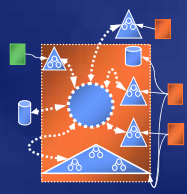
# DEV375
# 基于身份和访问平台的应用开发



蔺华
ISV开发合作经理
平台及开发技术部
微软（中国）有限公司

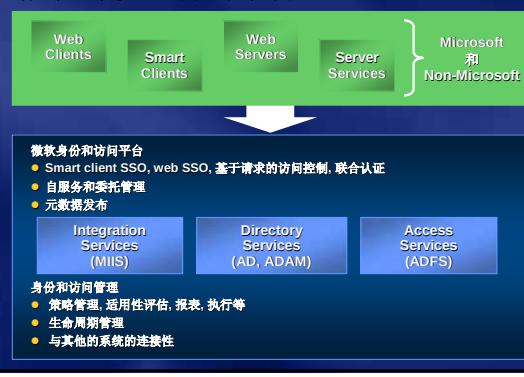
Microsoft Tech·Ed 2005中国

创新 · 远见 · 分享 · 协作

## 什么是身份？

- 数字身份: 某一个人、组、设备或者服务的唯一标识和说明性属性。
  - AD中用户或者电脑的帐号
  - 数据库表中的用户入口
  - 应用中用户登录凭证
  - 需要认证和授权的应用都是基于身份的

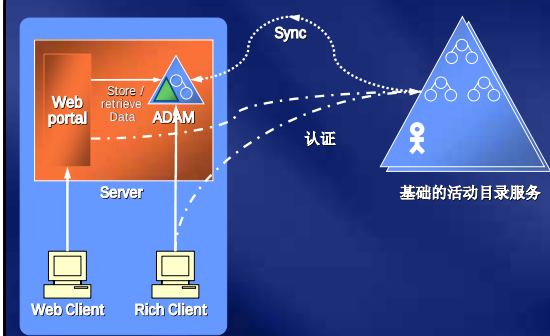
## 微软身份和访问平台




## 我们今天要解决什么问题？

- 对于那些把AD作为基础目录服务的机构
- 对于那些要跟踪自己的资产的机构
  - 或者，有相应需求的部门等
- 充分利用已有的架构
- 也可以开发其他的一些解决方案:
  - 自服务应用来管理自己的资产
  - 针对管理员的全服务管理功能

## 推荐的解决方案



demo

## 资产跟踪管理系统

我们将学习到:

- User experience with the applications
- How various app requirements are met by this solution
- Visualize the end result – you can now move towards this solution

## 应用开发的需求

- 访问管理
  - 认证(Authentication)
  - 授权(Authorization)
- 存储下面的信息：
  - 用户凭证和其他信息数据
  - 应用程序数据和配置数据

## 我们认证的需求

- 认证:
  确认某人是不是他说的那个人的方法
- 应用程序的需求:
  - 增加帐户但是不增加麻烦
  - 允许非常灵活和可扩展的增加新的用户

## 认证的一些选项

- 单点登陆
- 基本/摘要
  - HTTP
  - 基本的发送明文的密码
- 基于表单
  - 在应用表单中输入用户和密码
  - Cookie
- LDAP (Light Directory Access Protocol 轻量级目录访问协议)

## 集成认证

- 强有力的认证支持
- 单点登陆(SSO):
  - 当客户端连接到域时候，操作系统内建的功能就支持SSO
  - 非微软客户端也可以通过合作伙伴的产品来实现SSO
  - 在域中使用很好的扩展了集成认证的系统功能
  - 能通过ADFS来实现跨机构的集合认证
- 富客户端和Web应用都能充分利用集成认证
- IIS 集成整合
  - 非常简单

强烈推荐！

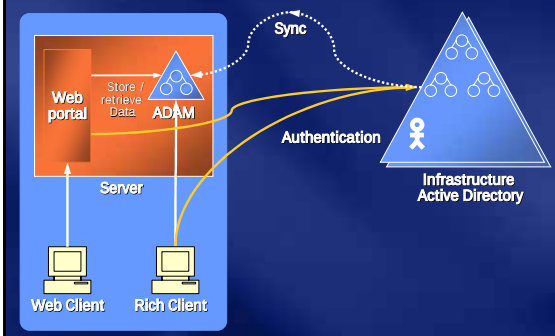
## 怎么用在我们的解决方案中？

- Windows集成认证
  - Web 应用: 不需要任何代码
  - 富客户端:

```
API: AcceptSecurityContext()
```

- 满足需求
  - 充分利用AD来做认证
  - 单点登陆（SSO）
  - 可扩展性: 很容易添加新的用户

## 认证的解决方案




## 应用开发的需求

- 访问管理
  - 认证(Authentication) ✓
  - 授权(Authorization)
- 存储下面的信息：
  - 用户凭证和其他信息数据
  - 应用程序数据和配置数据

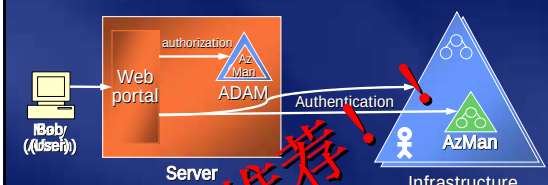
## 我们授权的需求

- 授权:
  基于身份来允许或者拒绝执行某个任务
- 应用程序的需求:
  - 不用代码在应用中授权
  - 管理需要能够进行许可/拒绝访问操作
  - 不同的应用需要使用相同模式(scheme)

## 授权的选项

- 授权管理器 (AzMan)
- ADFS
- Windows ACL(访问控制列表)模式
  - 最方便的授权访问模式
- LDAP 授权
  - 授权信息以数据的形式存储
  - 应用服务器都以合适的方式访问这些信息
- COM+ 和 ASP.NET 角色

## 授权管理器




- 基于角色的授权API
  - 管理角色成员和ACL对象
  - 简化授权和审计过程
  - 基于角色的细化很好的捕捉业务动态
- 应用程序调用AzMan里定义好的角色来进行授权
- 在设计时建立角色策略

强烈推荐！

## 授权管理器 API

- 内嵌于Windows Server 2003, Windows 2000 可以下载安装该组件
- 在AD/ADAM中针对角色和策略的可伸缩的，应用程序特定的存储

```
策略定义脚本:
Set App = AzManStore.CreateApplication("AssetTracker")
App.CreateOperation("ViewRpt")

Set Task=App.CreateTask("View Report")
Task1.AddOperation CStr("ViewRpt")
```

## 怎么用在我们的解决方案中？

- 授权管理器
- 极易使用的 API:

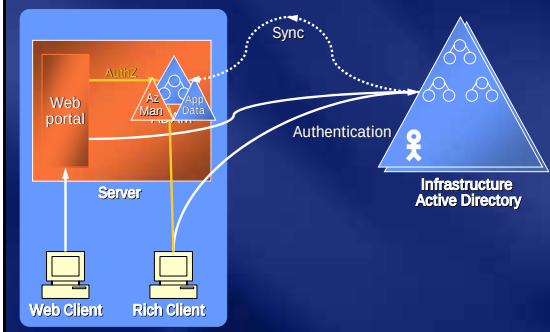
```
'------ at application boot --
AzPol.Initialize 0,"msldap://Server:port/CN=MyStore,DC=...
App = AzStore.OpenApplication("AssetTracker")

'------ at client Connect --
Context = App.InitializeClientContextFromName

'------ on request --
Context.AccessCheck("ViewRpt",Scope,Operations,Names,Values)
Context.GetRoles ()
```

- 满足需求
  - 一致的角色映射, 跨应用重用
  - 管理员在AzMan中完成角色指配，而不是在代码中完成

## 授权的解决方案




## demo

### 访问管理

我们将学习到:

- Integrated authentication
- Authorization : AzMan roles, tasks
- New user has no access, admin adds to group, automated access; disabled user, access automatically denied

## 应用开发的需求

- 访问管理
  - 认证(Authentication) ✓
  - 授权(Authorization) ✓
- 存储下面的信息：
  - 用户凭证和其他信息数据
  - 应用程序数据和配置数据

## 数据存储的需求

- 数据存储:
  - 用户凭证和其他信息数据
  - 应用程序数据和配置数据
- 应用程序的需求
  - 充分利用已有的基础架构和数据
  - 可伸缩，稳定的存储
  - 无缝的安装和配置
  - 易于管理

## 数据存储的选项

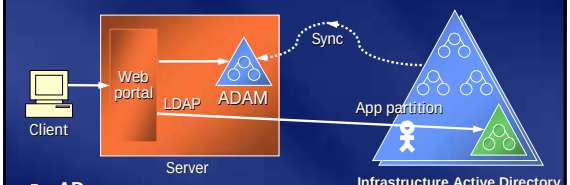
- 数据库
- 目录
- 关键因素
  - 编程和数据模型
  - 开发的经验
  - 管理成本及其经验
  - 易于部署
- 参看更多信息:
  http://www.microsoft.com/adam

## 对于用户来说

- 用活动目录(AD)来存储身份数据
- 组织范围内的用户信息都存放在AD
- 支持基于标准的认证支持(Kerberos, SSL, Digest)
- 在添加另外的身份系统时不会带来身份危机

强烈推荐！

## 应用程序数据的存储




- AD
  - 针对整个组织范围的应用程序数据
  - 针对对安全性要求很高的应用
- ADAM
  - 针对应用程序数据 – 相当独立
  - 应用特定的用户数据或者配置信息
  - 利用Windows身份机制

## 目录访问 API

- 托管代码:
  - System.DirectoryServices：简单易用的高度抽象的对象模型
  - System.DirectoryServices.Protocols：针对高性能的目录应用提供了全面的LDAP访问
  - 能通过 ADO.NET进行有限的访问
- 本地代码
  - Active Directory Service Interfaces (ADSI)
  - LDAP Win32 API(C and C++)

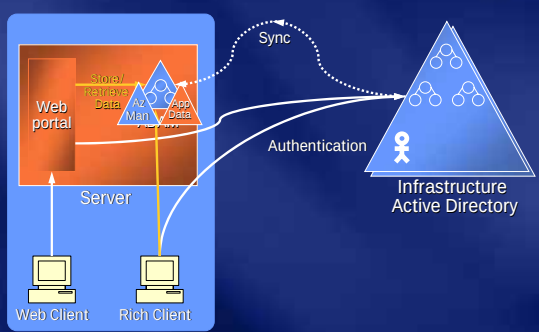
## 托管代码实例

```
-- UsersContainer entry
        return new DirectoryEntry(string.Format("CN=Users,{0}",
        partitionName), null, null, AuthenticationTypes.Secure |
        AuthenticationTypes.Sealing | AuthenticationTypes.Signing);

-- SaveAsset (SortedList attributes, string className)
        string cn = ((AttributeInfo)attributes["cn"]).Value;
        // Add a new but empty child entry to the Asset container
        DirectoryEntry newEntry = AssetContainer.Children.Add(
                string.Format("CN={0}", cn), string.Format("{0}{1}",
                schemaPrefix, className));
        // Commit the changes
        newEntry.CommitChanges();

-- Search in the directory
        DirectorySearcher search = new DirectorySearcher(topEntry);
        search.Filter = string.Format("({0})", entryCriteria);
        return search.FindAll();
```

## 数据存储的解决方案




## demo

### 身份和数据存储

我们将学习到:

- User in AD and app data in ADAM
- Schema and data in the store and how schema driven app is extensible
- Code to access directory data – simple and easy to develop apps rapidly