

DevForce Overview

DevForce is a powerful .NET framework that accelerates the development of data intensive enterprise applications. DevForce provides components that extend .NET and simplify data access, improve user-interface design, and enable rich applications to operate over the internet.

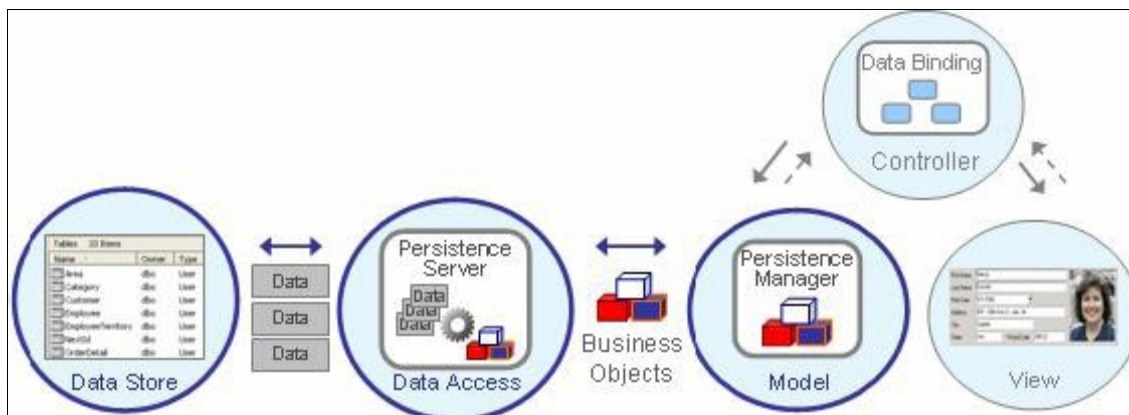
Some of the key components comprising DevForce include:

- Object-Relational Mapping – Simplifies data access by providing an object-oriented interface to databases, which eliminates the requirement for developers to write complex SQL queries.
- Persistence Management – An enterprise class data access layer that manages how objects are fetched, saved, merged, cached, or stored for offline access.
- Declarative UI Data Binding – Simplifies UI development by synchronizing the user-interface with the business objects and eliminates the need for complex event-driven code. Also handles data formatting, validation, error conditions, and broken rules.
- RAD Tools – A comprehensive tool suite that plugs into Visual Studio and allows developers to build business objects and populate the user-interface quickly and easily.
- Business Object Server – Allows applications to be deployed in a scaleable n-tier environment, enforces security, enables centralized monitoring, and allows WinForm applications to operate over the internet.

The following sections of this document provide information regarding the key features of the DevForce framework, and highlight the value of these features to enterprise application development.

Layered Architecture and Best Practices

DevForce supports a five-layer application architecture, as shown below, and makes it easy to separate the functionality into tiers.



- *Data Store Tier*
The DevForce framework supports all industry-standard relational database management products. A single DevForce application can access multiple databases from multiple database vendors. DevForce also supports views, stored procedures, triggers, custom SQL, and cross-database transactions with two-phase commits using Microsoft Enterprise Services.

Supported Databases:

- Microsoft SQL Server
- Oracle
- Sybase ASE
- IBM DB2
- Informix
- Any ANSI 92 & OLEDB supported database

- *Data Access Tier*
DevForce object relational persistence supplies the data access tier. It manages the flow and synchronization of data between the entities in the business model and their “homes” in data storage.
- *Business Logic (Business Object Model) Tier*
The DevForce business object model encapsulates the rules and processes that apply to the business domain of an application. DevForce creates business objects (entites) that are mapped to the underlying data store; these objects are then extended by the developer with custom business logic. Application developers write code using a standard object model instead of accessing the data store directly (although direct access is still possible if needed).
- *UI Tiers*
The DevForce UI data binding architecture facilitates data exchange and synchronization between the entities in the business object model and controls on the screen. It promotes a Model-View-Controller architecture and enables the separation of model from the UI view and controller classes.

In short, DevForce follows best practices in the industry and provides much of the plumbing and infrastructure required to build a distributed .NET application. The framework encourages good design patterns, decouples logic into a model-view-controller architecture, and allows development teams to focus on solving their business problems, not technology problems.

With DevForce, developers are not "boxed in" by a vendor specific technology. It integrates seamlessly into Visual Studio and extends .NET instead of hiding it. Even if a particular project's requirements have not been anticipated, the framework provides developers with a way to get the job done, whether it is using one of DevForce's extensible interfaces, or going completely around the architecture and using .NET directly.

Object Oriented Persistence

Perhaps the most fundamental challenge of any application is retrieving and saving data properly. Most contemporary systems store data in relational databases. At the same time, most contemporary applications are also developed in an objected-oriented fashion. However, relational and object models are separate and incongruent worlds. Databases consist of tables, rows, columns, and keys. By contrast, objects consist of properties, methods, rules, and relationships. Compounding these dissimilarities are associated differences in design principles, tools, skills, and staffing.

DevForce Object Mapping is an insulating layer bridging these two worlds, a technology that moves data between the database tables and the objects without the developer's awareness of the details. At the start of a project, developers use the DevForce Object Mapper to declare a mapping between Business Objects and their corresponding data source objects such as relational database tables and columns. They then extend the generated Business Object classes with application-specific logic. They continue building the application in an object-oriented fashion, navigating the object graph using standard dot notation and writing little or no SQL.

Studies show that development organizations who build their own ORM layer spend roughly 40% of application development and maintenance on that component alone. The DevForce ORM is a robust and production-tested solution that is fully integrated with Visual Studio .NET and works with any commercial relational database.

High performance Caching and Offline Execution

DevForce caches objects on the client without obvious effort from the developers (and users). When the application requests data, either explicitly with an entity query or implicitly via the object dot notation (e.g., "Customer.Orders"), the framework looks for the objects first in cache and then, only if necessary, on the server. Of course the developer may optionally force a query to go to the database.

The application may be designed to operate offline. The application first fills the cache from the database, perhaps with a set of customer objects. The application is disconnected and continues operation, confining queries to the cache and using the cache as a temporary store for additions, changes, and deletions. If the user exits the program, the cached objects can be saved to disk and restored when the application resumes. When the application reconnects with the host, pending modifications can be posted back to the database and fresh objects retrieved. Transactional support helps resolve conflicts with competing information that was posted to the database in the interim.

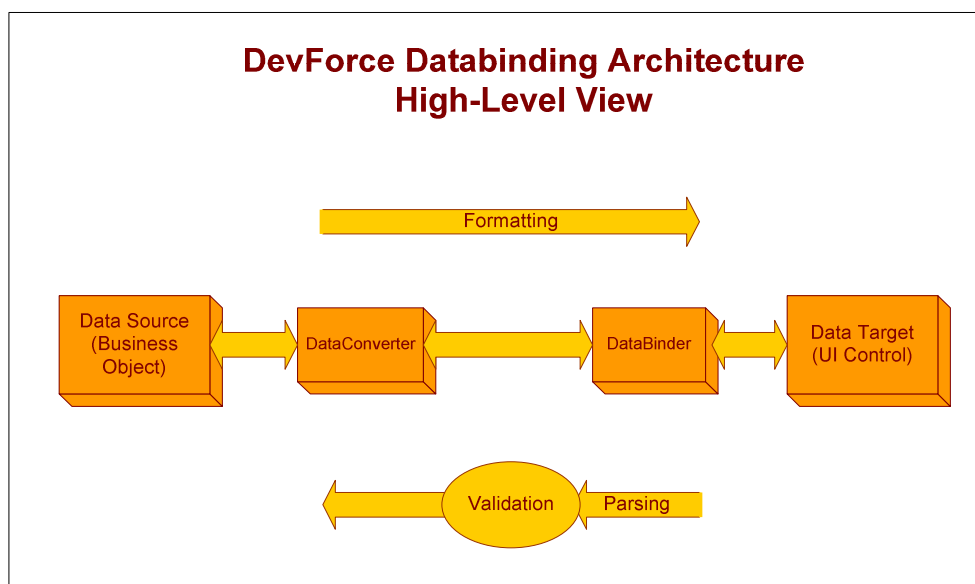
These features make it easy to insulate the application from transient connection interruptions and server shutdowns -- breakdowns that would typically abort a thin client session with a complete loss of the user's work.

Extensible UI Data Binding Framework

Most business applications are UI intensive with as much as half of the development effort devoted to making the user interface intuitive and effective. Synchronizing UI controls with their supporting application objects is the most time consuming and error prone part of UI development.

DevForce UI data binding automates the linkage between control and object so that developers can graduate to less tedious and higher value UI development. Data binding is essential for applications with sophisticated user interactions and, accordingly, .NET offers some functionality in this area. DevForce builds upon this basic infrastructure to provide a full object-oriented API for databinding and adds supplemental functionality such as support for grids, deeply nested properties, live list synchronization, business object metadata, and offers full validation/verification functionality that supports concepts such as ‘broken rules’. DevForce also integrates support for databinding to third party control suites such as Developer Express and Infragistics and provides an API to integrate other control suites or custom controls as desired.

The DevForce UI Databinding tool makes it easy for developers to declare a linkage between a business object property and a control on the screen, and will even autopopulate the controls onto the form. From then on, DevForce will make sure that the UI control stays synchronized with the business object. If the UI changes, the business object will be updated. If the business object changes, the UI will be updated. This is particularly convenient when a business object property is displayed in multiple locations in an application and saves the developer from having to update all of the modules that refer to that property.



2-tier and n-tier Deployment

While a DevForce application is inherently n-tier from a logical point of view, these tiers may be physically deployed in a variety of ways: single-tier (one machine), client-server, or n-tier. During development, it is convenient for the developer to host all of the tiers on one machine. However, the final deployment may be a full n-tier system traversing the internet.

DevForce simplifies this problem by minimizing the number of changes to the application when moving from 1- or 2-tier to n-tier. In fact, the DevForce API for n-tier is identical to that of 1-tier and no code-level changes are required to change the configuration. The deployment topology is controlled by a simple configuration file and can be modified after the application has been compiled or even installed.

Business Object Server

Most client-server applications interface directly with the database. There are scalability and security penalties in this approach; an intermediate, database “broker” layer is widely preferred. In DevForce, the client-side application exchanges objects with a stateless Business Object Server (BOS) running on one more central server machines. The BOS provides functionality that parallels a J2EE Application Server in a thin client architecture. It alone talks to the database, manages transactions, provides security, and executes server-side-only business logic. The client-side application issues no SQL and has no knowledge of the connection or connection string used to access the database.

The Business Object Server (BOS) is a middle-tier mediator between clients and networked resources such as databases and web services. It is a software component that executes on a server within a hosting application. The host can be any application that can execute managed code on the .NET 2.0 Framework.

The BOS’ primary job is to provide clients with business objects. It can also serve as a gateway to other kinds of services and resources that, for security or performance reasons, should execute on a server rather than on a client.

From the perspective of an application developer, here are some of the reasons businesses choose to implement the BOS:

Need	Reason
Security	<p>We must ensure confidentiality of transmitted data, authenticate each communication, and impose application-specific authorization logic that is invulnerable to client-side hacking.</p> <p>We want a consolidated security solution for all requests whether directed at a database or a web service.</p>
Performance over WAN	<p>We need to support users who connect over a wide area network (WAN) or who connect to our LAN via VPN.</p>
Access Everywhere	<p>We have business partners outside the firewall who are not authorized to access our network; we cannot give them VPN access. Such clients should be able to acquire data and services over port 80 (HTTP) or 443 (HTTPS).</p>

Limited Network Bandwidth	Some clients have low-bandwidth connections or lose their connection. We must limit the number of transmissions, compress the data transmitted, and be able to continue to operate when we lose the connection.
Central Monitoring	We need to track server and client application activity including data source operations such as queries and saves.
Web Services	We need centralized control over client access to web services so we know what clients are doing and that they are authorized to do it. We want to publish a subset of our business object model as a web service to both .NET and non-.NET clients.
Hosted Services	While much of the application runs best on the client, some business logic and services must run on the server either because the logic is proprietary or because the resources are scarce and cannot be distributed to clients.
Software-As-A-Service and Software Trial Options	We could reach a larger market if we could “rent” our product to customers who cannot or do not want to host it themselves. We could increase sales and shorten our sales cycle if prospects could experience it hands-on, online.
Server Push	Certain data can change at any moment and we want the server to tell our clients about those changes immediately.

ASP.NET Browser-Based Applications

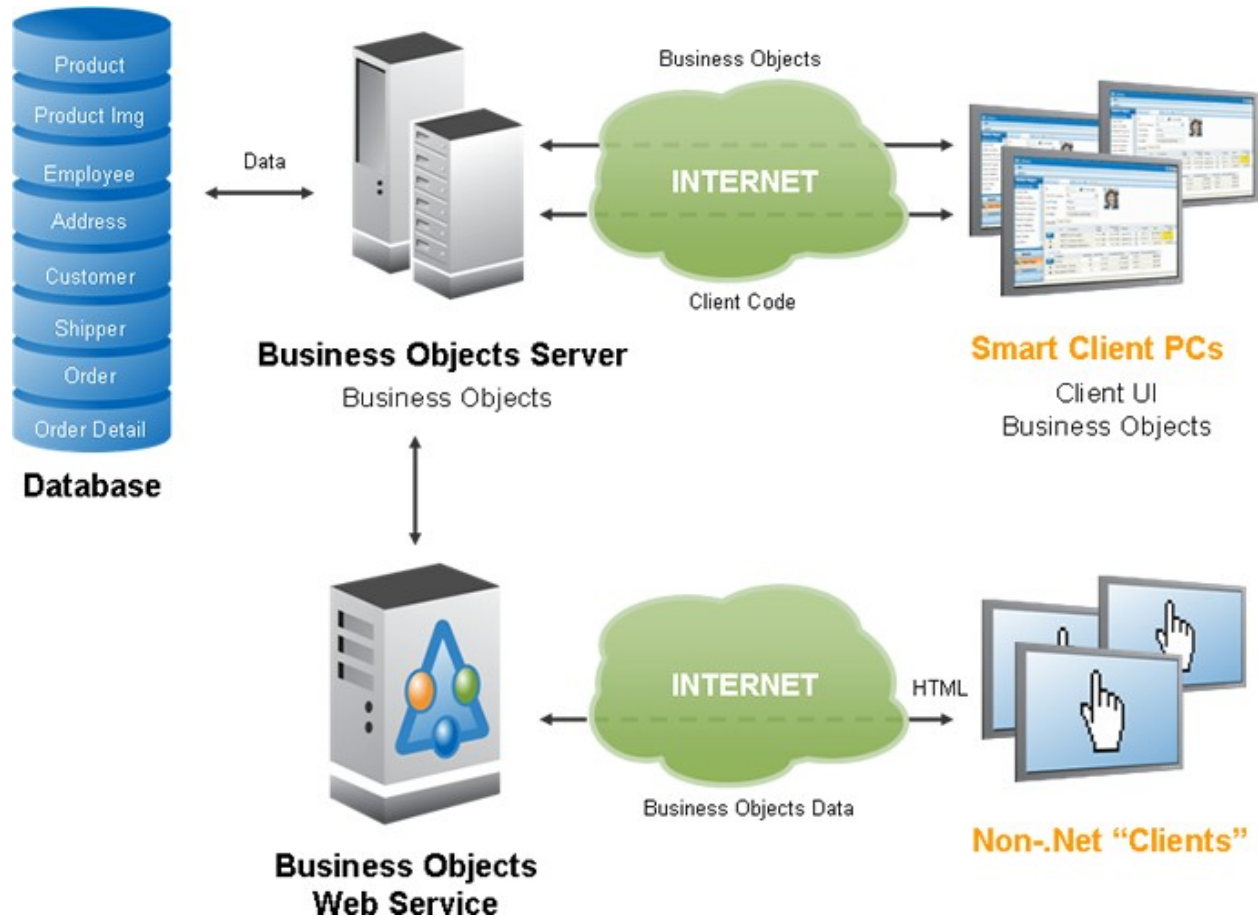
While DevForce focuses on delivering functionality to support WinForm smart client applications, the framework also offers a number of features that can help accelerate and enhance a browser-based application. In particular, building the data access infrastructure for an ASP.NET WebForm application can be a significant challenge. The object-relational impedance mismatch problem still exists, as well as performance, scalability, and standardization requirements.

Developers of a WebForm application should be able to write their application against a clean business object model instead of having to write individual SQL statements against a relational database. DevForce provides an elegant object model infrastructure that standardizes the way in which business objects are retrieved or saved across a development organization. The DevForce caching infrastructure also saves developers time by keeping track of the business objects that are being used by each user; this information can then be saved to session state. This can also dramatically improve performance and scalability by not having to retrieve the same data from the database on each page refresh. After using DevForce persistence layer to retrieve the objects, the developer can then bind the objects to any WebForm and the data displayed to the user.

Hybrid Topology with WinForm and WebForm Applications

When developing an application that will have both a WinForm and WebForm UI, there is a compelling advantage if each view can share the same business objects and business logic. The DevForce business object infrastructure provides this. As opposed to duplicating the logic in each implementation and maintaining two sets of data and logic that need to be kept synchronized and consistent, there is just one business object model that encapsulates the domain logic.

Casual end-users will be able to access the application through a browser. Power users will enjoy the responsiveness and functionality of a WinForms application that works across the internet. Both users will access identical business logic, keeping data and processes consistent throughout both applications and reducing the coding and maintenance time of the development organization.



Feature List

Full Visual Studio Integration

- Object Mapping Tool
- UI Mapping and Design Tools
- Business Object Server

Object Mapping / Persistence Features

- Optimistic Concurrency
- Transactions
- Distributed Transactions
- Stored Procedures
- Triggers
- Cross-database Relations
- Support for Complex Schemas
- User-defined Columns
- Enumerations
- Identity Columns
- Custom ID Generators
- Remote Procedure Call
- Pass-through SQL
- Asynchronous Queries
- Span Queries
- Web Services Support
- 1-tier, 2-tier, and n-tier Internet deployment without recompiling

Business Object Features

- Mobile Business Objects that move across the internet
- Client-side Caching
- Disconnected Access
- Cache Coherency Protocol
- In-memory Checkpointing
- Object Lifecycle Events
- Object Identity
- Referential Integrity
- Abstract Classes

UI Databinding Features

- Model-View-Presenter Architecture
- Advanced UI Databinding
- Managed Lists
- Dynamic Properties
- Nested Properties
- Grid Binding
- Validation
- 3rd Party Control & Grid Support
- Infragistics Integration
- Developer Express Integration
- Form Autopopulation

Security Features

- SSL 3.0 / Encryption
- Custom Login Managers
- Server-side Security Checks
- Auditing / Logging

About IdeaBlade

IdeaBlade Inc, is a leading supplier of .NET Framework tools and components for rapid development and deployment of enterprise Internet applications. IdeaBlade's DevForce framework, which extends the .NET Framework with class libraries, components, RAD tools, and a business object server, enables developers to focus on delivering business value while they shorten their development cycles by more than half. Founded in 2001, IdeaBlade has delivered .NET Framework development solutions to hundreds of independent software vendors (ISVs) and enterprises in many industries including: healthcare, insurance, finance, education, agriculture, biotechnology, and business services. IdeaBlade is a Microsoft Gold Certified Partner and member of the Microsoft Visual Studio Industry Partner (VSIP) program. For more information on IdeaBlade products and services visit www.ideablade.com.
