
- ***Transaction Management***
- Testing
- Data Access
- Web
- Q&A

## Transaction Support

- Spring provides transaction managers which provide granular control over the transactions in regards to:
  - Atomic
  - Consistent
  - Isolated
  - Durable
- When defining a transaction manager in Spring, the propagation behavior and isolation levels need to be set
  - Default values are:
    - Propagation Required
    - Isolation Default

## Transaction Support

- Separate Transaction Managers (which act as proxies) are available for the different Data Managers supported by Spring
- Examples include managers for:
  - NHibernate

# Transaction Support in Spring .NET

- Example of using AOP for Transaction Management:

```
<object id="daoProxy"
  type="Spring.Transaction.Interceptor.TransactionProxyFactoryObject,
  Spring.Data">
  <property name="platformTransactionManager"
    value="adoTransactionManager" />
  <property name="target">
    <object name="sampleDao" type="Assembly.sampleDao, Assembly" />
  </property>
  <property name="transactionAttributes">
    <name-values>
      <add name="save*" value="PROPAGATION_REQUIRED" />
    </name-values>
  </property>
</object>
```

## Agenda

- Intro to Spring .NET
- Dependency Injection
- Aspect Oriented Programming
- Transaction Management
- **Testing**
- Data Access
- Web
- Q&A

## Unit Testing support

- Spring.Testing.NUnit.dll supports NUnit 2.4.1
- Caching of Spring IoC container between test case execution
- Dependency injection of test fixture instances

## Unit Testing example

- ApplicationContext.xml

```
<db:provider id="dbProvider" provider="SqlServer-2.0"
  connectionString="Server=YORTCH;Initial Catalog=Northwind;Integrated
  Security=SSPI"/>
<object id="adoTemplate" type="Spring.Data.Generic.AdoTemplate, Spring.Data">
  <property name="DbProvider" ref="dbProvider"/>
  <property name="DataReaderWrapperType"
    value="Spring.Data.Support.NullMappingDataReader, Spring.Data"/>
</object>
<object id="customerDao" type="Summa.Demo.Spring.DataAccess.CustomerDao,
  Summa.Demo.Spring.DataAccess">
  <property name="AdoTemplate" ref="adoTemplate"/>
</object>
```

## Unit Testing example

- Test Class

```
[TestFixture]
public class CustomerDaoTest
{
    private IApplicationContext ctx;

    [SetUp]
    public void InitContext()
    {
        // Configure Spring programmatically
        NamespaceParserRegistry.RegisterParser(typeof(DatabaseNamespaceParser));
        ctx = new XmlApplicationContext(
            "assembly://Summa.Demo.Spring.DataAccess/Summa.Demo.Spring.DataAccess/ApplicationContext.xml"
        )
    }
}
```

## Unit Testing example

- Test case

```
[Test]
public void GetCustomersTest()
{
    CustomerDao dao = ctx["customerDao"] as CustomerDao;
    IList<Customer> customerList = dao.GetAllCustomers();
    Assert.IsNotNull(customerList);
    Assert.AreEqual(91, customerList.Count);
}
```

## Agenda

- Intro to Spring .NET
- Dependency Injection
- Aspect Oriented Programming
- Transaction Management
- Testing
- **Data Access**
- Web
- Q&A

summa

Summa 2008

27

## Data Access

- Aimed at abstracting the data access layer from a specific implementation
- Provides wrapper classes and generic unchecked exceptions
- Supported Database frameworks:
  - ADO .NET
  - NHibernate
- Supported Database providers:
  - SqlServer, Oracle, OleDb, MySql, Npgsql, DB2, SQLite, Sybase, Odbc

summa

Summa 2008

28

## Exception handling

- Provides a consistent exception hierarchy
- DB specific error codes can be mapped using the **ErrorCodeExceptionTranslator**

```
<add
  key="SqlServer2005.DbMetadata.ErrorCodes.DataIntegrityViolationCodes"
  value="544,2601,2627,8114,8115"/>
```

- Sample data access exceptions:
  - **BadSqlGrammarException**,
  - DataIntegrityViolationException**,
  - PermissionDeniedDataAccessException**, etc.

summa

Summa 2008

29

## Typical ADO .NET code

```
public IList<Customer> FindAllCustomers() {
    IList<Customer> list = new List<Customer>();
    try
    {
        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            string sql = "select name, id from Customer";
            using (SqlCommand command = new SqlCommand(sql, connection))
            {
                connection.Open();
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        string name = reader.IsDBNull(0) ? string.Empty :
                            reader.GetString(0);
                        int id = reader.IsDBNull(1) ? 0 : reader.GetInt32(1);
                        Customer customer = new Customer(name, id);
                        list.Add(customer);
                    }
                }
            }
        }
    }
    catch (Exception e) { //throw application exception }
    return list; }
}
```

summa

Summa 2008

30

## AdoTemplate

- Thread-safe class based on IoC (i.e. callback) design
- Developers need to worry primarily on the IDbCommand instance, resource management handled by the framework

```
public int FindCountWithPostalCodeWithDelegate(string postalCode) {  
    return AdoTemplate.Execute<int>(delegate(DbCommand command)  
        command.CommandText = cmdText;  
        DbParameter p = command.CreateParameter();  
        p.ParameterName = "@PostalCode";  
        p.Value = postalCode;  
        command.Parameters.Add(p);  
        return (int)command.ExecuteScalar();  
    });  
}
```

summa

Summa 2008

31

## RowMapper Example

```
public virtual IList<Customer> GetAllCustomers()  
{  
    return AdoTemplate.QueryWithRowMapperDelegate<Customer>(CommandType.Text,  
        cmdText, delegate(IDataReader dataReader, int rowNum)  
        {  
            Customer customer = new Customer();  
            customer.Address = dataReader.GetString(0);  
            customer.City = dataReader.GetString(1);  
            customer.CompanyName = dataReader.GetString(2);  
            customer.ContactName = dataReader.GetString(3);  
            customer.ContactTitle = dataReader.GetString(4);  
            customer.Country = dataReader.GetString(5);  
            customer.Id = dataReader.GetString(6);  
            return customer;  
        });  
}
```

summa

Summa 2008

32

## Agenda

- Intro to Spring .NET
- Dependency Injection
- Aspect Oriented Programming
- Transaction Management
- Testing
- Data Access
- **Web**
- Q&A

## Web

- Introduces object scopes: page, session or application
- Dependency injection for controls and pages
- Provides support for Master Pages for ASP .NET 1.1
- Allows exposing plain .NET objects (PONO) as Web Services
- Must inherit from Spring.Web.UI.Page

## Web

- ASP .NET is not a true MVC implementation
  - Controller within the page is tightly coupled with the view
    - Spring solution: Bi-directional data binding
  - Flow of control is typically handled with Redirect or Transfer calls
    - Spring solution: result mapping (aliases)

## Decoupling view from controller

- Instead of modifying view element directly:

```
protected void btnCapitalize_Click(object sender, EventArgs e)
{
    txtName.Text = Name.ToUpper();
}
```

- Update model and let controller update view:

```
protected void btnCapitalize_Click(object sender, EventArgs e)
{
    Name = Name.ToUpper();
}
```

## How to enable Spring on a web project

1. Add references to **Spring.Web** and **Spring.Core** assemblies
2. **Web.config** changes
  - Add spring `sectionGroup`
  - Add “`PageHandlerFactory`” `HttpHandler` for \*.aspx path
  - Add the “`SpringModule`” `HttpModule`
  - Add spring context configuration

## Model management

```
protected override void InitializeDataBindings()
{
    BindingManager.AddBinding("lblCount.Text", "customers.Count");
}
protected override void InitializeModel()
{
    customers = (IList<Customer>)this.customerBusinessDelegate.GetAllCustomers();
}
protected override void LoadModel(object savedModel)
{
    customers = (IList<Customer>)savedModel;
}
protected override object SaveModel()
{
    return customers;
}
```

## Agenda

- Intro to Spring .NET
- Dependency Injection
- Aspect Oriented Programming
- Transaction Management
- Testing
- Data Access
- Web
- **Q&A**

## References

- Spring Framework .NET  
[www.springframework.net](http://www.springframework.net)
- Spring .NET 1.1 Reference Documentation  
<http://springframework.net/doc-latest/reference/html/index.html>
- Spring .NET Download (includes QuickStart samples)  
[http://sourceforge.net/project/showfiles.php?group\\_id=106751&package\\_id=115147&release\\_id=560046](http://sourceforge.net/project/showfiles.php?group_id=106751&package_id=115147&release_id=560046)