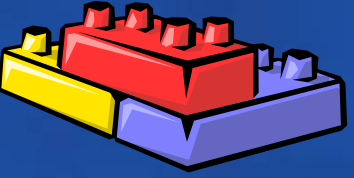


您的潜力，我们的动力

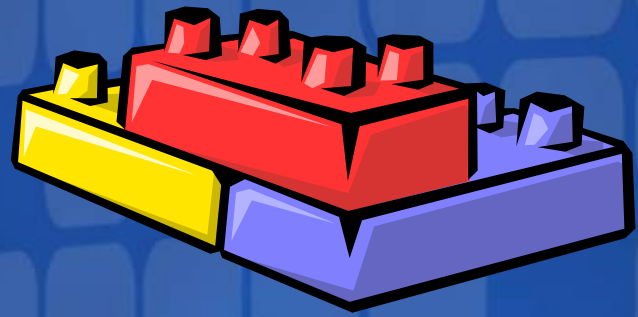
Microsoft
微软（中国）有限公司

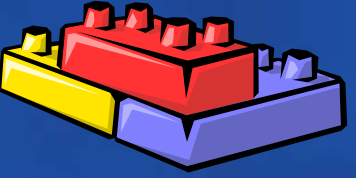
Enterprise Library –

Security Application Block

微软（中国）开发合作部

吴延安
.NET 首席顾问

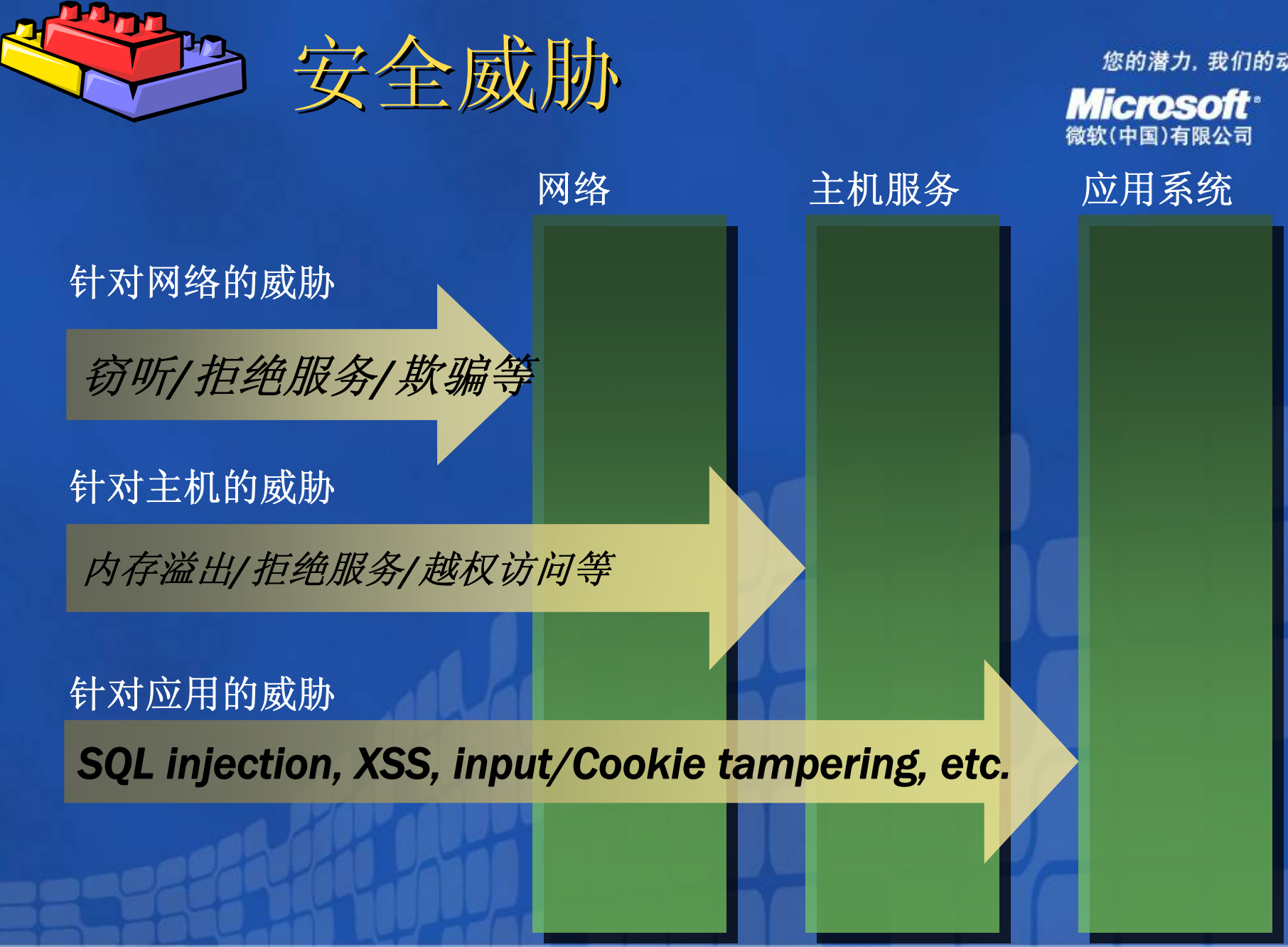




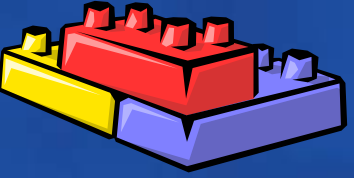
日程

- Security Application Block概述
- 使用Application Block的三部步曲
  - Defining your configuration
  - Creating an instance of the security provider objects
  - Executing the methods
- 深层探秘
  - Selecting the right options for security
- 面向高级开发人员的扩展机制
  - Key extensibility points

安全威胁



应用系统的安全架构



避免将例外产生的敏感信息直接暴露给客户

安全审计

SQL 许可

信任连接

Exception处理

Logging机制

非Web客户端

Web 客户端

表现层

业务逻辑

服务代理

信息确认层

数据访问层

数据保护层

.Net Framework

Database

跨越系统边界的安全问题

认证和授权管理

Framework Crypt Class

认证中心（如CA…）

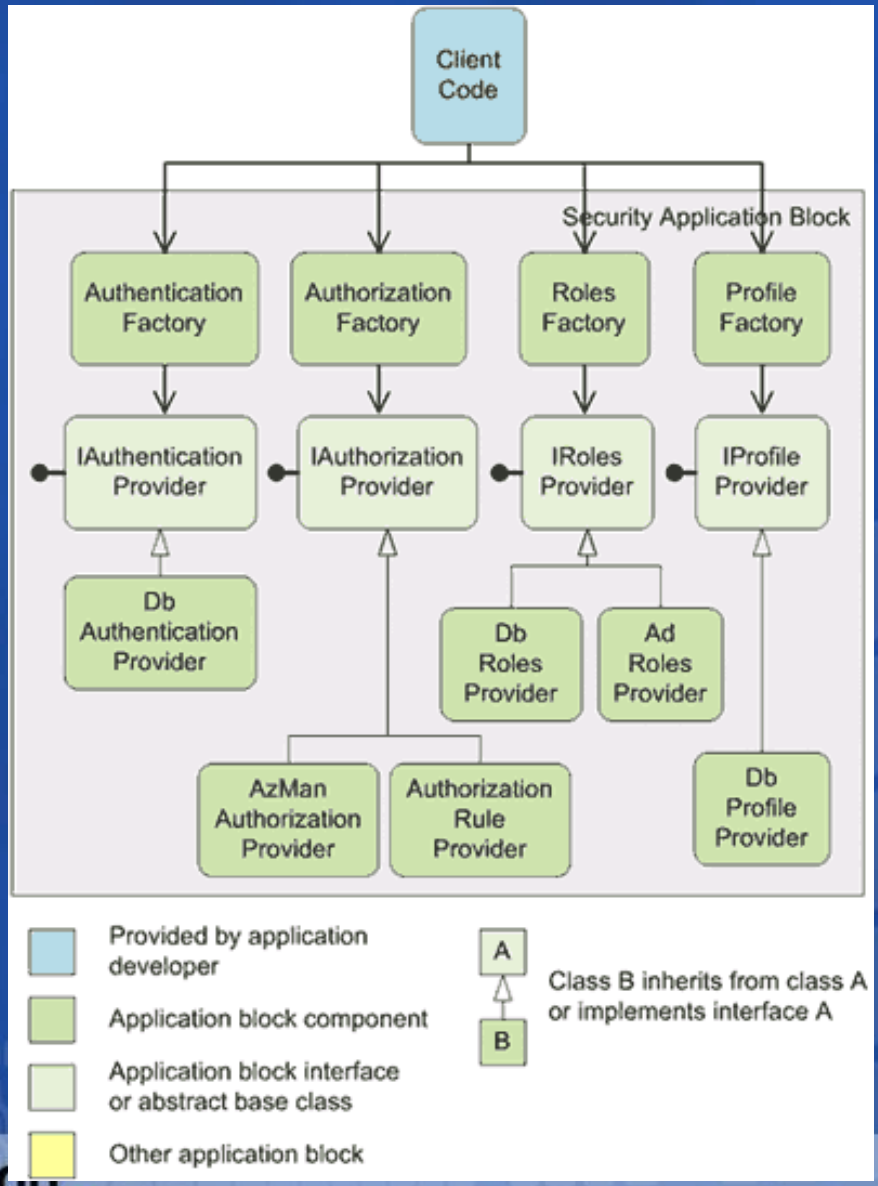
授权中心（如AD/AzMan/…）

敏感信息
Password/
Credit Card/数据库连接字符串

Security Scenarios and Goal

- 通过一个或多个安全机制，帮助开发人员在应用程序中实现通用的安全相关任务
  - 一致性、可展性、配置驱动的安全机制（不需修改代码）
- 需要提供高扩展性，以便在不改变应用程序代码的情况下更该认证或授权方式
- 提供以下功能
  - 认证
    - Authenticating a User using Credentials
    - Authenticating a User Using a Token
    - Obtaining a Temporary Token for an Authenticated User
    - Terminating a User Session (Expire a Token)
  - 授权
    - Determining Whether a User Is Authorized to Perform a Task
  - 角色管理
    - Determining the Roles a User Is In
  - Profile管理
    - Reading and Writing Profile Information about a User
  - 缓存登陆时的相关安全凭证（认证、授权信息）
- 通过图形化 Configuration Console工具，提高了对配置管理的易用性及对配置信息进行校验

Security Application Block

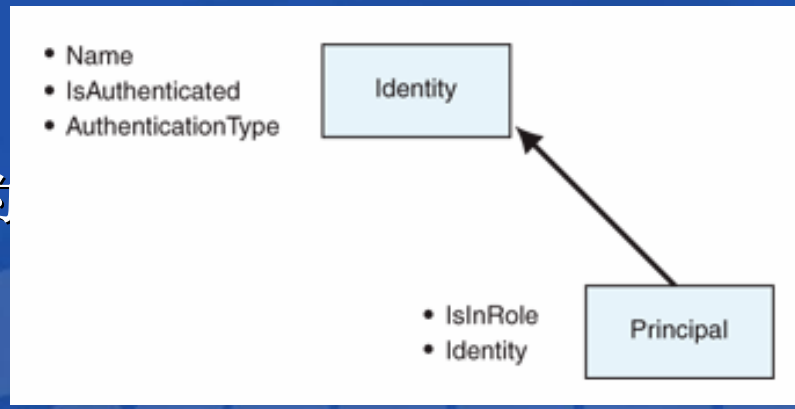


- 封装通用的应用安全相关的任务
- 最小化应用安全相关的编码（通过少量的方法Authenticate, Authorize, GetRoles, SetProfile, GetProfile…）
- 提供标准的安全Provider模型
  - **DbAuthenticationProvider** – authenticate users against database
  - **AzManAuthorizationProvider** – authorize users against Authorization Manager
  - **DbRolesProvider** – retrieve roles from a database
  - **DbProfileProvider** – retrieve profiles from a database
  - **AdRolesProvider** – retrieve roles from Active Directory
- 基于应用安全的最佳实践

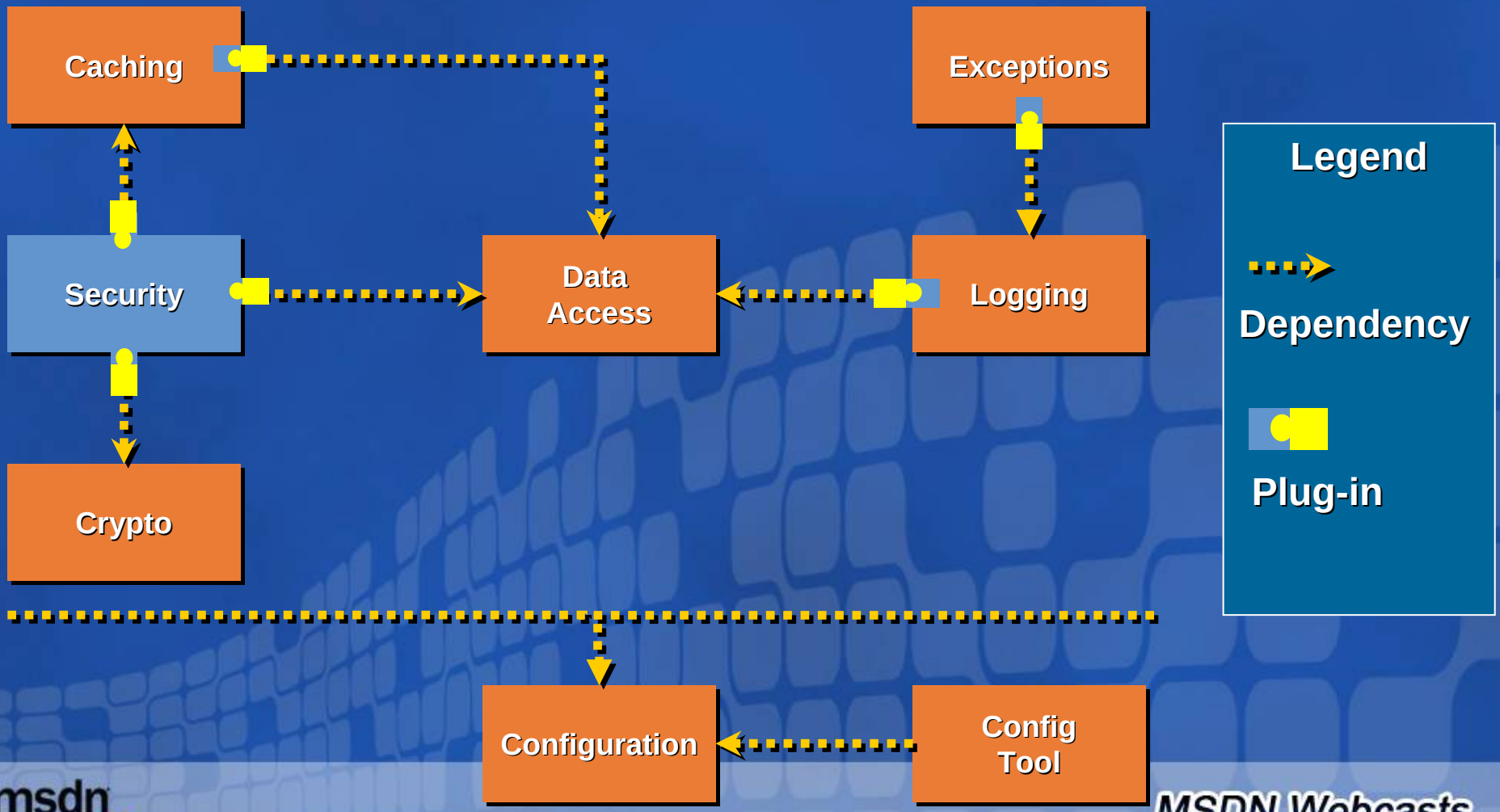
Security Application Block

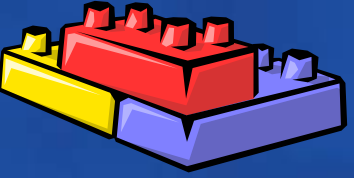
Role-based Security: Principals and Identities

- Security Application Block实现了.Net支持的基于角色的安全特性
  - Credential（令牌）
    - 用户ID、密码或证书等确认身份的
  - Identity（身份）
    - Encapsulates information about user
    – Name and Authentication type
  - Principal（主体特征）
    - Represents the security context the code is running under – Identity and Roles the user belongs to
    - Used in Authorization process
- 通过Identity与Principal的分离实现了认证和授权两方面的功能



Enterprise Library v1



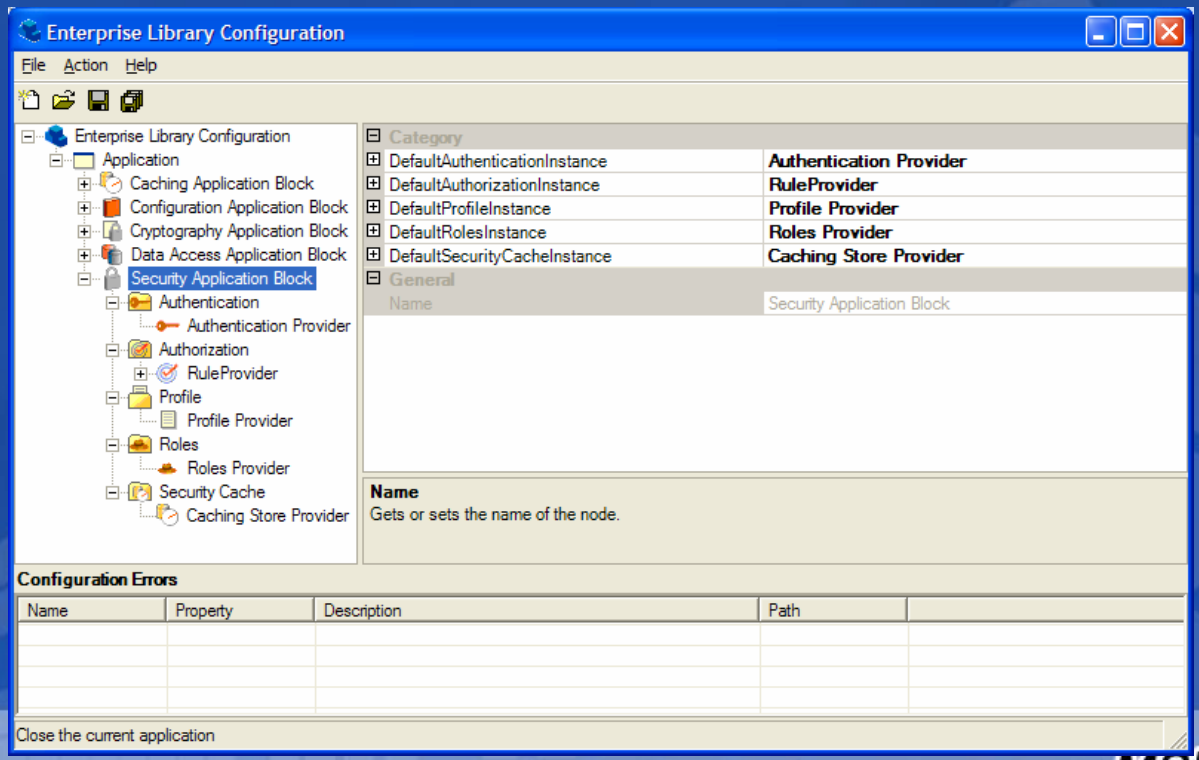


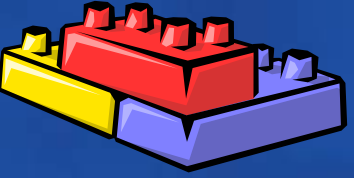
使用Application Block的三步曲

Step 1: 定义配置文件

- 应用系统需要一个 app.config (or web.config)文件
- 使用Enterprise Library配置工具为Security创建配置信息
- 使用VS中的 post-build 将config files拷贝到运行目录

```
copy "$(ProjectDir)\*.config" "$(TargetDir)"
```





Step 2: 创建 Security Provider 实例

- Security Application Block中使用了Martin Fowler提出的”工厂设计模式”创建Security相关的Provider.
  - 可以使用内置的AD, Database, AzMan Provider
  - 也可通过Plugin的方式进行Provider的扩展

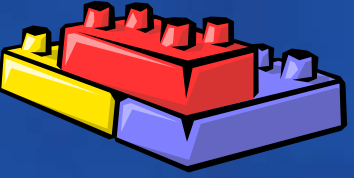
```
' Create the default authentication provider instance
Dim authProvider As IAuthenticationProvider =
AuthenticationFactory.GetAuthenticationProvider()

' Use a named instance to map to configuration
Dim authProvider As IAuthenticationProvider =
AuthenticationFactory.GetAuthenticationProvider("Authe
ntication Provider")
```

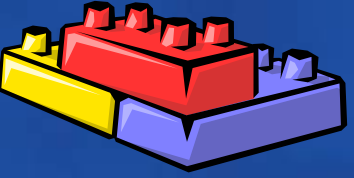
Step 3: 执行Security Provider 命令

- Authentication（认证）
  - Authenticate
  - Cache identity
  - Expire a session
- Authorization（授权）
  - Determine if user is authorized to perform a task
- Roles（角色）
  - Determine what roles a user is in
- Profiles（个性化）
  - Read and write profile information

```
NamePasswordCredential credentials =
                                    new NamePasswordCredential(username, password);
‘通过Name/Password令牌进行认证
bool authenticated = authProvider.Authenticate(credentials, out identity);
```

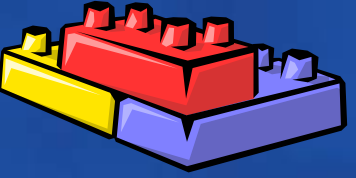


演示：Security使用三步曲



深层探秘...

...this is where it gets interesting



Authentication（认证）

- Authentication是一个确认调用者身份的过程. 使用时需考虑以下方面:
  - 界定认证的使用边界（boundary），尤其当应用系统跨越信任边界时，一个信任边界通常包括 assemblies, processes, and hosts.
  - 确认调用者身份（caller），通常情况下是用户名及其密码。

严格的帐号（Account）管理策略

- 您的应用是否需要强密码?
- 您是否限制登陆尝试的失败次数?
- 您是否对登陆失败的原因给与过多的提示?
- 是否强制推行密码的有效期限?
- 能否根据特殊事件快速对Account进行无效处理?
- 是否对Login进行日志处理?

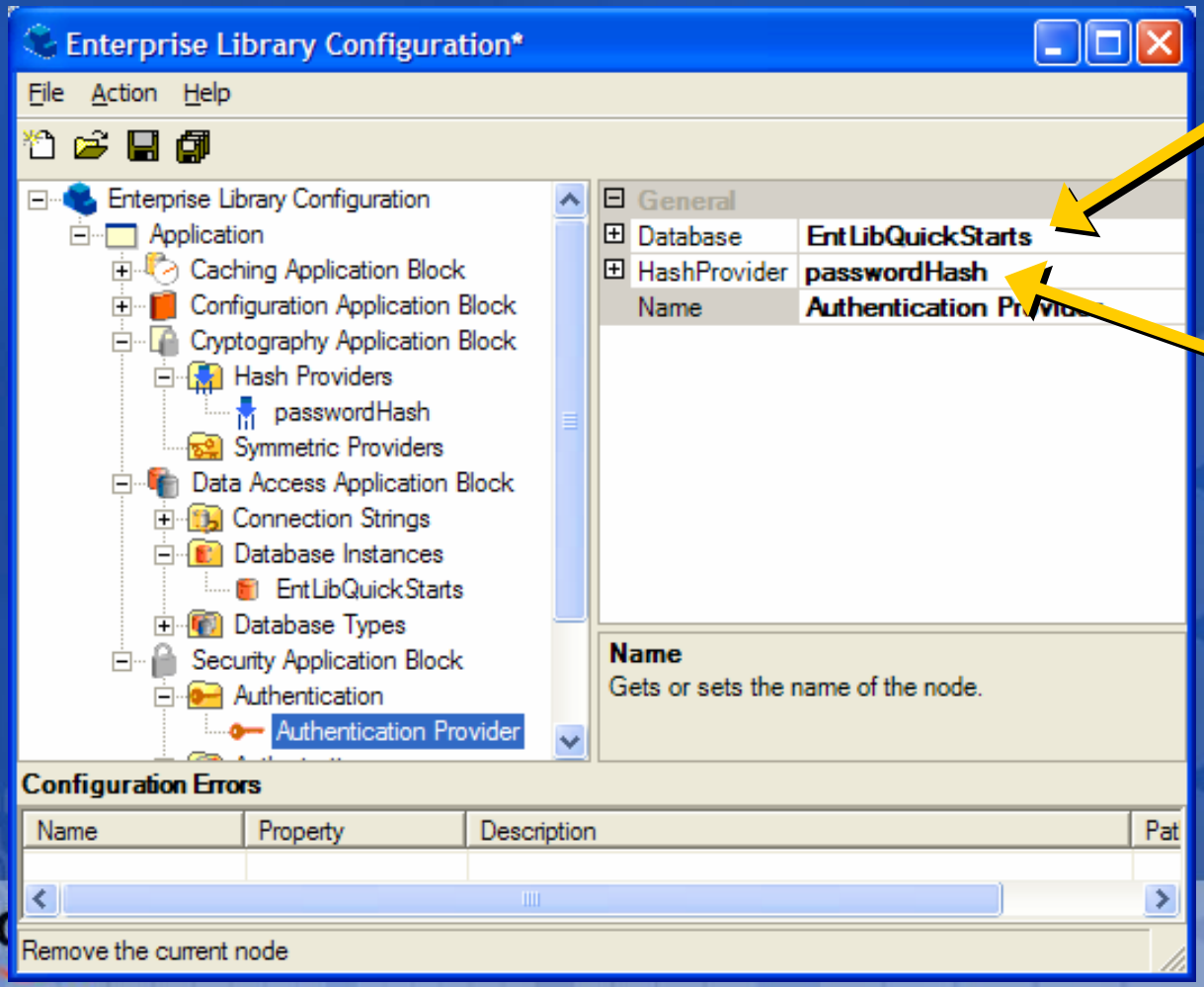
如何改善 Web应用的安全性请参阅以下资料：
Threats and Countermeasures
Chapter 4 – Design Guidelines for Secure Web Applications
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/ThreatCounter.asp

Authentication Database Provider

- 使用数据库对用户帐号及密码进行管理达到认证的方式
- 所依赖的Application Block
  - 该Provider中使用Data Access Application Block作为数据访问的实现
    - SQL script included for required schema
  - 使用Cryptography Application Block用于密码的散列算法（ Hash ）
  - 仅当适用内置的基于数据库的Profiler、roles provider时才有以上的依赖关系

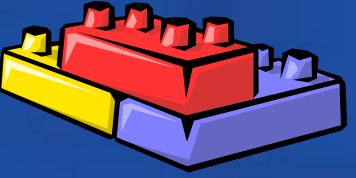
配置 Authentication Database Provider

- 使用 Configuration Console



配置所适用的数据库

配置 **hash provider**



对一个User进行认证处理

- 创建authentication provider

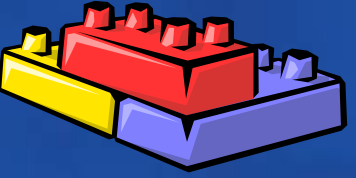
```
IAuthenticationProvider authenticationProvider =
    AuthenticationFactory.GetAuthenticationProvider(“My Provider");
```

- 创建NamePasswordCredentials组合的令牌

```
NamePasswordCredential credentials = new
    NamePasswordCredential(“JohnS”, “MyPassword”);
```

- 调用Authenticate

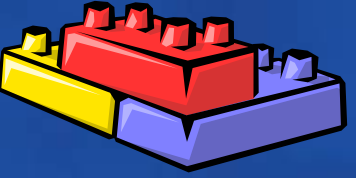
```
IIdentity identity;
bool authenticated =
    authenticationProvider.Authenticate(credentials, out identity);
```



Authorization（授权）

- 授权的作用在于决定一个被认证的用户是否具有某种业务操作的权限。
- 不适当的或弱授权可能导致信息泄露或篡改的风险
- 通过授权进行深度防范是安全应用的一个重要策略

- 关于授权的思考
  - 是否需要深度防范策略?
  - 使用何种防范机制?
  - 是否考虑使用基于角色的解决方案?

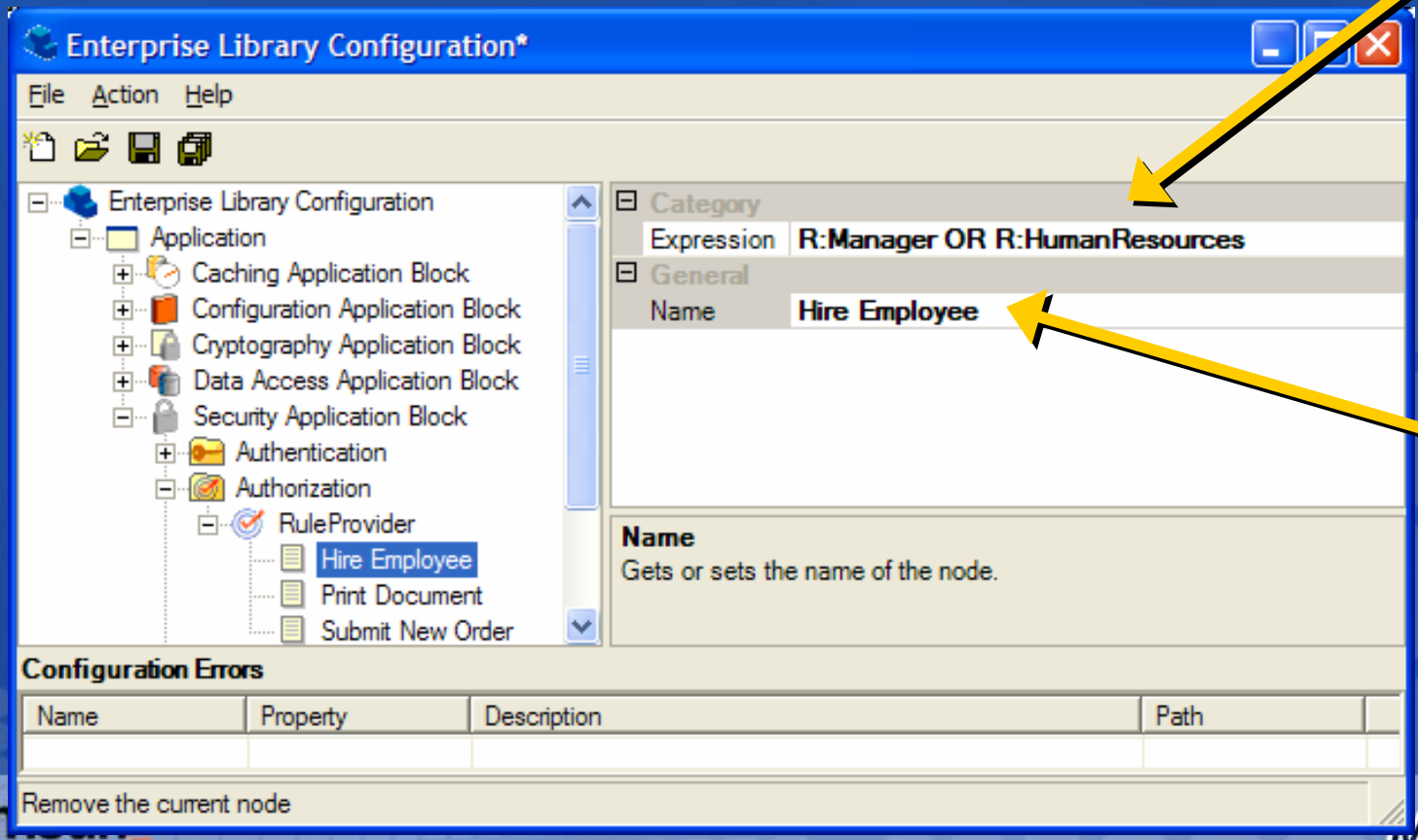


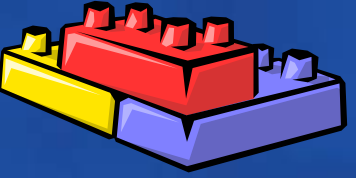
Authorization Providers

- AzMan (Authorization Manager)
  - Windows自带的授权管理机制
  - Microsoft® Windows® 2000™ Server with Service Pack 4, Microsoft Windows Server 2003, or Windows XP SP1 with Windows Server 2003 Administration Tools Pack
- 授权规则（Authorization Rule）
  - 允许创建与业务操作关联的规则用于运行时的评估
  - Configuration Console提供了”规则表达式”编辑器
  - 定义的规则被保存于配置文件中
  - 定义的规则也可通过扩展的方式保存于数据库中

配置 Authorization Rule Provider

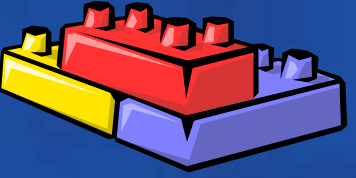
- 使用 Configuration Console





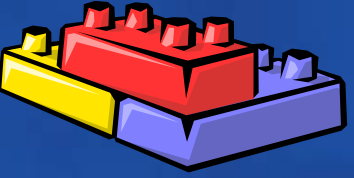
授权规则的诸要素

- I：Identities（身份）
  - Specific (for example, “Bob”)
  - Anonymous (?)
  - Any (*)
- R：Roles（角色）
  - Specific (for example, “Managers”)
  - Any (*)
- Operators（关系运算操作）
  - AND, OR, NOT and ()

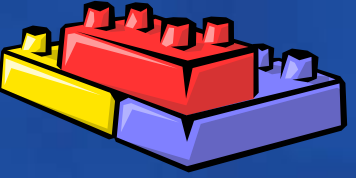


授权规则表达式

- **I:Bob**
  - Only a user with the identity Bob is authorized
- ((R:HumanResources OR R:GeneralManagers) AND (NOT R:HRSpecialist))
  - Only users that are either in the HumanResources or GeneralManagers roles and not in the HRSpecialist role are authorized



演示：Rule Provider配置及Rule Expression Editor



对用户进行授权的过程

- 创建authorization provider

```
IAuthorizationProvider ruleProvider=
    AuthorizationFactory.GetAuthorizationProvider
    ("RuleProvider")
```

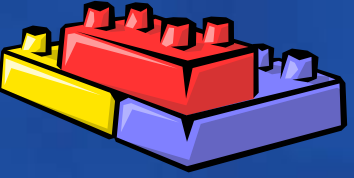
- 调用Authorize
  - Accepts an IPrinciple and a rule name

```
bool authorized =
    this.ruleProvider.Authorize(principal, “Hire
    Employee”);
```

授权策略–角色（Roles）

- 两类基本的授权策略:
  - 基于角色（**Role based**）.

    Access to operations (typically methods) is secured based on the role membership of the caller. Roles are used to partition your application's user base into sets of users who share the same security privileges within the application.
  - 基于资源（**Resource based**）. Individual resources are secured using Windows ACLs. The application impersonates the caller prior to accessing resources, which allows the operating system to perform standard access checks.

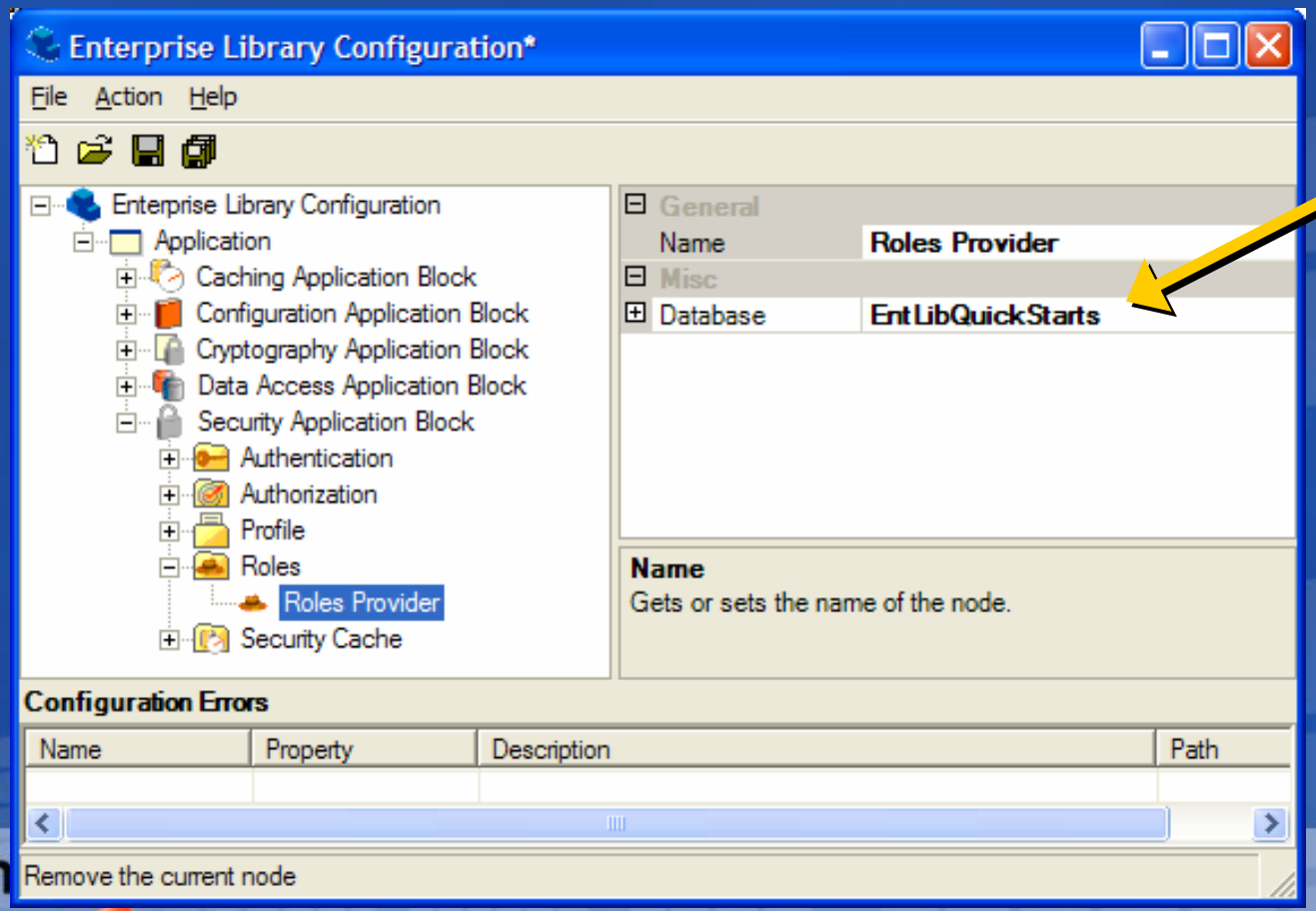


Roles Providers

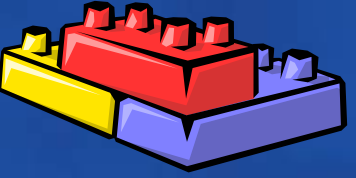
- Active Directory®
- Database
  - Uses Data Access Application Block for database access
    - SQL script included for required schema
  - Out-of-box requires using supplied authentication database provider (roles are tied to a userid in the Users table)

配置 Roles Database Provider

- 使用配置工具指定Roles数据库



Configured database instance



判断某个用户是否属于某个角色

- 创建roles provider

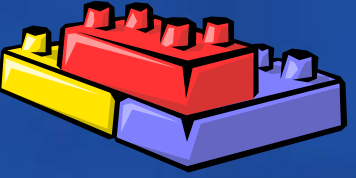
```
IRolesProvider rolesProvider =
    RolesFactory.GetRolesProvider("Roles
    Provider");
```

- 获取用户的主体特征（Principal）

```
IPrincipal principal =
    rolesProvider.GetRoles(this.identity);
```

- Call IsInRoles

```
bool isManager = principal.IsInRole(“Manager”);
```

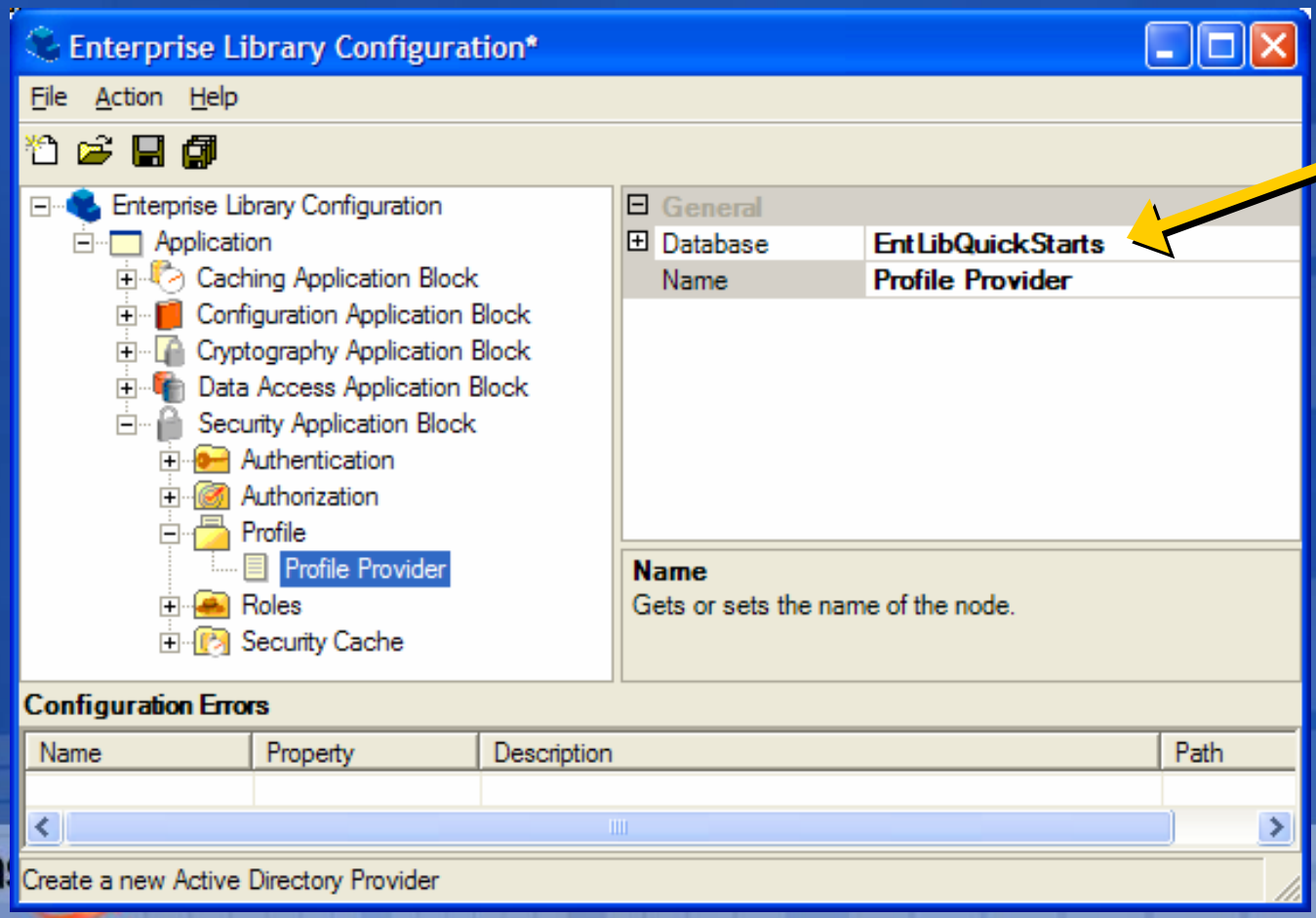


个性化服务（Profiles）

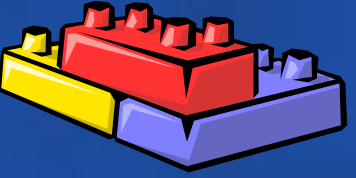
- Profile是系统面向用户提供的灵活性的个体信息的容器，一个用户的Profile可以使以下一种或多种的集合:
  - 简单的字符串或其他基础类型
  - 一个序列化的实体
  - 基础类型及序列化实体的Dictionary. (all values are stored as strings in the database).

配置 Profile Database Provider

- Using the Configuration Console



Configured database instance



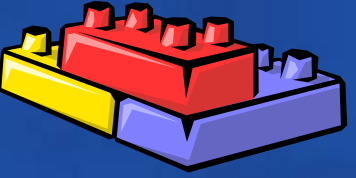
如何写出Profile信息

- 创建profile provider

```
IProfileProvider profileProvider =
    ProfileFactory.GetProfileProvider();
```

- Call SetProfile
  - Pass IIDentity of existing user
  - Pass object with profile information (e.g., serializable class)

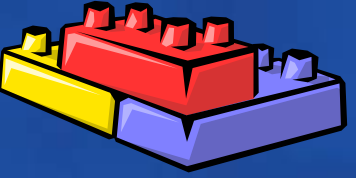
```
profileProvider.SetProfile(identity, profile);
```



获取 Profile 信息

- Call GetProfile
  - Pass IIDentity of existing user
  - Returns object with profile information

```
ProfileInformation userProfile =
    profileProvider.GetProfile(identity) as
    ProfileInformation;
```

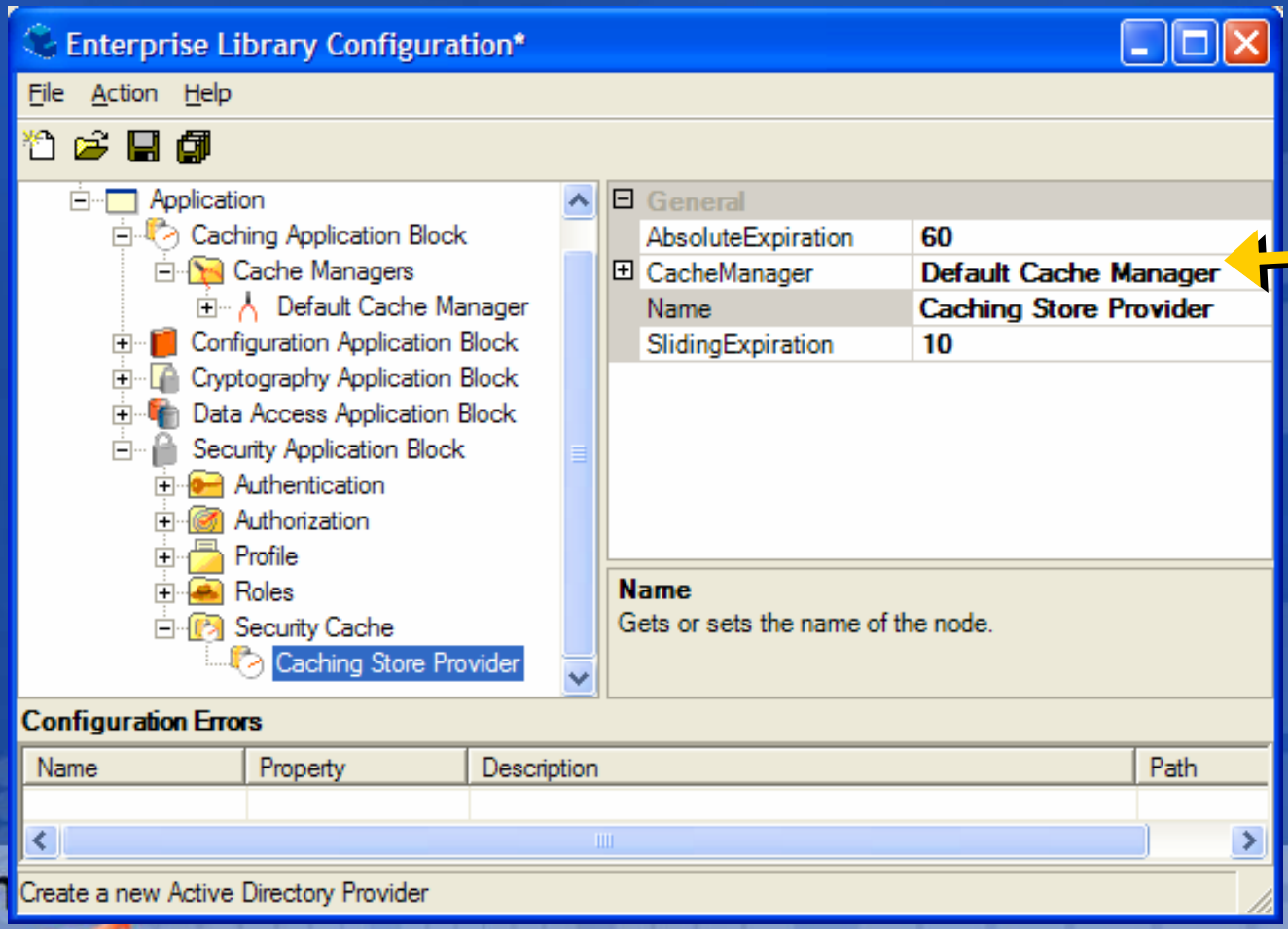


The Security Cache

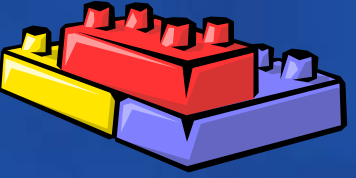
- Security application block允许对security相关的信息进行缓存
- 被缓存的信息可以通过token进行访问
- 典型的使用场景：通过缓存而不是每次都进行认证的方式提高系统的效率

配置 Security Cache

- Using the Configuration Console



Configured cache manager



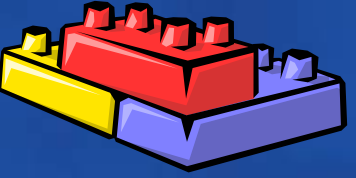
获取临时 Token

- Create security cache

```
ISecurityCacheProvider cache =
    SecurityCacheFactory.GetSecurityCacheProvid
    er(“MyCacheProvider");
```

- Call SaveIdentity
  - Pass IIDentity of existing user
  - Returns a token

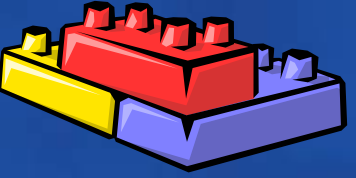
```
IToken token = cache.SaveIdentity(this.identity)
```



使用Token进行认证

- Call GetIdentity
  - Pass token returned by SaveIdentity
  - Returns IIDentity or null

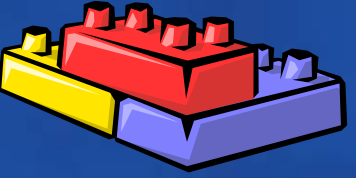
```
IIdentity savedIdentity = cache.GetIdentity(token);
if (savedIdentity != null)
{
   // user is authenticated
}
else
{
  // user not authenticated
}
```



强制让一个 Token 失效

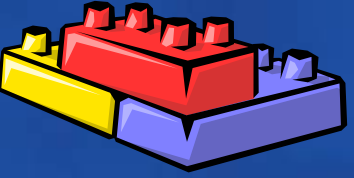
- Call ExpireIdentity
  - Pass token returned by SaveIdentity

```
cache.ExpireIdentity(token);
```



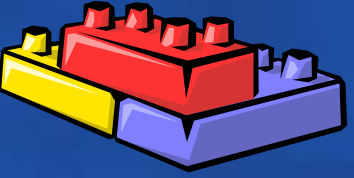
关键扩展点

- 可客户化的security providers
- Enhancing/expanding database providers
- Plus…
  - Anything and everything – you have the source code!
  - Please post extensions and suggestions to the community
    - http://workspaces.gotdotnet.com/entlib



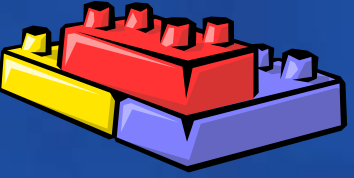
演示:

通过Rule Provider扩展使授权规则保存于Database中



Session Summary

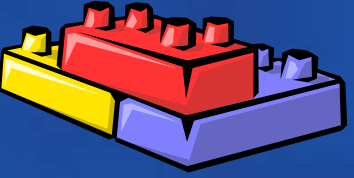
- Security概要、功能与特点
- 使用色curity application block三步曲
  - Defining your configuration
  - Creating an instance of the security provider objects
  - Executing the methods
- 深层探秘
  - Selecting the right options for security
- 对高级开发人员的扩展点
  - Key extensibility points



Microsoft
patterns & practices
proven practices for predictable results

Announcing: Enterprise Library 1.0

http://www.microsoft.com/practices
http://www.microsoft.com/downloads/details.aspx?familyid=0325b97a-9534-4349-8038-d56b38ec394c&displaylang=en (EntLib download)
http://workspaces.gotdotnet.com/entlib



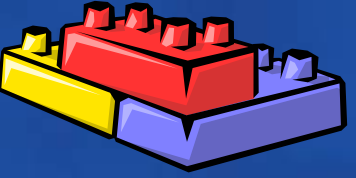
资源

- Improving Web Application Security

  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/ThreatCounter.asp
- Improving .NET Application Performance and Scalability

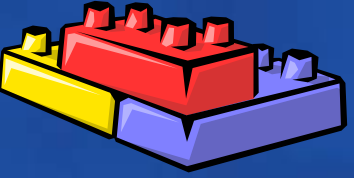
  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpag/html/scalenet.asp
- Application Architecture for .NET

  http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/distapp.asp
- PatternShare.org
- Enterprise Library Community
  http://go.microsoft.com/fwlink/?linkid=39209&clcid=0x09
- www.ronjacobs.com
  - Slides
  - Tech Tips
  - Podcasts



Webcasts

- Webcasts China
- 2005/5/20 – patterns & practices Live:企业库系列讲座(1): Entlib概述
- 2005/5/27 – patterns & practices Live:企业库系列讲座(2): 配置管理应用程序块
- 2005/6/3 – patterns & practices Live:企业库系列讲座(3)：数据访问应用程序块
- 2005/6/10 – patterns & practices Live:企业库系列讲座(4)：安全应用程序库
- 2005/6/17 – patterns & practices Live:企业库系列讲座(5)：日志和仪表盘管理应用程序块
- 2005/6/24 – patterns & practices Live:企业库系列讲座(6)：缓存应用程序块



您的潜力，我们的动力

Microsoft®
微软（中国）有限公司

Microsoft®

Your potential. Our passion.™

msdn

MSDN Webcasts

Q&A

如需提出问题，请单击“提问”按钮并在随后显示的浮动面板中输入问题内容。一旦完成问题输入后，请单击“提问”按钮。