Introduction

Developers building modern applications have a range of tools and technologies that they can employ. Commonly, developers building applications for the Windows® platform are faced with choosing between selecting Java and its associated technologies, or the Microsoft® .NET Framework. Java and the .NET Framework both support tools for building a range of software, from mobile applications running on a smart phone, through desktop applications with graphical user interfaces (GUIs), to large-scale enterprise applications that are used to underpin the business functions of an organization. In theory, there is a lot of commonality between the features and facilities that Java and the .NET Framework provide. However, there are some significant differences in the way in which these environments operate, and the way in which you design, build, and host applications.

If you are an experienced Java developer, you will already be familiar with the architecture of a typical Java enterprise solution and the technologies that you use to construct it. The purpose of this paper is to describe the equivalent structure of a typical .NET Framework application, and to provide a mapping from the technologies that are frequently used in the Java world to their .NET Framework brethren.
Comparing Java and .NET Framework Architectures

The term “Java” covers several items, ranging from the programming language with the same name through to the various technologies that underpin a typical solution that is built by using this language. However, it is important to understand that Java is principally the definition of a set of specifications and not a product in its own right. Many vendors implement a Java compiler, a Java virtual machine (JVM), and a Java class library. These implementations provide the core functionality and runtime environment that are necessary to build portable Java applications. As long as developers avoid using any vendor-specific extensions in their applications, their Java code will run unchanged on any computer that has a conformant implementation of the JVM and Java class library installed. To support the wide range of solutions that modern organizations commonly require, Java defines a large number of specifications covering items such as GUIs, data access, security, Web services, scalable enterprise components, and so on. Figure 1 depicts the common specifications and other elements that most enterprise Java solutions utilize. The figure partitions the items between those used by client and mobile applications, and those used to build server-side components.

Server-side components execute in an environment that an application server provides. It is the application server that implements the various server-side specifications, such as those covering Enterprise JavaBeans (session beans and message-driven beans). The application server also works in cooperation with other providers, such as the Java Database Connectivity (JDBC) interface to a database management system, to access their technologies through these specifications. Frequently used application servers include Tomcat and OpenEJB from Apache, and JBoss. Additionally, many Java application servers provide additional facilities that the various Java specifications do not currently cover; a prime example is workflow management, and JBoss provides jBPM to implement this feature.

In comparison to Java, the .NET Framework is a concrete (and free) product that is developed by Microsoft and integrated into the Windows operating system. It does not rely on third-party vendors to provide an implementation, although it is highly extensible and many third-party additions are available. The .NET Framework also includes the equivalent of many of the features beyond those of the Java specifications, but commonly implemented by Java application server vendors; for example, it provides a workflow engine. Figure 2 illustrates the technologies that constitute the .NET Framework, following the same structure as far as possible as Figure 1. In this way, you can see how elements that you construct by using the .NET Framework might map to the corresponding Java components.
Comparing Java and .NET Framework Architectures

Figure 1. Common Java specifications and technologies

Figure 2. .NET Framework architecture
Scenario: The Hotel Reservation System

To fully understand the similarities and differences between Java and the .NET Framework, consider the example of a Hotel Reservation System. The logical model shown in Figure 3 depicts the functionality that this system implements.

The principal users of the system are customers, who can access the system from a Web browser or mobile phone across the Internet, and reception staff who run desktop applications on-premises inside the hotel.

The operations that customers can perform include:

- Registering themselves, and saving their details such as name, billing address, telephone number, and e-mail address.
- Querying the availability of rooms according to various criteria such as date, style of room (standard or executive), and occupancy (single or double).
- Making a booking and reserving a room, and also querying and canceling bookings that have already been made.

Hotel reception staff have access to extended functionality. They can register customers, query room availability, and make and cancel bookings for any customer, but they can also check customers in when they arrive at the hotel, generate bills, and record payment.
All information about customers, bookings, and room availability is held in the hotel information database.

Apart from the features that are available to customers and hotel reception staff, the Hotel Reservation System also supports:

- **Management reporting.** These are a range of reports, standard and ad hoc, that a hotel manager can request about room occupancy, customer preferences, reservations, and so on. The reports are generated by a series of modules that query the hotel information database and generate spreadsheets and other documents.

- **Batch processing.** A hotel manager can use the system to generate mailshots, promotions, and other items. These operations are run as batch jobs.

- **Integration with external systems.** The hotel frequently needs to exchange data and cooperate with systems that are implemented externally, such as partner organizations of the hotel, external services that handle customer payments, and third-party providers that implement hotel reservation brokerage services across the Internet.

**The Java Implementation of the Hotel Reservation System**

Figure 4 illustrates the architecture of one possible implementation of the Hotel Reservation System, constructed by using Java technologies.
The core of the system is implemented by using the Apache OpenEJB server to host a collection of session beans that implement the business logic for managing customers, rooms, and bookings. The session beans contain the business logic for the various services, and make use of a set of Java classes that model the data for customers, rooms, and bookings; these are the Customer, Booking, and Room entity classes, which are persisted by using the Java Persistence application programming interface (API) on top of Hibernate, which in turn saves and retrieves data to a MySQL database. In this implementation, Hibernate uses JDBC to connect to the MySQL database.

The business logic behind making a booking and reserving a room is a potentially long-running, complex, multistep process that involves the customer providing payment details, and sending the customer a confirmation e-mail message. The Booking session bean encapsulates this logic in a jBPM workflow, which includes the necessary support for suspending and resuming the booking process if required. Another jBPM workflow handles requests that are received from partner hotels for transferring bookings made at those hotels into the Hotel Reservation System (see later).
The core of the system makes use of the Spring Framework and its configuration and dependency injection functionality to deploy, instantiate, and manage the various components that constitute the solution.

Customer Access

Customers connect to the application either by using a Web browser, or from a custom mobile application. The Web browser runs a Java application that uses JavaFX to provide a rich user interface.

Two versions of the mobile application are available. The original version is based on Java Platform, Micro Edition (Java ME), which is intended for use on handheld devices. A later version has been constructed for the Android platform and is intended for use on more recent mobile phones. Both applications provide the same functionality, and communicate with the Hotel Reservation System over a standard HTTP connection, using Secure Sockets Layer (SSL) to protect any sensitive information, as does the browser-based version of the application.

The Web browser and mobile applications both connect to an Apache HTTP server that the hotel hosts. The HTTP server serves requests for static HTML pages and CGI-generated output. However, the bulk of the traffic from customers involves processing requests that are implemented by using JavaServer Pages (JSP) pages. The HTTP server is configured with the mod_jk Apache/Tomcat connector to forward these requests to a Tomcat server. The user interface logic is actually implemented by using JavaServer Faces (employing JSP pages and servlets behind the scenes), and the business logic is delegated to the various session beans described earlier.

Note:

The architects also considered implementing the Web front-end by using Apache Struts, but opted for JavaServer Faces due to its extensibility (it provides a relatively simple way to plug in modules that support non-HTML clients), and its ability to handle complex user interfaces in an arguably more organized manner than Struts.

Reception Staff Access

Hotel reception staff run a Java desktop application that makes use of a user interface that was developed by using Swing. This application acts as an Enterprise JavaBeans (EJB) client, making use of the session beans that the OpenEJB server hosts. The desktop application connects to the OpenEJB server by using Java Remote Method Invocation (Java RMI) over TCP/IP (these are local connections inside the hotel).
The Java Implementation of the Hotel Reservation System

Reporting

The Reporting application is implemented by using the OpenOffice suite, utilizing the MySQL connector for OpenOffice to connect to the database and extract data. The Reporting application does not make use of the entity classes or the session beans because it simply needs to query data in the database. This structure is fast and supports ad-hoc querying as long as the employee who is generating a report understands how to phrase Structured Query Language (SQL) statements.

Batch Processing

The batch processing elements are a small set of stand-alone custom Java applications that initiate a set of background tasks in the Hotel Reservation System. Some of these applications are executed on a regular basis (such as mailshots), while others are created on demand to support a specific marketing campaign (such as hotel promotions and special offers). Due to the potential processing requirements of these applications, and to avoid adversely affecting the quality of service to customers and hotel reception staff, these applications simply post requests to a Java Message Service (JMS) queue (implemented by using Apache ActiveMQ). A collection of message-driven beans (each designed to support a specific batch job) hosted by OpenEJB receives these requests and processes them during off-peak hours.

Note:

An alternative approach would be to use the Spring Batch framework and a collection of regular Java classes rather than message-driven beans to implement the batch processing elements.

Integration with Third-Party Systems

The Hotel Reservation System exposes data about room availability and pricing through a small set of Web services. Third-party Web client applications can invoke these Web services, and use the information that is retrieved as part of their own solutions, acting as hotel room brokerage services or portals. The Hotel Reservation System implements these services by using the Apache CXF framework.

The Hotel Reservation System also supports exchange of data and interaction with systems that partner organizations run, including other hotels that may request to transfer bookings from their own premises (if a customer requests to switch hotels, for example), and organizations that implement secure payment services. Such exchanges involve an interchange and possible transformation of documentation (different hotels may use different data formats and schemas), and may involve multiple sites. The hotel
The .NET Framework Implementation of the Hotel Reservation System

The .NET Framework exchanges data with other systems through a set of well-defined processes, but requires the careful use of enterprise application integration technologies to maintain consistency and security, and to prevent data loss. The logic for these exchanges is encapsulated in a set of business process orchestrations that are implemented by using the jBPM workflows described earlier (booking payment and booking transfer) in conjunction with JBoss ESB to connect to these external systems.

The .NET Framework Implementation of the Hotel Reservation System

The Microsoft .NET Framework is a software framework for developing and executing applications on Windows client and server operating systems. It provides features and functionality that are designed specifically for creating and running applications on the Windows platform, but also supports connectivity with non-Windows systems. .NET Framework applications are executed by using the common language runtime (CLR). The CLR implements an abstraction layer over the operating system and provides .NET Framework applications with controlled access to machine resources; its role is similar to that of the JVM that Java applications use. Figure 5 shows an implementation of the Hotel Reservation System, but this time based on the .NET Framework and incorporating Windows technologies and services.
Core System

In this solution, the Customer, Booking, and Room services provide the core functionality. These items are Windows Communication Foundation (WCF) services. WCF provides a framework for building services that can operate in a manner that is very similar to EJB session beans. Like a session bean, a WCF service can be stateful or stateless. WCF enables you to define the logic for a service by using ordinary application code, or by using Windows Workflow Foundation (WF) workflows. Listings 1 and 2 show part of the implementation of the WCF Customer Service, providing operations for creating and managing customers, followed by the equivalent elements from the corresponding Java session bean in Listings 3 and 4. Note that both implementations follow a similar approach, defining the operations in an interface and providing a class that implements this interface.

```csharp
using System;
```
using System.Collections.Generic;
using System.ServiceModel;

namespace HotelReservationSystemServices
{
    [ServiceContract]
    public interface ICustomerService
    {
        [OperationContract]
        int CreateNewCustomer(string foreName, string middleInitial, string lastName,
                                DateTime dateOfBirth);

        [OperationContract]
        bool DeleteCustomer(int customerID);

        [OperationContract]
        List<Booking> FindBookings(int customerID, DateTime fromDate, DateTime toDate);
    }
}

Listing 1. The WCF interface for the Customer Service

using System;
using System.Collections.Generic;
using System.Transactions;

namespace HotelReservationSystemServices
{
    public class CustomerService : ICustomerService
    {
        public int CreateNewCustomer(string foreName, string middleInitial, string lastName,
                                      DateTime dateOfBirth)
        {
            int customerID = -1;
            Customer customer = null;

            // The HotelReservationSystemDBEntities class is an Entity Framework context object
            // that provides the connection to the database, and provides the
            // necessary transaction management
            HotelReservationSystemDBEntities context = new HotelReservationSystemDBEntities();

            using (TransactionScope transaction = new TransactionScope())
            {
                try
                {
                    // Create a new Customer entity object
                    Customer = new Customer
                    {
                        ForeName = foreName,
                        MiddleInitial = middleInitial,
                        LastName = lastName,
                        DateOfBirth = dateOfBirth
                    };

                    // Save the Customer entity to the database
                    context.Customers.AddObject(customer);
                }
            }
        }
    }
}
context.SaveChanges();
customerID = customer.CustomerID;
transaction.Complete();
context.AcceptAllChanges();
}  // If an error occurs, the transaction is aborted and the customer is not saved
   // Set the customerID to -1 to indicate that no customer was created
   catch (Exception e)
   {
      customerID = -1;
   }
   context.Dispose();
   return customerID;
}

public bool DeleteCustomer(int customerID)
{
   ... // Implementation not shown
}

public List<Booking> FindBookings(int customerID, DateTime fromDate, DateTime toDate)
{
   ... // Implementation not shown
}

Listing 2. The WCF implementation of the Customer Service

import HotelReservationSystemEntities.Booking;
import java.util.Date;
import java.util.List;
import javax.ejb.Remote;

@Remote
public interface CustomerServiceRemote {

   int createNewCustomer(String foreName, char middleInitial, String lastName,
                         Date dateOfBirth);

   boolean deleteCustomer(int customerID);

   List<Booking> findBookings(int customerID, Date fromDate, Date toDate);
}

Listing 3. The remote interface for the EJB version of the Customer Service

package HotelReservationSystemEnterpriseBeans;
import HotelReservationSystemEntities.Customer;
import HotelReservationSystemEntities.Booking;
import java.util.Date;
import java.util.List;
import javax.ejb.Stateless;
import org.hibernate.Transaction;
import org.hibernate.Session;

@Stateless
public class CustomerService implements CustomerServiceRemote {

    @Override
    public int createNewCustomer(String foreName, char middleInitial, String lastName, Date dateOfBirth) {

        Integer customerID = null;
        Customer customer = null;
        Transaction transaction = null;

        // The HibernateManagment class (not shown) provides the static getSessionFactory
        // method which returns an org.hibernate.SessionFactory object for creating and
        // managing Hibernate sessions
        Session session = HibernateManagement.getSessionFactory().openSession();

        try {
            transaction = session.beginTransaction();

            // Create and save the new booking
            customer = new Customer();
            customer.setForeName(foreName);
            customer.setMiddleInitial(middleInitial);
            customer.setLastName(lastName);
            customer.setDateOfBirth(dateOfBirth);

            customerID = (Integer) session.save(customer);
            transaction.commit();
        }
        // If an error occurs, rollback the transaction
        catch (Exception e) {
            customerID = null;
            if (transaction != null) {
                transaction.rollback();
            }
        }
        finally {
            session.close();
            if (customerID == null) {
                return -1;
            } else {
                return customerID.intValue();
            }
        }
    }
}
In the Java version of the Hotel Reservation System, the Booking Service session bean employed a jBPM workflow for the reasons described earlier (the booking process is a potentially long-running and complex multistep process). In the .NET Framework version, the Booking Service is implemented as a WF workflow. Long-running WF services can be persisted to a database and their state restored at a later date to resume running. Microsoft Visual Studio® (the primary development tool for building .NET Framework applications) provides a graphical designer that you can use to build and test your workflows. Figure 6 shows an example.
An important difference between a WF service and a jBPM workflow concerns integration with external systems. WF services are intended to handle the internal business logic inside an organization. You can use them to orchestrate interbusiness processes, but Microsoft provides **Microsoft BizTalk® Server** as a better solution for this purpose (an orchestration may involve complex data transformations and integration with external security services, for which BizTalk Server is ideally suited). Therefore, in the .NET Framework version of the Hotel Reservation System, only the internal part of the logic associated with reserving a room is implemented by using a WF workflow; the
interaction with the external payment service is handled by the BizTalk Server Booking Payment Orchestration, which is invoked from the WF Booking Service.

The WCF services are hosted by using the Windows Process Activation Service (WAS) in conjunction with Internet Information Services (IIS). IIS provides a host environment for Web services and Web applications, and is roughly equivalent to Tomcat with OpenEJB in the Java solution. WAS extends IIS by supporting non-Web protocols such as TCP, named pipes, and message queues.

As in the Java solution, the WCF services depend on a collection of entity classes (the Booking and Customer types referenced in Listings 1 and 2) to retrieve data from, and update, the database. The .NET Framework provides a variety of data access technologies that you can use to implement this functionality. These technologies include ADO.NET and the Entity Framework. ADO.NET implements a programmatic model that is similar in concept to JDBC, and that can connect to a variety of data sources by plugging in the appropriate database driver. The Entity Framework implements an Object-Relational Management system, which is similar to that provided by the Java Persistence API (JPA) and Hibernate. The Entity Framework includes a suite of Visual Studio templates and other code-generation elements that you can use to construct an entity model from a database. These tools can generate a set of entity classes, together with the necessary infrastructure classes for retrieving data into an entity collection; creating, deleting, and updating entities in a collection; and saving these entities back to the database. Figure 7 shows the Entity Designer in Visual Studio, which you can use to define the entities for a solution and manage the relationships between them.
The .NET Framework Implementation of the Hotel Reservation System

Figure 7. The Entity Designer in Visual Studio

Listings 5 and 6 show part of the code that is generated by using the Entity Framework. The Entity Framework generates two principal types of classes:

- **Entity classes.** These model the data from the data source and are directly equivalent to the entity classes that JPA and Hibernate use. The code in Listing 5 shows the detail for some selected properties in the `Customer` entity class. Note that these properties provide controlled access to fields in the class, but that they also invoke methods alerting the Entity Framework and the user application of any changes made to the data.

- **Context classes.** These are used to connect to the underlying data source to fetch, insert, update, and delete data. They fulfill a role that is similar to that of a Hibernate `Session` object. The context object contains an `ObjectSet` collection for each type of entity class, and populates this collection with the data from the data source. The `ObjectSet` collection class exposes methods such as `AddObject` to add an entity.
object to an ObjectSet collection. Listing 2 showed an example, adding a newly created Customer object to the Customer's ObjectSet for a context object.

Context classes are derived from theObjectContext class, which provides most of the core functionality for interacting with a data source. For example, the SaveChanges method (also illustrated in Listing 2) propagates any changes made to entities that are held in an ObjectSet collection associated with a context object back to the data source. The code in Listing 6 shows the HotelReservationSystemDBEntities context class that the Entity Framework generates, illustrating how such a class is structured.

```csharp
namespace HotelReservationSystemServices
{
    [EdmEntityTypeAttribute(NamespaceName="HotelReservationSystemDBModel", Name="Address")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class Address : EntityObject
    {
        ...
    }

    [EdmEntityTypeAttribute(NamespaceName="HotelReservationSystemDBModel", Name="Booking")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class Booking : EntityObject
    {
        ...
    }

    [EdmEntityTypeAttribute(NamespaceName="HotelReservationSystemDBModel", Name="Customer")]
    [Serializable()]
    [DataContractAttribute(IsReference=true)]
    public partial class Customer : EntityObject
    {
        public static Customer CreateCustomer(global::System.Int32 customerID, global::System.String foreName, global::System.String lastName, global::System.DateTime dateOfBirth, global::System.Int32 addressAddressID)
        {
            Customer customer = new Customer();
            customer.CustomerID = customerID;
            customer.ForeName = foreName;
            customer.LastName = lastName;
            customer.DateOfBirth = dateOfBirth;
            customer.AddressAddressID = addressAddressID;
            return customer;
        }

        [EdmScalarPropertyAttribute(EntityKeyProperty=true, IsNullable=false)]
        [DataMemberAttribute()]
        public global::System.Int32 CustomerID
    }
```
```csharp
{
    get
    {
        return _CustomerID;
    }
    set
    {
        if (_CustomerID != value)
        {
            OnCustomerIDChanging(value);
            ReportPropertyChanging("CustomerID");
            _CustomerID = StructuralObject.SetValidValue(value);
            ReportPropertyChanged("CustomerID");
            OnCustomerIDChanged();
        }
    }
}
private global::System.Int32 _CustomerID;
partial void OnCustomerIDChanging(global::System.Int32 value);
partial void OnCustomerIDChanged();

[EdmScalarPropertyAttribute(EntityKeyProperty=false, IsNullable=false)]
[DataMemberAttribute()]
public global::System.String ForeName
{
    get
    {
        return _ForeName;
    }
    set
    {
        OnForeNameChanging(value);
        ReportPropertyChanging("ForeName");
        _ForeName = StructuralObject.SetValidValue(value, false);
        ReportPropertyChanged("ForeName");
        OnForeNameChanged();
    }
}
private global::System.String _ForeName;
partial void OnForeNameChanging(global::System.String value);
partial void OnForeNameChanged();
...
[XmlIgnoreAttribute()]
[SoapIgnoreAttribute()]
[DataMemberAttribute()]
[EdmRelationshipNavigationPropertyAttribute("HotelReservationSystemDBModel", "FK_Booking_Customer", "Booking")]
public EntityCollection<Booking> Bookings
{
    get
    {
        return ((IEntityWithRelationships)this).RelationshipManager.GetRelatedCollection<Booking>("HotelReservationSystemDBModel.FK_Booking_Customer", "Booking");
    }
    set
```
Listing 5. Entity classes generated by using the Entity Framework

namespace HotelReservationSystemServices
{
    public partial class HotelReservationSystemDBEntities : ObjectContext
    {
        /// <summary>
        /// Initializes a new HotelReservationSystemDBEntities object using the connection string
        /// found in the 'HotelReservationSystemDBEntities' section of the application
        /// configuration file.
        /// </summary>
        public HotelReservationSystemDBEntities() :
            base("name=HotelReservationSystemDBEntities", "HotelReservationSystemDBEntities")
        {
            this.ContextOptions.LazyLoadingEnabled = true;
            OnContextCreated();
        }

        ... public ObjectSet<Booking> Bookings
        {
            get
            {
                if ((_Bookings == null))
                {
                    _Bookings = base.CreateObjectSet<Booking>("Bookings");
                }
                return _Bookings;
            }
        }
        private ObjectSet<Booking> _Bookings;

        public ObjectSet<Customer> Customers
The .NET Framework Implementation of the Hotel Reservation System

```csharp
private ObjectSet<Customer> _Customers;
...
Listing 6. Context generated by using the Entity Framework

For comparison purposes with Listing 5, Listing 7 shows the equivalent Java `Customer` entity class generated by using the JPA tools with NetBeans.

```java
package HotelReservationSystemEntities;
...
@Entity
@Table(name = "Customer")
public class Customer implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "CustomerID")
    private Integer customerID;
    @Basic(optional = false)
    @Column(name = "ForeName")
    private String foreName;
    @Column(name = "MiddleInitial")
    private String middleInitial;
    @Basic(optional = false)
    @Column(name = "LastName")
    private String lastName;
    @Basic(optional = false)
    @Column(name = "DateOfBirth")
    @Temporal(TemporalType.DATE)
    private Date dateOfBirth;
    @OneToMany(mappedBy = "customer")
    private Collection<Booking> bookingCollection;
...
    public Customer() {
    }
    public Customer(Integer customerID) {
        this.customerID = customerID;
    }
```
The .NET Framework Implementation of the Hotel Reservation System

```java
public Customer(Integer customerID, String foreName, String lastName, Date dateOfBirth) {
    this.customerID = customerID;
    this.foreName = foreName;
    this.lastName = lastName;
    this.dateOfBirth = dateOfBirth;
}

public Integer getCustomerID() {
    return customerID;
}

public void setCustomerID(Integer customerID) {
    this.customerID = customerID;
}

public String getForeName() {
    return foreName;
}

public void setForeName(String foreName) {
    this.foreName = foreName;
}

...

public Collection<Booking> getBookingCollection() {
    return bookingCollection;
}

public void setBookingCollection(Collection<Booking> bookingCollection) {
    this.bookingCollection = bookingCollection;
}

...
```

Listing 7. JPA Customer entity class generated by using NetBeans

The Entity Framework is built on top of ADO.NET, and as such it can connect to any data source for which an ADO.NET driver is available. In the Hotel Reservation System, the hotel information database is implemented by using Microsoft SQL Server®, and the Entity Framework connects to the database by using the native SQL Server driver, which implements the Tabular Data Stream protocol.

In any multiuser system that involves data management, data consistency is an issue. Most database management systems support consistency through the use of transactions. .NET Framework applications that access transactional data sources can employ TransactionScope objects that wrap the method calls that define the operations in a transaction. When control reaches the end of a transaction scope, the changes that are made to data held in various databases by these operations can either be committed or rolled back, depending on the outcome of the underlying transaction. The CreateNewCustomer method in the WCF implementation of the CustomerService class
in Listing 2 showed an example of using a **TransactionScope** object; the pertinent
details are reproduced in Listing 8.

```csharp
public class CustomerService : ICustomerService
{
    public int CreateNewCustomer(string foreName, string middleInitial, string lastName, 
        DateTime dateOfBirth)
    {
        ...
        using (TransactionScope transaction = new TransactionScope())
        {
            try
            {
                // Create a new Customer entity object
                customer = new Customer
                { ... }; 

                // Save the Customer entity to the database 
                ...
                transaction.Complete();
            }
            ...
        }
    }
}
```

**Listing 8. A TransactionScope object delineating a transaction**

The Java version of the Hotel Reservation System made use of the Spring Framework to
implement common design patterns and reduce close-coupling and dependencies
between components. A .NET Framework version of the Spring Framework (**Spring.NET**) 
is available from SpringSource, but designers and developers building .NET Framework
applications should also consult the patterns and best practices that are documented on
the **Microsoft Patterns and Practices** Web site. A key resource that is worth examining
on this site is the **Microsoft Enterprise Library**, which provides a set of prebuilt modules that
are designed to aid developers facing common enterprise development challenges. The 
Enterprise Library includes the **Unity Application Block**, which is an extensible
dependency injection container that supports constructor injection, property injection,
and method call injection.

**Customer Access**

Like the Java implementation, customers can connect to the Hotel Reservation System
and make bookings by using a Web browser or a **Windows Phone 7** device. Customers
accessing the system through a Web browser run a Microsoft Silverlight® application,
which provides a graphically appealing user interface, and acts in a manner similar to
that of the JavaFX interface in the Java solution. The application that runs on Windows Phone 7 devices is also Silverlight-based. It provides the same functionality as the browser application, but it is adapted for the form factor of the mobile device.

The Web browser and mobile applications both connect to the IIS server that the hotel hosts. IIS supports several models for implementing content, including static HTML, CGI, and dynamically generated pages. The key Microsoft technology in this area is **ASP.NET**, which provides functionality similar to that available with JSP pages and servlets, but with several extensions. Developers can use the **ASP.NET Model-View-Controller (MVC)** framework to implement complex Web applications. Additionally, Web applications can invoke WCF services to retrieve data and perform the various business operations that the system implements.

**Reception Staff Access**

Hotel reception staff run a desktop application that is implemented by using **Windows Presentation Foundation** (WPF) to provide the user interface. WPF enables you to separate the layout of the user interface from the code that provides the user interface logic through the use of data binding. Visual Studio provides a GUI design window that a developer can use to lay out a user interface. Alternatively, a specialist graphics designer can define a complex user interface by using a separate tool such as **Microsoft Expression®**. The user interface for a WPF application is defined by using a language called XAML. XAML employs an XML-like syntax to describe the elements of a user interface, and the relationships between them. Figure 8 shows an example of the XAML markup and layout for part of the user interface, in the Visual Studio design view window.
You can implement the user interface logic by using a programming language such as C#; like Swing, a WPF user interface is event-driven and you write code that responds to the various events that can occur as the user interacts with the various user interface elements. Data binding is a powerful mechanism that enables a developer to declaratively associate the visual elements in a user interface with sets of data, and display this data in these elements. As the data changes, the user interface automatically updates.
The WPF application connects to the WCF services that IIS and WAS host to retrieve data and perform the business operations. The WPF application and the WCF services are both located on the same site (the hotel), so the WPF application can connect to the WCF services by using TCP/IP rather than HTTP. A key feature of WCF and WAS is the ability to expose multiple endpoints for a service, with each endpoint associated with a different network protocol.

**Reporting**

The Reporting application follows a very similar pattern to that employed in the Java solution. It is implemented by using the Microsoft Office suite, connecting to the SQL Server database holding the hotel information to extract data. Figure 9 shows an example of a Microsoft Excel® spreadsheet being configured to retrieve data from a SQL Server database.

![Figure 9. Connecting an Excel spreadsheet to SQL Server](image)
Batch Processing

The batch processing infrastructure of the Hotel Reservation System consists of a series of stand-alone C# applications (some written to support a specific marketing campaign, others created to support regular, repeated campaigns) that post requests to a queue in Message Queuing (also known as MSMQ). A custom Windows service is scheduled to run at off-peak hours, retrieve the jobs from this queue, and send them to the appropriate batch processing service running under WCF. As with the message-driven beans in the Java solution, there is a collection of these services, each one developed to support a specific batch job.

Integration with Third-Party Systems

WCF services can be exposed internally inside an organization (like EJB session beans), but they can also be made available through HTTP endpoints to the outside world as Web services. The Hotel Reservation System follows this approach to implement the Web services that enable third-party Web client applications and brokerage services to query room availability and pricing.

Finally, to support the secure payment and bookings transfer requirements from partner hotels, the Hotel Reservation System implements a set of business orchestrations and data transformations by using BizTalk Server. BizTalk Server includes a collection of Visual Studio tools and templates for defining orchestrations and transformations, as shown in Figure 10. BizTalk Server also provides several connectors and adapters for sending data to, and receiving data from, a variety of systems.
Nonfunctional Requirements of the Hotel Reservation System

The discussion so far has concentrated on the functional implementation of the Hotel Reservation System, but the nonfunctional requirements of any enterprise are equally important. These requirements include security, performance and scalability, system reliability, and management and monitoring.
Nonfunctional Requirements of the Hotel Reservation System

Security

Security is a somewhat nebulous term that covers a variety of requirements concerning protecting resources on a computer and limiting access to data and operations that a user can request while running an application. The security features that are provided with the .NET Framework and Windows services include:

- **Code signing.** A .NET Framework application consists of a series of executable files called assemblies—an assembly is similar to a JAR file in a Java application. A developer can sign an assembly with a private key, and distribute the corresponding public key to each application that references the assembly. When the application loads an assembly, the CLR automatically verifies the signature of the assembly by using the public key to establish the provenance of the assembly. If the verification fails, the assembly is deemed to be suspect and is not loaded, and the application throws an exception.

- **Sandboxing.** To protect the resources on a computer from rogue code, a .NET Framework assembly that is downloaded as part of an ASP.NET Web application, or an application that is invoked from a remote network location, can be executed in a restricted environment that limits the operations that it can perform, such as writing to files on disk. An administrator can configure the permissions that are available to a sandboxed application.

- **Role-based security.** This is an authorization mechanism. You can annotate the methods that make up an application, or that implement the operations in a service, with attributes that specify whether a user or role has access to those methods. When a user runs the application or invokes an operation, the CLR checks the user’s identity against these attributes and only performs the method if there is a match, throwing an exception otherwise.

**Note:**

Attributes in the .NET Framework are similar to annotations in Java. You can use them to specify additional metadata or indicate nonfunctional cross-cutting concerns of a class or method. The .NET Framework implements a large number of attributes that address a variety of concerns, and you can also define your own custom attributes.

The .NET Framework also provides a series of APIs that an application can employ to implement a finer grain of access control inside methods. Role-based security works in conjunction with most Windows authentication mechanisms.

**Note:**
Nonfunctional Requirements of the Hotel Reservation System

If you implement a claims-based federated security model, you can incorporate Windows Identity Foundation into your solution.

- **Transport-level security.** This is a feature of IIS and WAS, and enables a host environment to limit access to the services running in that environment based on the identity of the user or process invoking the service. Transport-level security is transparent to the service.

- **Encryption.** Information passing between a client application and a service can be encrypted to reduce the chances of tampering by a third party who is monitoring the network. WCF services support encryption at the message level and the transport level, and it is transparent to the code running in client applications and WCF services.

Performance, Scalability, and Reliability

Windows Server® 2008 is an excellent base for enterprise-scale applications. It provides several services that you can use to implement a highly scalable and reliable platform. These services include:

- **Failover clustering.** This service can build redundancy into a system and help to eliminate single points of failure.

- **Network Load Balancing.** This service enables you to distribute TCP/IP requests across multiple servers to optimize resource utilization, decrease computing time, and ensure system availability.

- **Windows Server AppFabric.** This service provides a scalable and reliable hosting environment for persistent WCF workflow services, and a distributed cache solution. AppFabric implements a clustered environment for hosting workflow services, which can be persisted to a database at key points in their execution. If the service or its host environment subsequently fails, the service can be resumed on another node in an AppFabric cluster from the point at which it was last persisted. The distributed cache helps to improve performance by keeping cached data local to the workflow services that use it.

.NET Framework applications can make use of logging and diagnostic functionality that is built into the Windows operating system and provides features that are analogous to those found in the Java logging API and the Log4j framework. An administrator can configure a .NET Framework application to record diagnostic information about significant events and failures to event logs and files. The **Logging Application Block** in the Microsoft Patterns and Practices Enterprise Library illustrates best practices for incorporating logging functionality into a .NET Framework application. The logging
support in the .NET Framework is highly extensible, and you can implement your own custom log sources if necessary.

Management and Monitoring

Windows also provides several built-in features that an administrator can use to monitor the performance of applications. The most immediately accessible of these items is Windows Performance Monitor, which enables an administrator to collect key performance information about an application, service, or computer, such as the volume of input/output (I/O) being performed, the amount of memory that an application is consuming, or the CPU time that is being consumed. Performance Monitor can generate graphs and reports with this information to help analyze the reasons for any bottlenecks that may occur. Figure 11 shows an example of Performance Monitor tracking the CPU utilization and volume of physical disk I/O occurring on a computer.

![Performance Monitor](image)

**Figure 11. Windows Performance Monitor**

For enterprise and large-scale data center environments, an administrator can use Microsoft System Center Operations Manager to maintain the performance of computers and services. System Center Operations Manager enables you to examine the health of
an enterprise environment, trigger alerts if available resources drop below specified thresholds, and run scripts that can take the necessary corrective action before a situation becomes critical.

**Visual Studio Profiling**

To help locate the source of potential bottlenecks in an application and reduce the chances of an application exhibiting consistent poor performance, an application developer using Visual Studio can make use of application profiling. The Visual Studio Profiler attaches to an application running in Debug mode and periodically inspects the state of the application, noting which methods are being invoked and collecting other data concerning the health of the application. Using this information, Visual Studio can generate reports highlighting which methods are being invoked most frequently. Visual Studio also supports probing; it can generate instructions that record when each method starts and finishes, enabling it to generate a report that indicates how long each method call takes.

**Summary**

The .NET Framework, together with tools such as Visual Studio, provides a comprehensive development environment for quickly designing, building, and deploying applications that run on the Windows platform. Using the .NET Framework, you can construct the whole gamut of applications, incorporating mobile, desktop, and enterprise technologies according to the requirements of your organization.

In this paper, you have seen how the .NET Framework implementation of the Hotel Reservation System maps to the corresponding system built by using Java. This is just one possible rendition, but it illustrates how the .NET Framework in conjunction with other services available from Microsoft can be combined into a fully functional solution.

The .NET Framework provides an optimized and integrated environment for Windows, enabling you to more easily build applications that combine disparate components and communicate with services across the network, minimizing the need to incorporate technologies from third parties. Guidance such as that available on the Microsoft Patterns and Practices Web site describe best practices for designing solutions based on the .NET Framework. Additionally, tools such as Visual Studio that are designed specifically to work with the .NET Framework can greatly enhance developer productivity when building complex solutions.

For more information about building applications by using the .NET Framework, visit the .NET Framework Developer Center.