Enterprise Library 概览

赵俊其

微软(中国)

平台和开发技术部

议程

- 什么是企业库
- 企业库能做什么
- 企业库的历史
- 企业库主要内容
- 企业库的未来发展
- 相关资源

企业库是什么

- 企业库是一组应用程序块(Application Block)所组成的库.用来帮助开发者解决企业级软件开发中遇到的通用问题和挑战

- 企业库提供了企业级应用中所用到的基础功能,覆盖了大部分的非功能性需求.例如数据存取,缓存,异常处理,认证,安全等等

什么是应用程序块(Application Block)

- 应用程序块(Application Block)是一个包括源代码的程序集合,这些程序集合共同完成一个功能
- 它包括了很多微软在.NET开发上的最佳实践.
- 开发者可以在项目中直接利用,也可以进行扩展,修改,并提供给社区使用

企业库是什么

企业库是…

- 一组应用程序块所组成的库,帮助开发者解决企业级软件开发中遇到的通用问题和挑战
- 一组用良好架构设计好的辅助类库
- 带有源代码的架构指导,开发者可以进一步修改,扩展
- 可以免费下载

企业库不是…

- .NET Framework的一部分
- 一个强制架构的应用程序框架
- 一个受微软支持的产品(兼容性,本地化)
- 出售

企业库能做什么

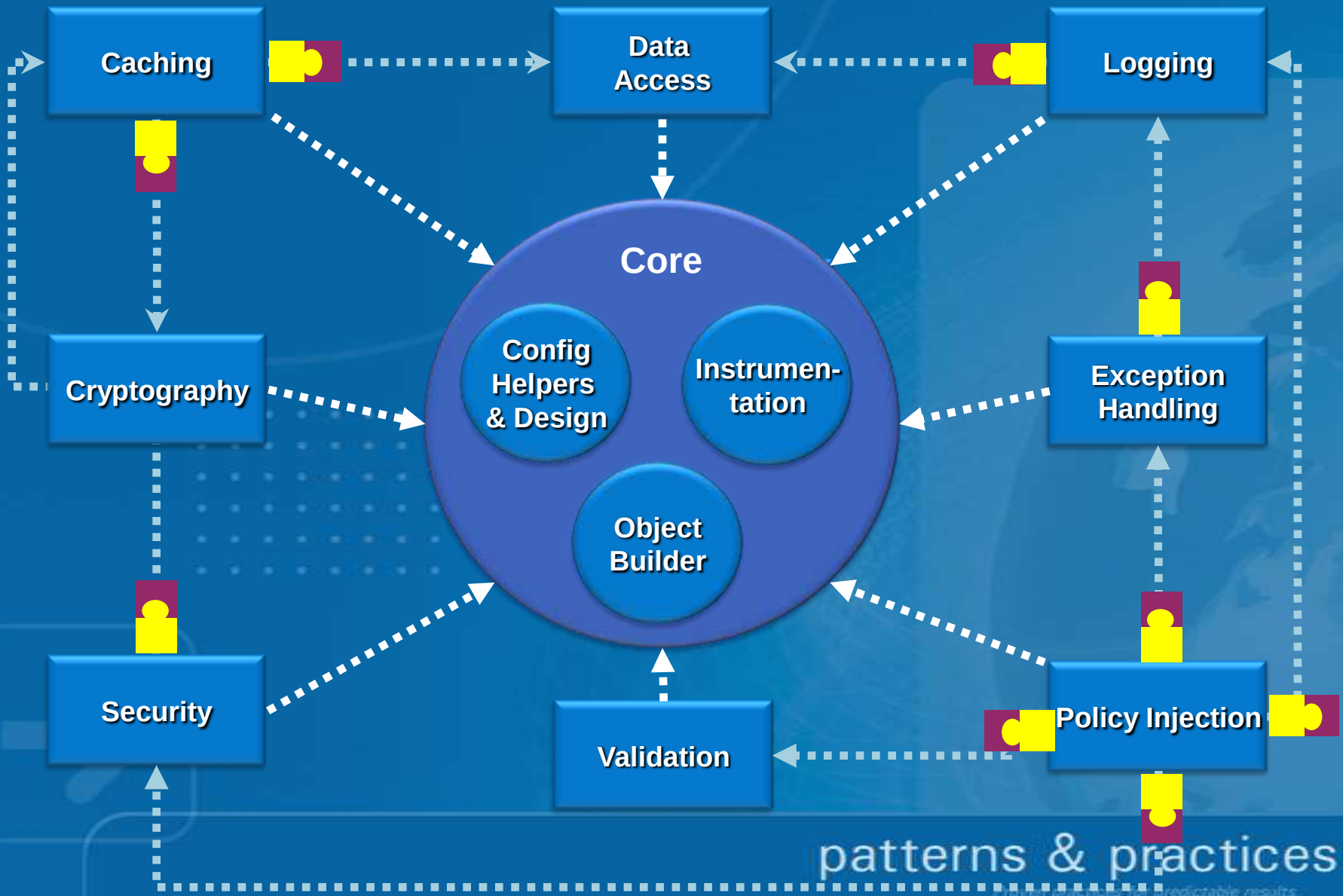
- 作为企业级开发的底层模块,为项目提供良好的基础
- 作为微软在.NET开发上的最佳实践,供架构设计和实现的参考.
- 扩展,修改它,为自己的应用服务
- 把自己的智慧融入企业库,并分享给更多人(开源社区)

企业库的历史

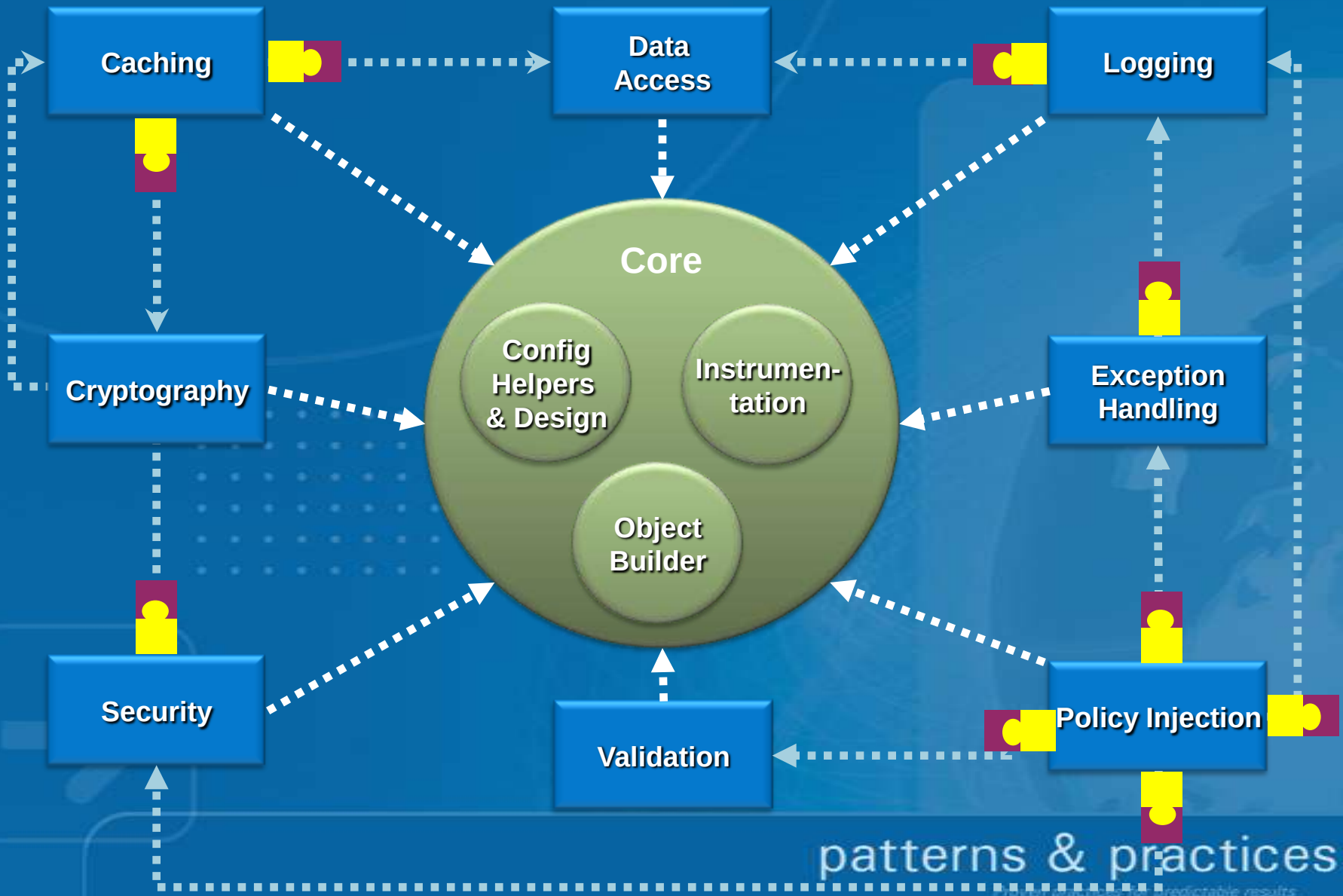
- 企业库的前身是Microsoft Application Blocks for .NET,是微软推出的开源项目

- Microsoft Application Blocks for .NET从最初只有Data Access Application Block一个发展到企业库3.1的8个Application Block,并且在企业库的4.0中还要加入更多的Application Block

企业库的历史


Caching
Data Access
Logging
Core
Config Helpers & Design
Instrumen-tation
Object Builder
Cryptography
Exception Handling
Security
Validation
Policy Injection
patterns & practices

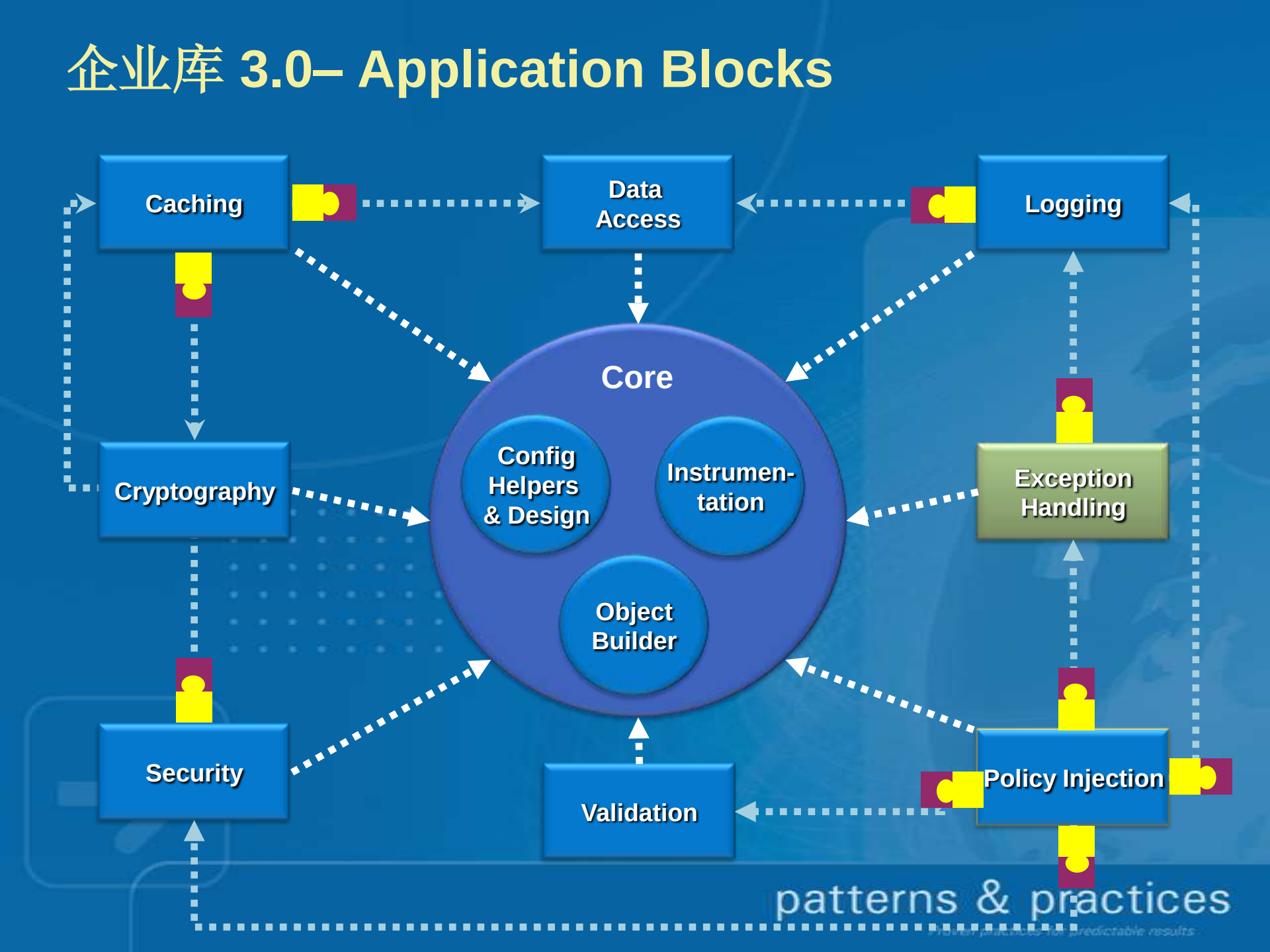
企业库 3.0 – Application Blocks



内核

- 配置
  - 每个Application Block都是可配置的
- 配置工具
  - 提供单独的配置工具,运行时和设计时配置
- 系统接口
  - 提供事件日志,性能计数器,WMI事件等功能
- Object Builder
  - 为各个Application Block创建组件

企业库 3.0– Application Blocks

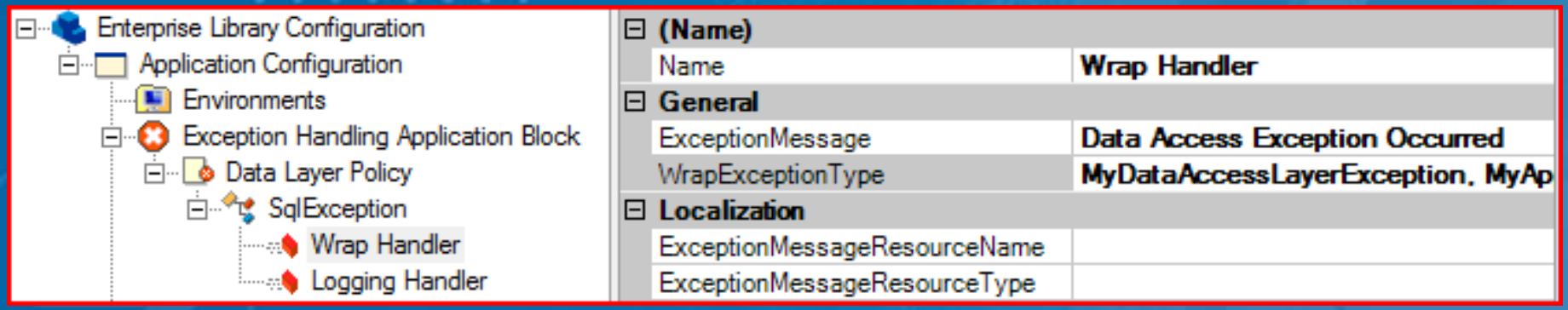


Exception Handling Application Block

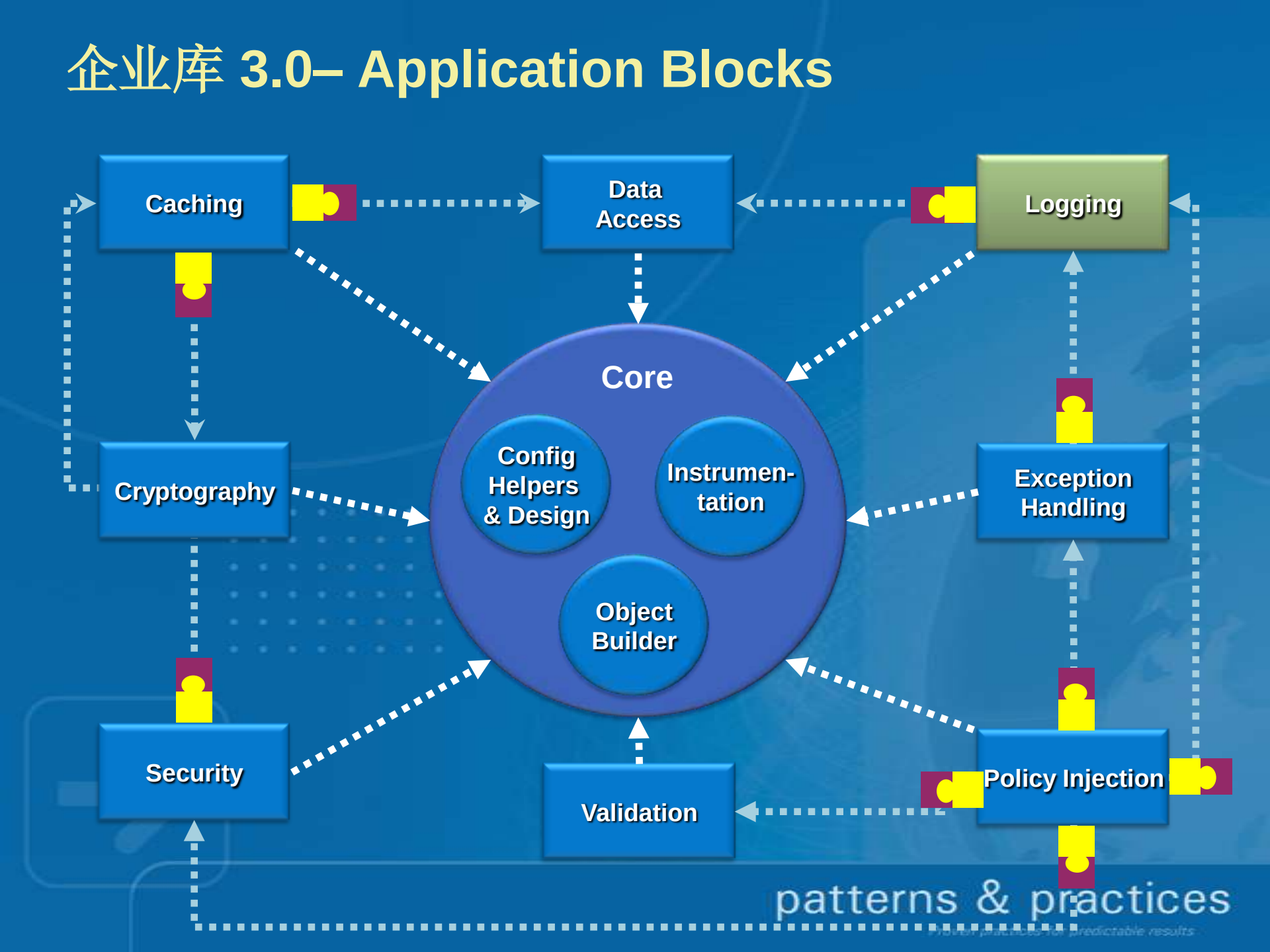
- 提供一种简单的,一致的异常处理机制
- 可以自定义异常处理策略,使得截获一个异常的同时引发另一个操作
  - 应用程序自定义的异常(ApplicationException)应当写日志
  - 数据库异常(SqlClientException)被截获后,被包装成另一个异常(DataLayerException) 并重新抛出
  - 安全异常(SecurityException)应当被截获,并被另外的异常(AccessDeniedException)替换,并抛出
- 提供如下内置操作
  - 写日志
  - 包装成另一个异常类型
  - 替换成另一个异常类型
  - 映射到 WCF Fault Contract
  - 用户自定义操作…

异常处理- 示例

```
Try
    ' some code that may throw
Catch Ex As Exception
    If ExceptionPolicy.HandleException(ex, “Data Layer Policy”) Then Throw
End Try
```



企业库 3.0– Application Blocks



Logging Application Block

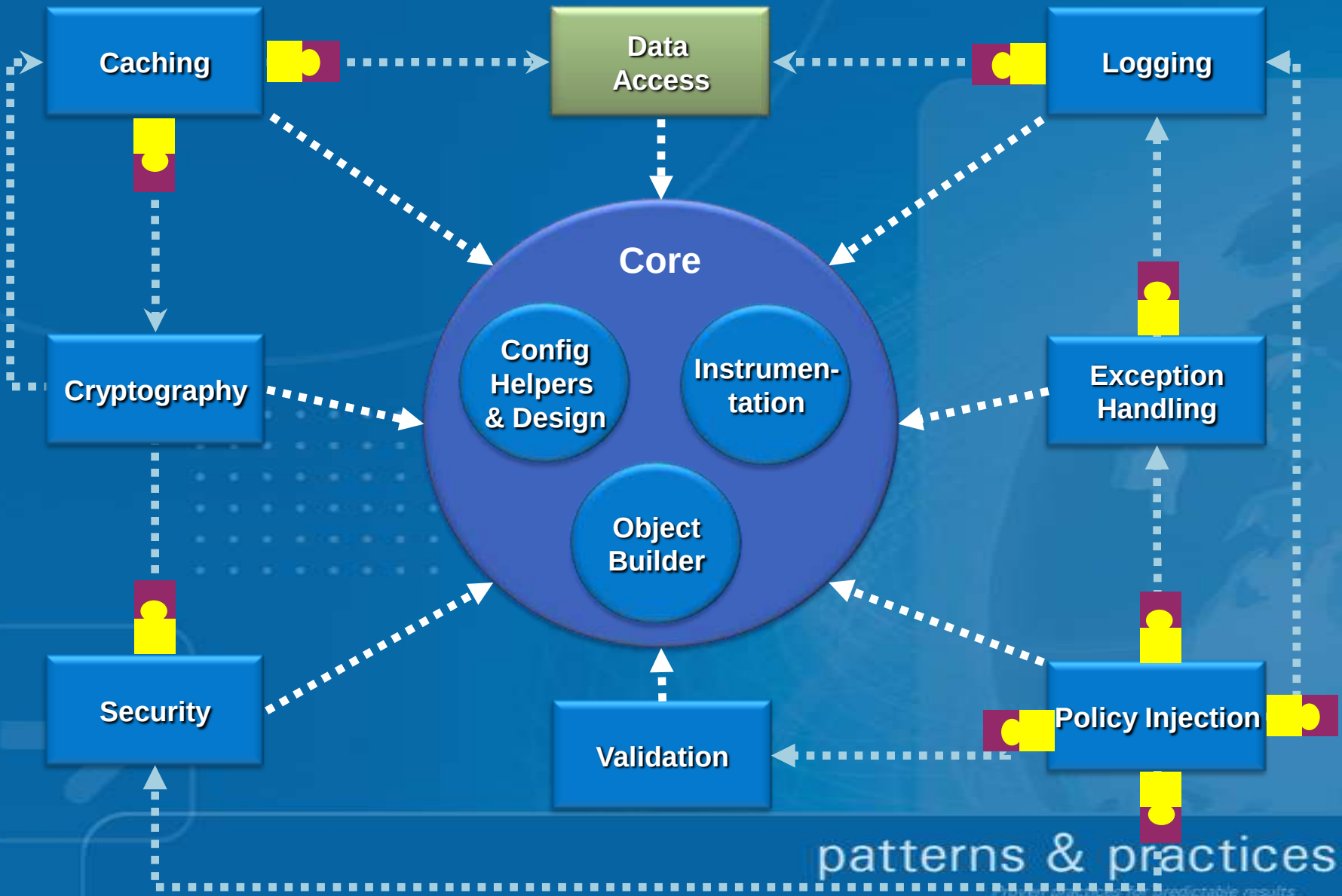
- 提供简单的日志模型
  - 强类型,可扩展的日志架构
- 在 System.Diagnostics 基础之上
- 配置驱动 – 运行时可配置什么信息被纪录
- 可以使用任何的 .NET TraceListener, 已包括如下已知的 Listener:
  - 事件日志, 数据库, 文件, MSMQ, E-mail, WMI, XML 或者自定义的 Listener

Logging – 示例

```
Dim log As LogEntry = New LogEntry
log.Message = “Your message here…”
log.Priority = 1
log.EventId = 100
log.Categories.Add("UI")
log.Categories.Add("Debug")
Logger.Write(log)
```

```
// Or if you prefer one line...
Customer cust = GetCustomer(123);
// Log the customer – will call cust.ToString() for the log entry
Logger.Write(cust, category, priority);
```

企业库 3.0– Application Blocks



Data Access Application Block

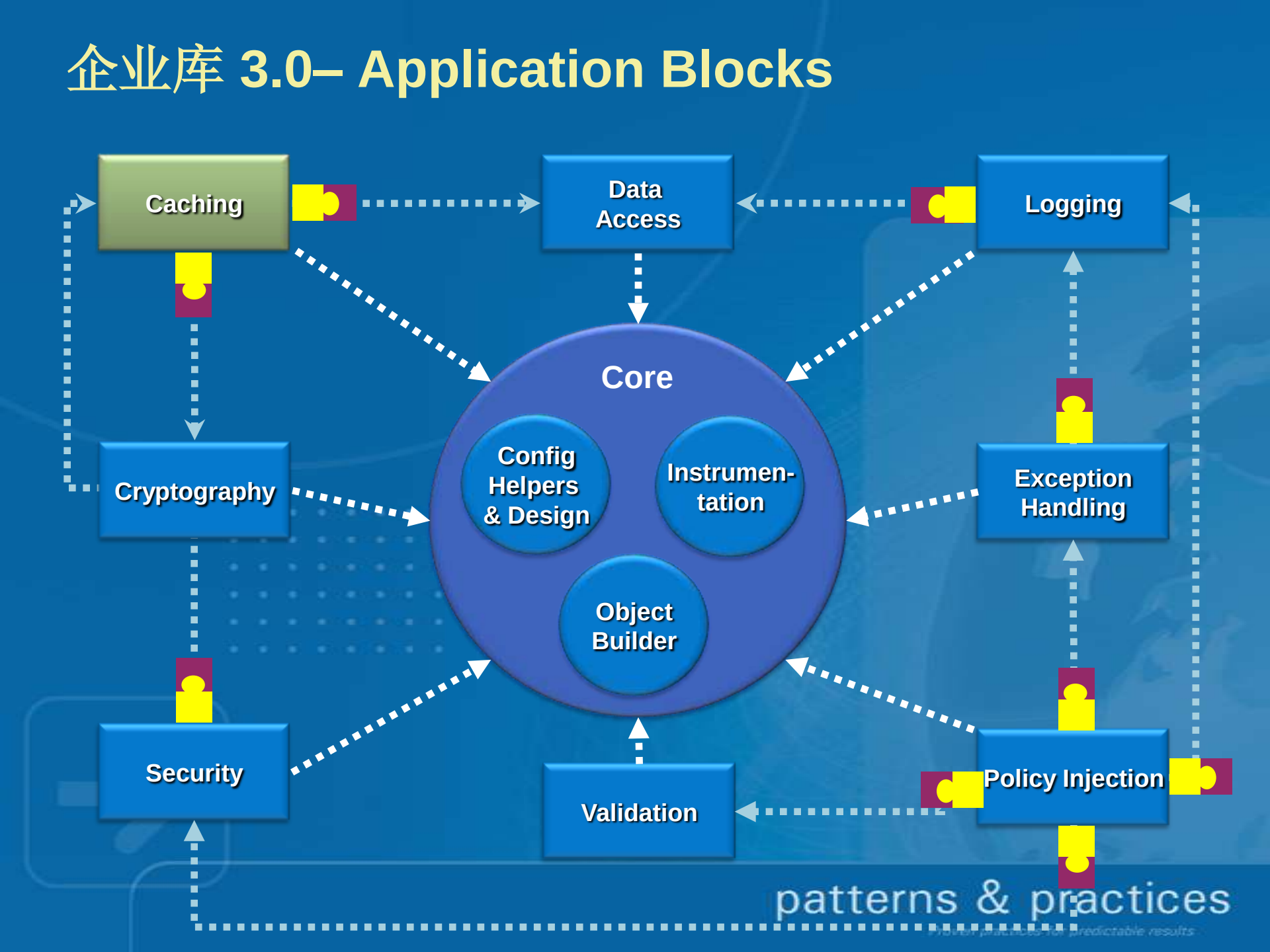
- 提供通过ADO.NET存取数据的最佳实践的简易模型
- 改善一致性
  - 适用多种数据库
  - 整合 System.Transactions 的事务
- 易用性
  - 一行代码调用存储过程
  - 使用 Block 管理数据库连接
  - 数据库连接字符串可配置(加密),或内置在的代码中

Data Access - Examples

```
Public Function GetProductsInCategory(ByVal Category As Integer) As DataSet
        ' Create the Database object, using the database instance with the
        ' specified logical name. This is mapped to a connection string in
        ' the configuration file
        Dim db As Database = DatabaseFactory.CreateDatabase("Sales")

        ' Invoke the stored procedure with one line of code!
        return db.ExecuteDataSet("GetProductsByCategory", Category)

        ' Note: connection was closed by ExecuteDataSet method call
End Function
```

```
public Dataset GetProductsInCategory(string connectionString, int category)
{
        // Create the Database object, using the specified connection string
        SqlDatabase db = new SqlDatabase(connectionString);

        // Invoke the stored procedure with one line of code!
        return db.ExecuteDataSet("GetProductsByCategory", category);

        // Note: connection was closed by ExecuteDataSet method call
}
```

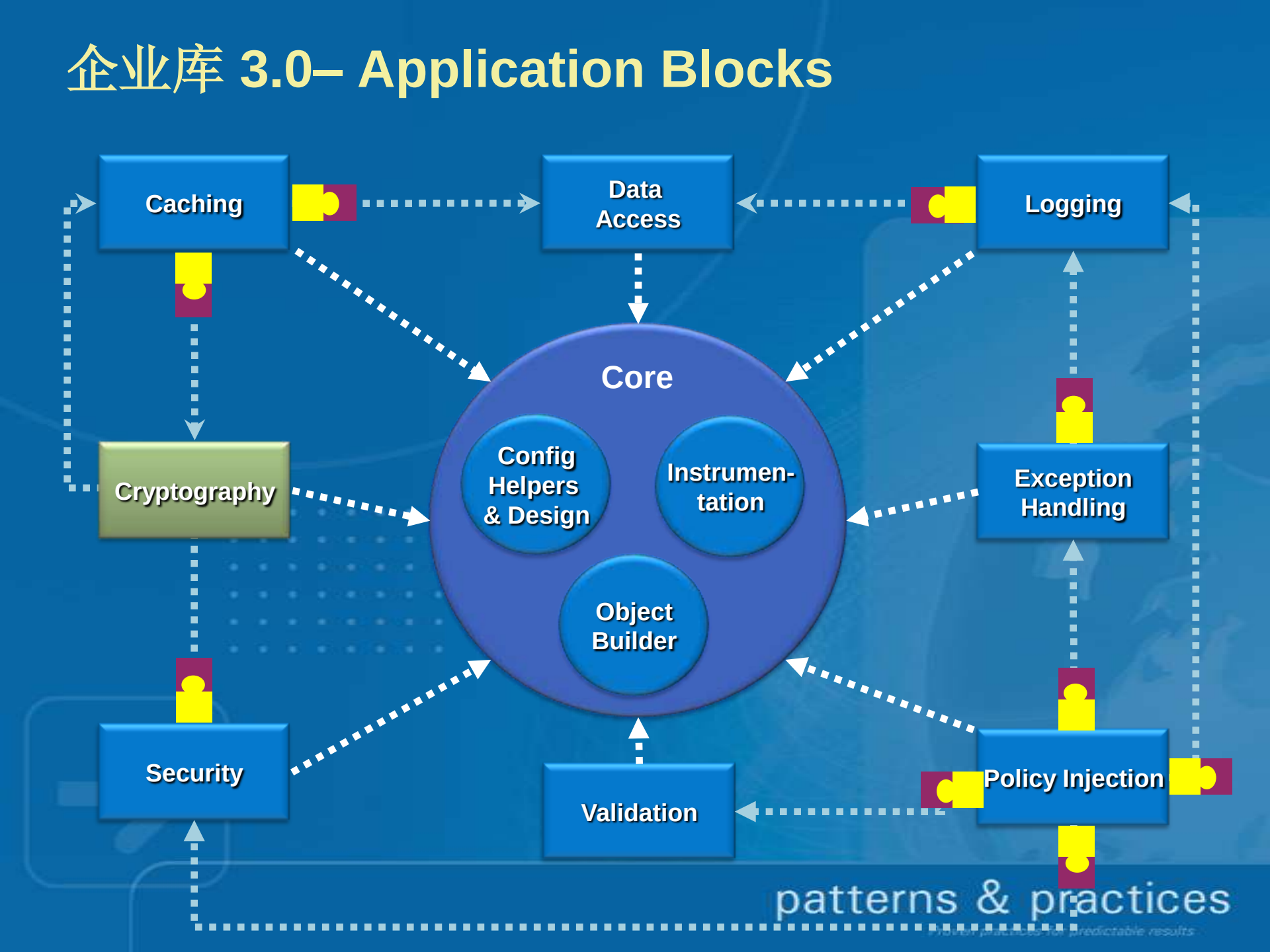
企业库 3.0– Application Blocks



Caching Application Block

- 提供可应用于多层的,灵活的,可扩展的缓存机制
- 支持缓存数据的备份(持久化),以便应用程序重启时回复缓存数据
- 线程安全
  - 确保内存中的缓存状态与备份(持久化)的状态保持同步

企业库 3.0– Application Blocks

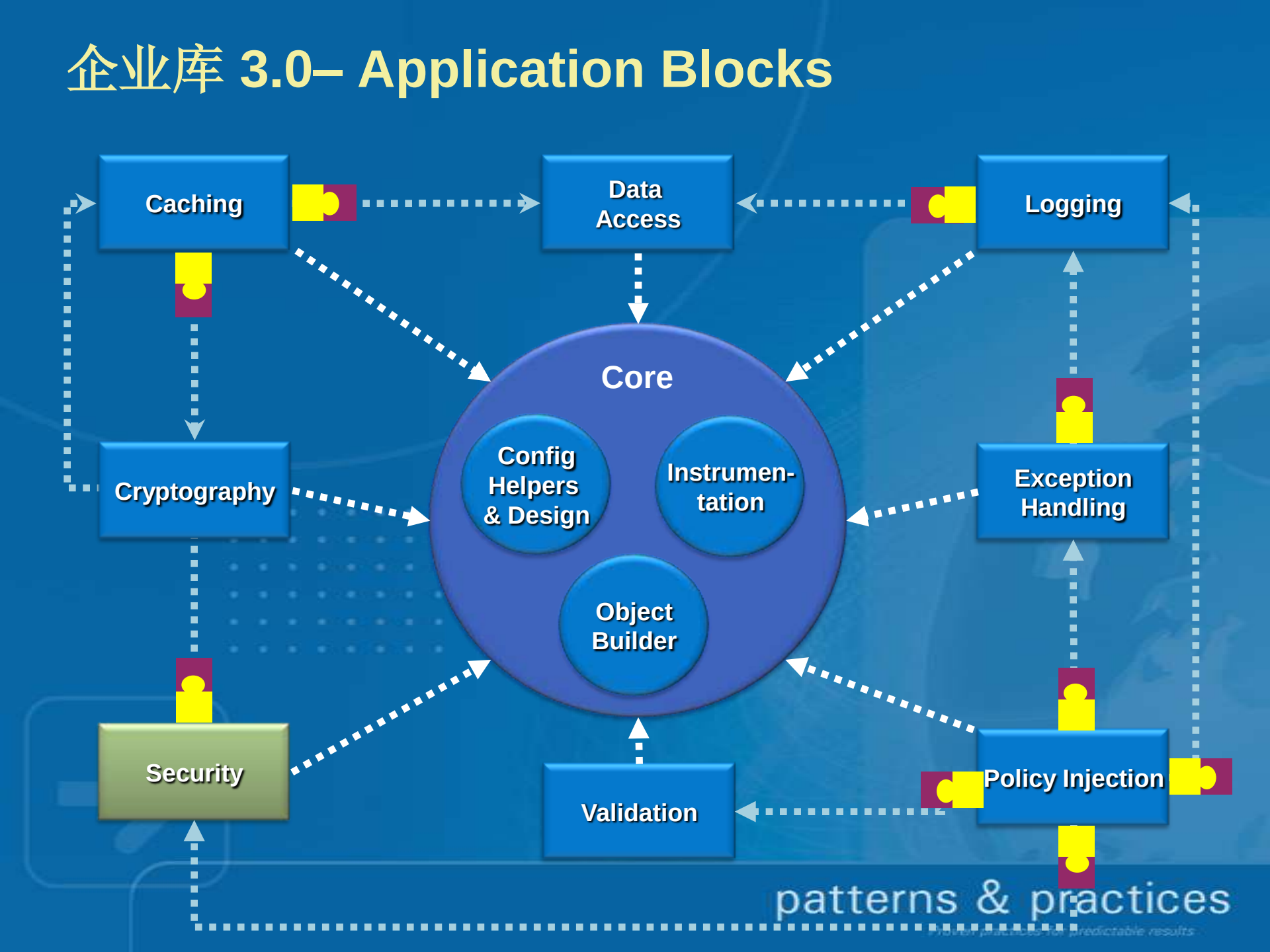


Cryptography Application Block

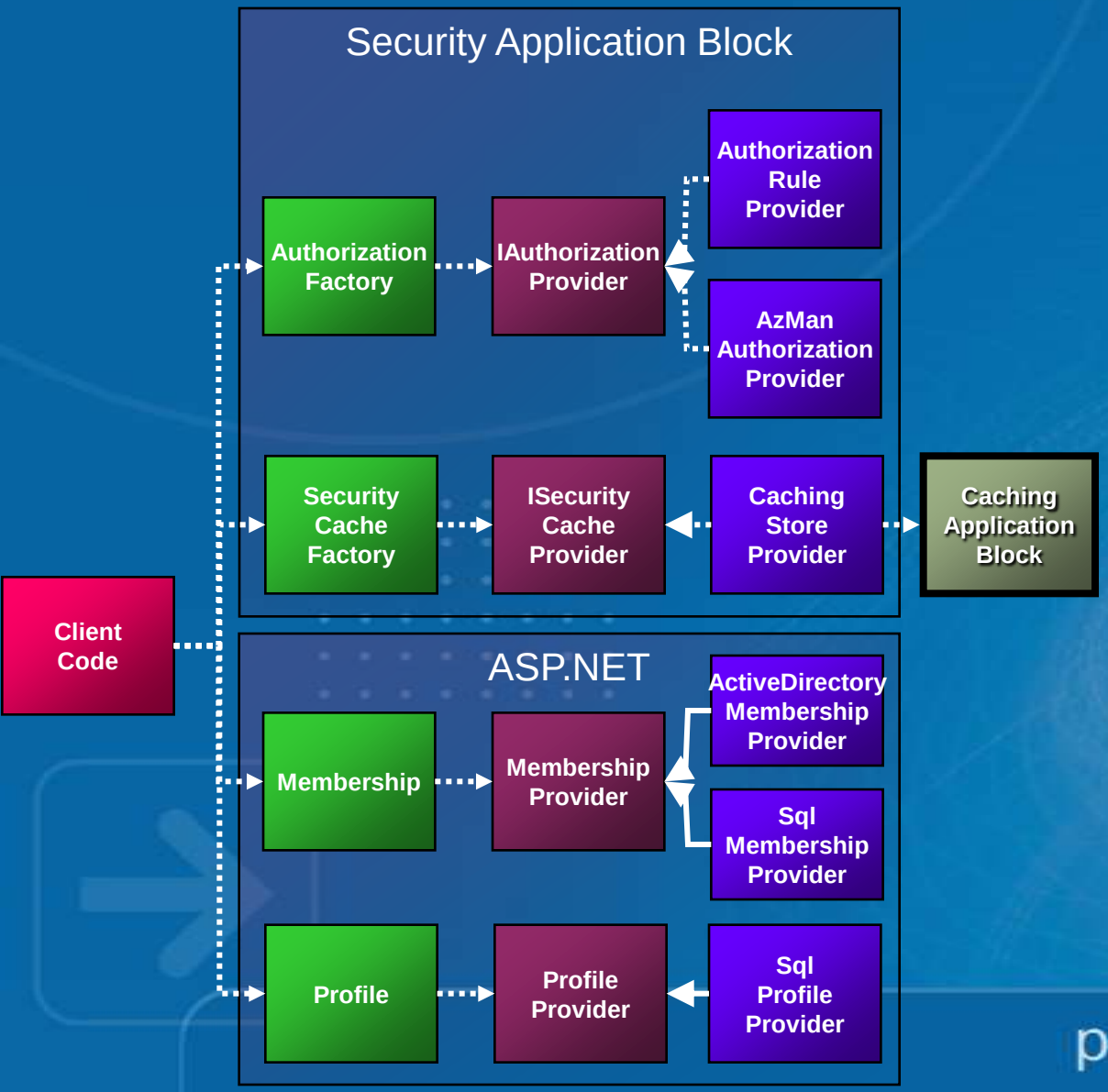
- 改善安全性
  - 提供简单的方式实现公共的加密机制
- 易用性
  - 提供字符串和流的两种加密接口

    CreateHash, CompareHash, EncryptSymmetric, DecryptSymmetric
- 整合性
  - 支持所有 .NET 内置的加密算法,也可以创建自己的算法
  - 支持 DPAPI 加密
  - 加密算法和密钥可以配置

企业库 3.0– Application Blocks

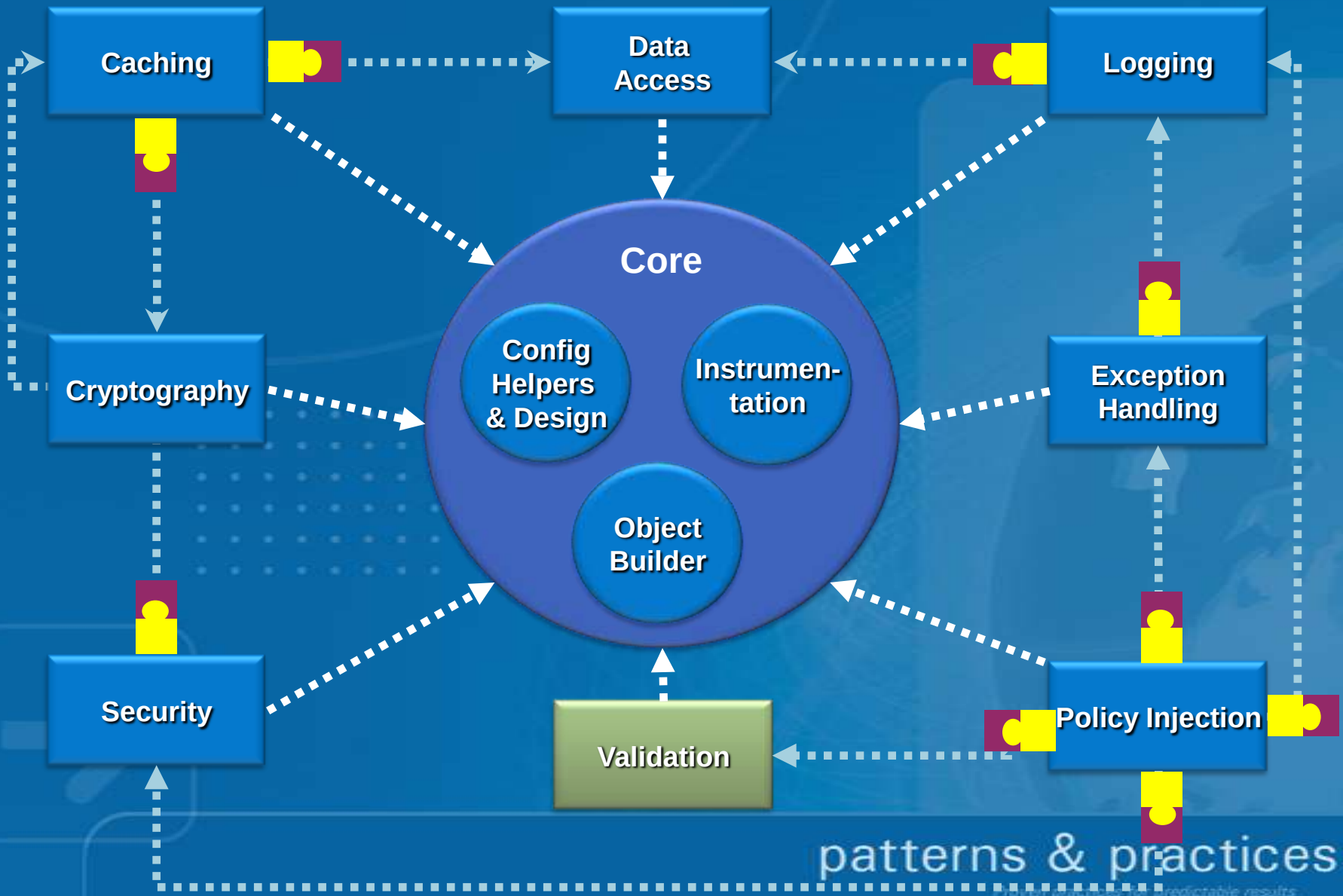


Security Application Block + ASP.NET



- 封装应用程序中的通用安全任务
- 提供一个关于应用程序安全性的标准的提供者模型
- 让用户编写最少的安全代码
- 整合应用程序安全领域的最佳实践

企业库 3.0– Application Blocks



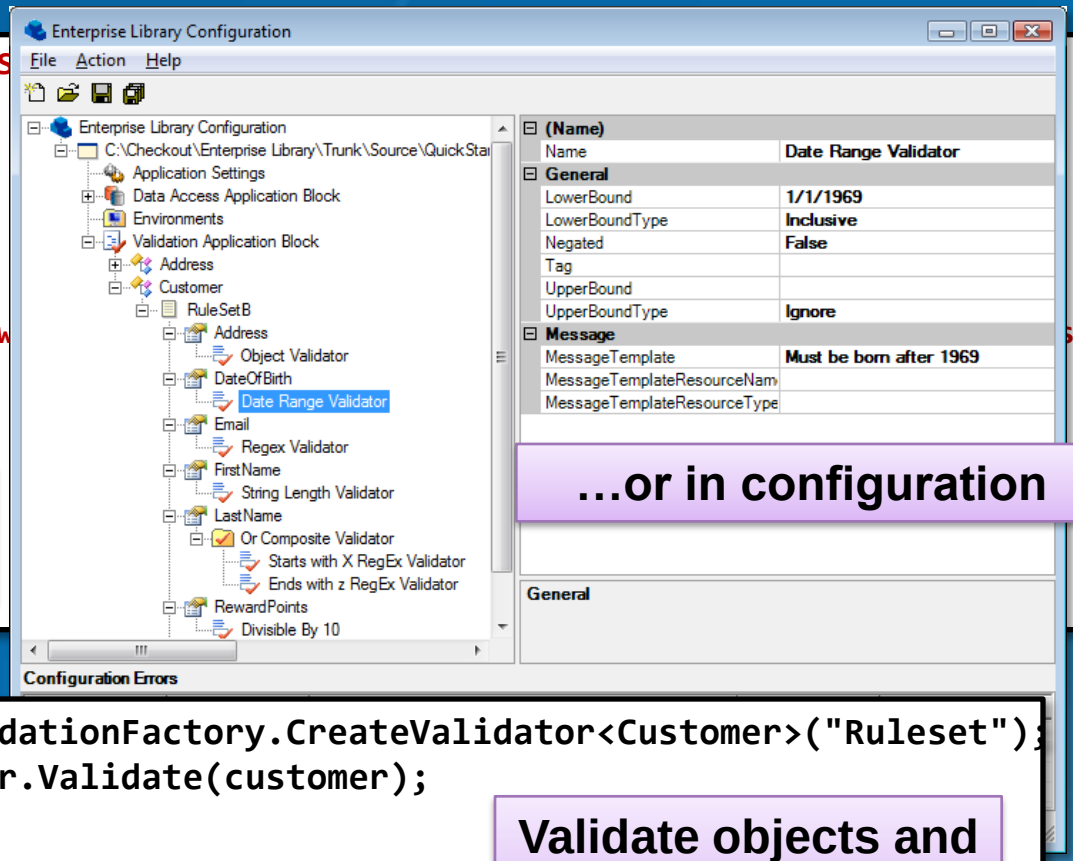
Validation Application Block

- 一次指定你的验证规则
  - 使用配置,Attribute,或者编程的方式指定
- 在应用程序的任何地方方便地验证数据
  - 编程
  - 与 Windows Forms, ASP.NET 或者 WCF 整合
- 复合验证逻辑
  - 内置的基本验证规则
  - 验证指定类型的数据
  - 在单一类型上应用多个验证规则

Validation Example

```
[StringLengthValidator(1, 50, Ruleset="RuleS
public string LastName
{
    get { return lastName; }
    set { lastName = value; }
}

[RegexValidator(@"\w+([-+.']\w+)*@\w+([-.]\w
    Ruleset="RuleSetA")]
public string Email
{
    get {
    set {
}
```

Specify validation rules in attributes…

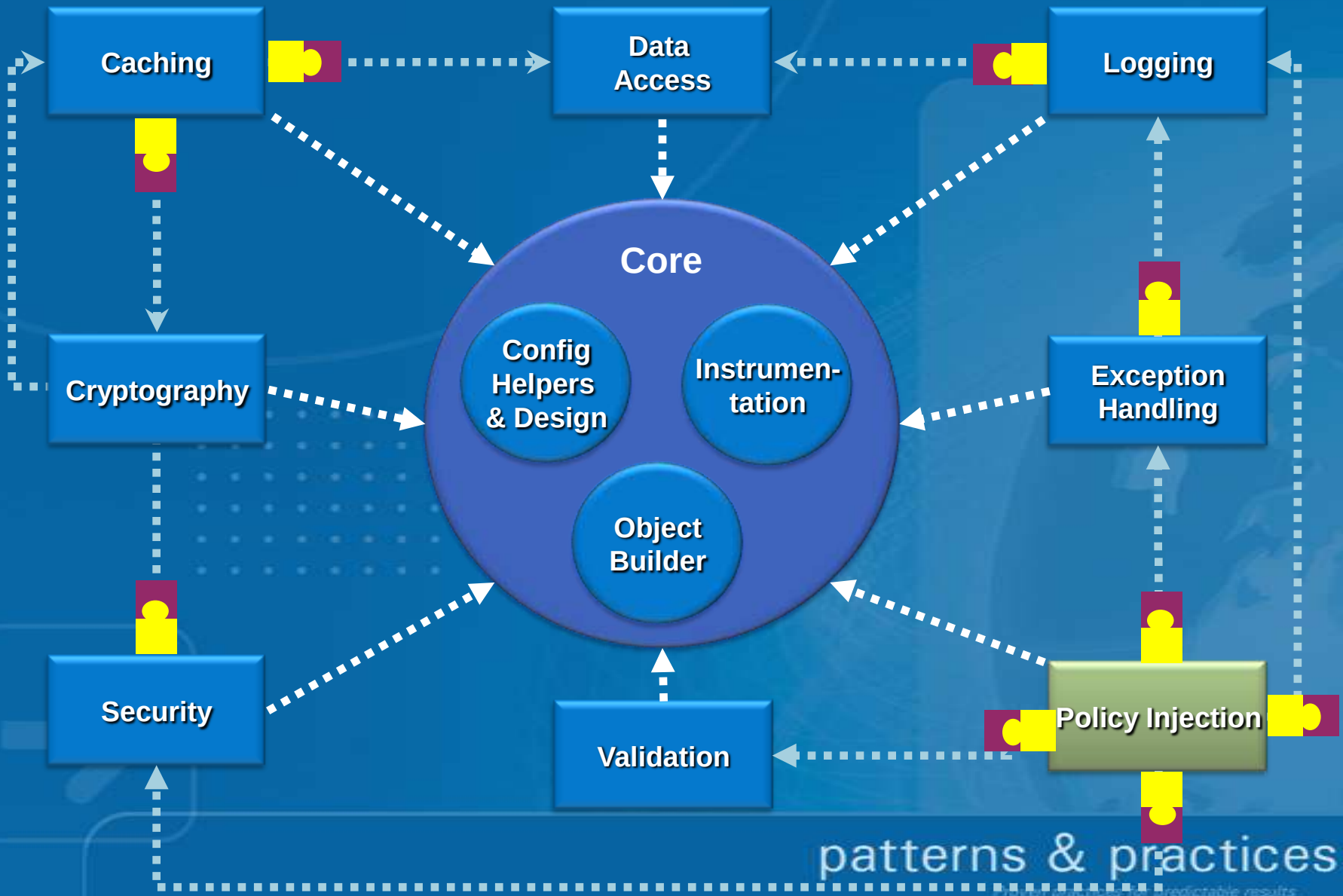


…or in configuration

```
Validator<Customer> validator = ValidationFactory.CreateValidator<Customer>("Ruleset");
ValidationResults results = validator.Validate(customer);
if (!results.IsValid)
{
    foreach (ValidationResult result in results)
    {
        Console.WriteLine("Message={0}, Key={1}, "Tag={2}", result.Message,
            result.Key.ToString(),
            result.Tag == null ? "null" : "\"" + result.Tag.ToString() + "\"");
    }
}
```

Validate objects and process results

企业库 3.0– Application Blocks

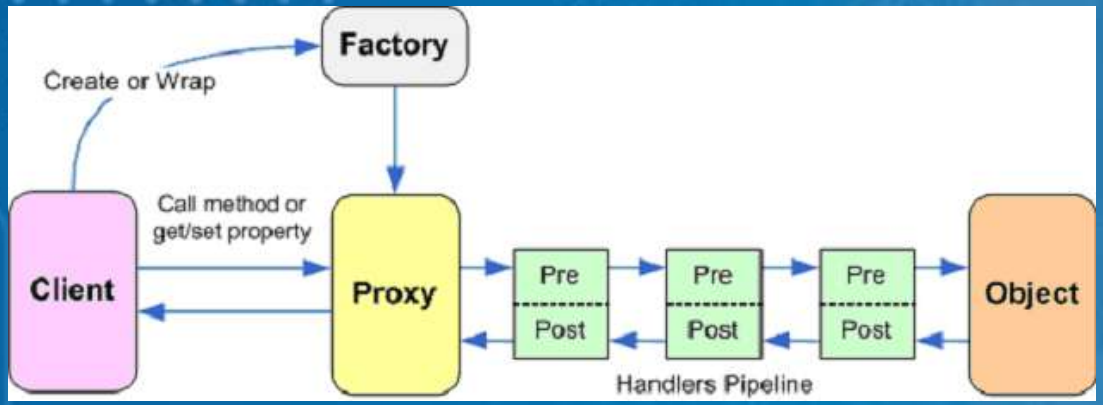


Policy Injection

- 对业务逻辑进行横向切割
  - 在运行时使用拦截和注入为应用程序添加策略
  - 通过声明的方式指定策略被应用到何处
  - 策略允许通过配置或者Attribute定义
- 通过策略注入可以方便地添加任何一个Application Block到应用程序中,而不写一行代码
  - Validation, Logging, Authorization, Exception Handling, Caching, Performance Counters

Policy Injection Application Block

- 策略注入Application Block 提供一个工厂创建或包装一个策略对象
- 如果策略被指定,一个代理将代替真实对象被返回给调用端
- 当调用启用策略的成员时,一个处理管道在真实调用之前和之后被调用
- 每一个处理管道可以读写调用中用到的数据

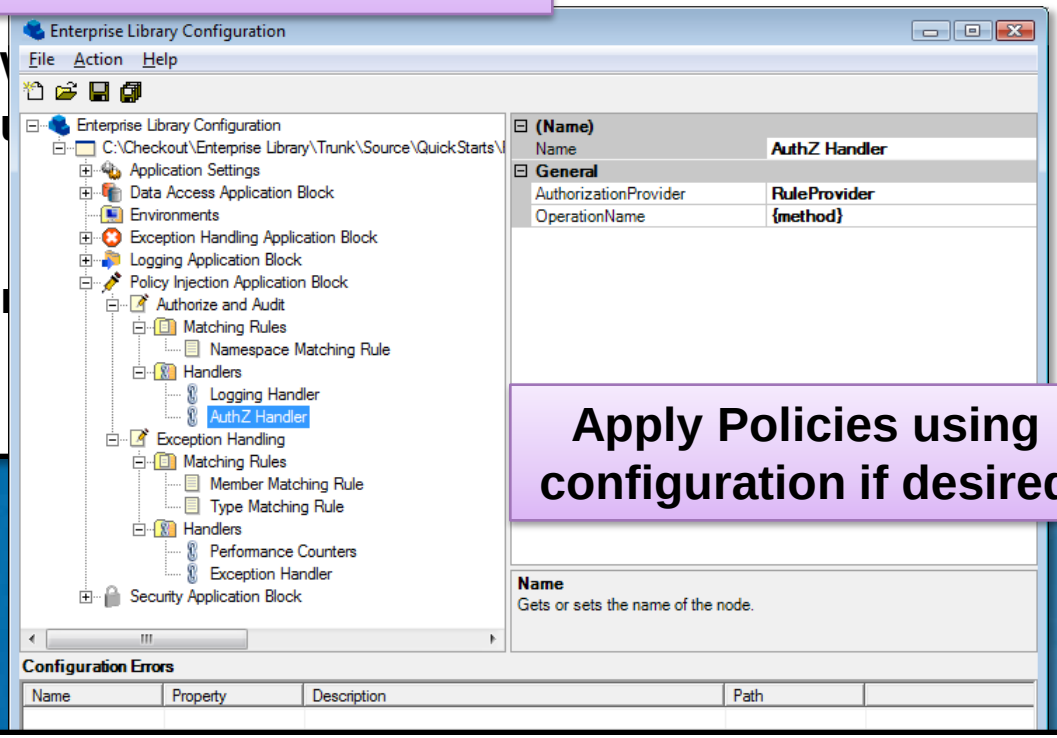


Policy Injection Example

Write classes that extend MBRO or implement an interface

```
public class BankAccount : MarshalByRefObject
{
    // Constructors and fiel

    [ValidationCallHandler]
    public void Deposit([Range
        RangeBoundaryType.Exclu
        decimal depositAmount)
    {
        balance += depositAmou
    }
}
```

Apply Handlers using attributes if desired



Apply Policies using configuration if desired

Create objects using PolicyInjection class

```
BankAccount account = PolicyInjection.Create<BankAccount>(customerId);
account.Deposit(1234.56M);
```

Call your methods the usual way

其它

- Application Block Software Factory
  - 使用户构建自己的Application Block或者扩展现有的Application Block
- Strong-Naming Guidance Package
  - 对程序集进行自动强命名

资源

- 下载企业库及相关资源:
  - http://msdn.microsoft.com/practices
- 企业库社区:
  - http://codeplex.com/entlib
- 企业库开发团队的博客:
  - http://msdn.microsoft.com/practices/Comm/EntLibBlogs/

回顾

- 什么是企业库
- 企业库能做什么
- 企业库的历史
- 企业库主要内容
- 企业库的未来发展
- 相关资源

感谢大家！


Microsoft®
patterns & practices
Proven practices for predictable results