The **validation** infrastructure provided by DevForce is made up of a collection of interoperating validation components that are both easy to use and capable of handling sophisticated scenarios.

With DevForce, the developer can:

- **Generate validity checking** into business objects automatically via code generation and the DevForce EDM Designer extension.
- **Write rules of any complexity**. The developer can draw upon pre-defined rules (required value, range check, field length) or write custom rules of any complexity, including rules that compare multiple fields and span multiple objects.
- **Validate any kind of object**, not just objects that derive from base business classes.
- **Trigger validity checking at any time** such as upon display, before save, or when setting properties. The engine can fire “pre-set” to block critically errant data from entering the object or fire “post-set” to accommodate temporarily invalid values. The UI can inspect the engine for rules of interest, fire them, and adjust the display accordingly. It could color a text box, for example, or hide a portion of the form until applicable criteria were met.
- **Display a localized message** in the UI without special programming using localized resources delivered by a .NET standard or custom Resource Manager.
- **Classify validation results by type**. Code that inspects validation results could display just the “validation failed” message but it might also show warnings or “ok” messages and it might supplement the message be re-directing the application focus to the offending object and property. Each rule returns a rich, extensible object with all the information necessary for the developer to deliver a helpful response.
- **Discover rules in the code or retrieve them at runtime from a central store**. The engine automatically discovers rules in the code and can acquire rules defined externally in configuration XML, a database, or some other store of rules. The application can inspect, add, and remove rules at any time.
- **Leverage rules inheritance**. Rules defined in base classes propagate to their derived classes where they are “inherited” or overridden.
- **Adjust validation behavior** based on a custom validation context. The developer must have the flexibility to convey custom information to the validation process to cope with the variety of situational factors that arise in real applications.
- **Inspect and intervene as the engine validates**. The application can monitor the engine’s progress and interrupt, modify, or terminate a validation run at any point.
  We'll read more about these capabilities in the following pages.