spring source
A division of VMware

springOne 2GX

Open Source VI Java API for Managing VMware Platforms

Steve Jin – VMware Inc.

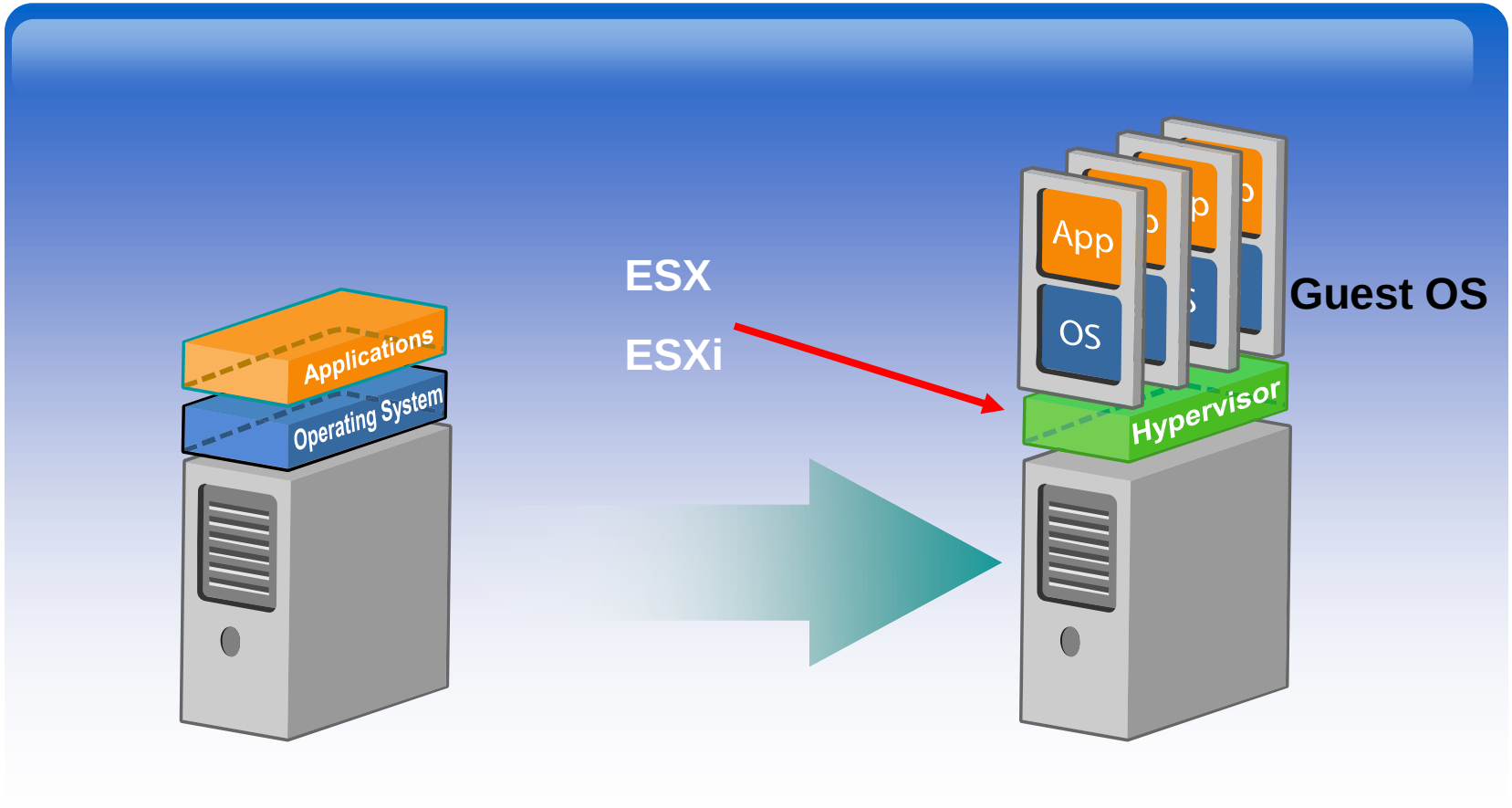
About the Speaker

- Author of VMware VI and vSphere SDK (Prentice Hall, 2009) and other two books.
  - http://www.amazon.com/dp/0137153635/
- Founder of the VI Java API project
  - http://vijava.sf.net
- Sr. MTS at VMware Engineering
- Previously, engineering and management roles at IBM Research, Rational, ASDC, etc.
- Ph.D., prestigious Tsinghua University

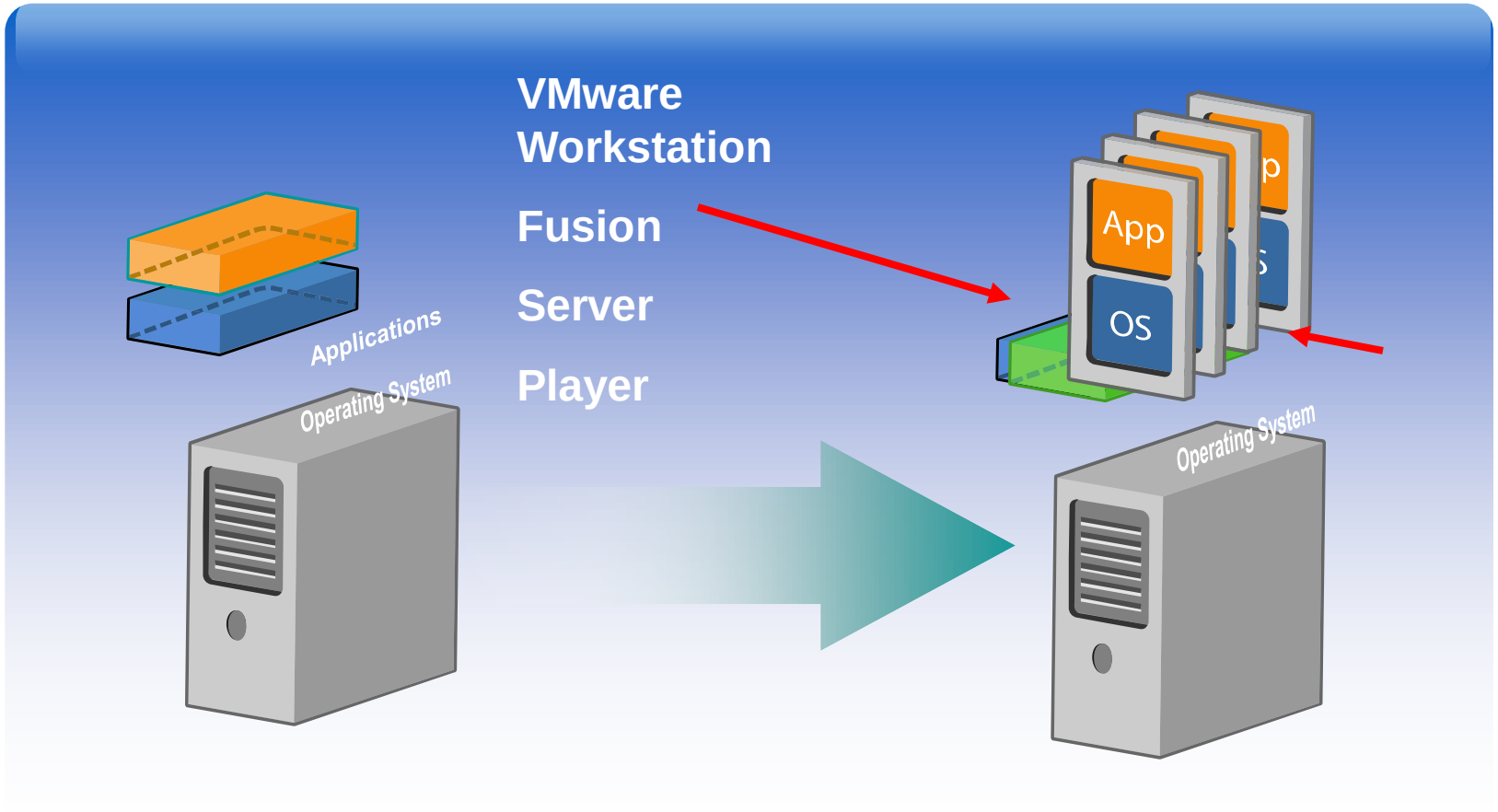
Agenda

- Virtualization 101
- VMware APIs
- VI Java Overview
- Architecture and Implementation
- New in 2.0
- How to Use the API?
- Use Case Samples
- Jython Scripting

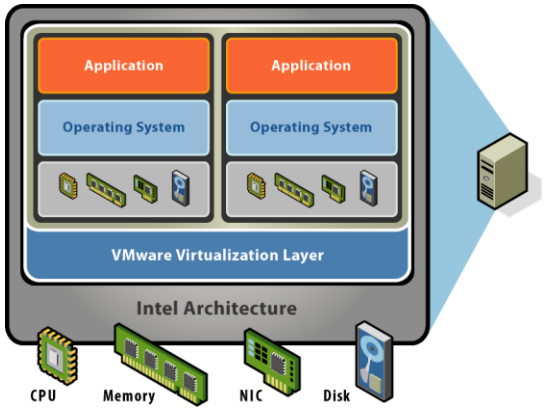
Bare Metal Architecture

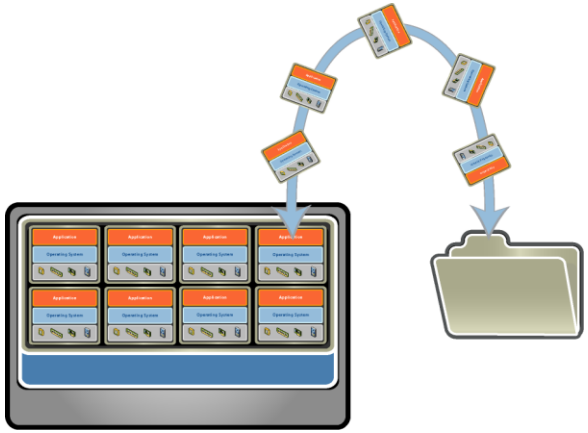


Hosted Architecture



Why Virtualization?





Partitioning

- Run multiple operating systems on one physical machine
- Fully utilize server resources
- Support high availability by clustering virtual machines

Isolation

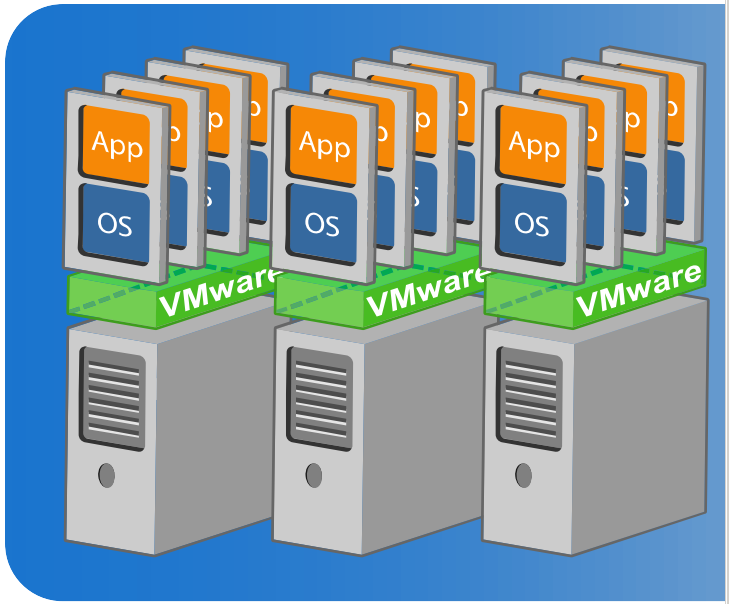
- Isolate faults and security at the virtual-machine level
- Dynamically control CPU, memory, disk and network resources per virtual machine
- Guarantee service levels

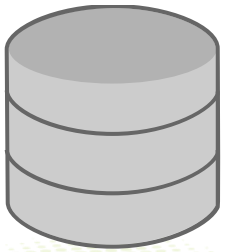
Encapsulation

- Encapsulate the entire state of the virtual machine in hardware-independent files
- Save the virtual machine state as a snapshot in time
- Re-use or transfer whole virtual machines with a simple file copy

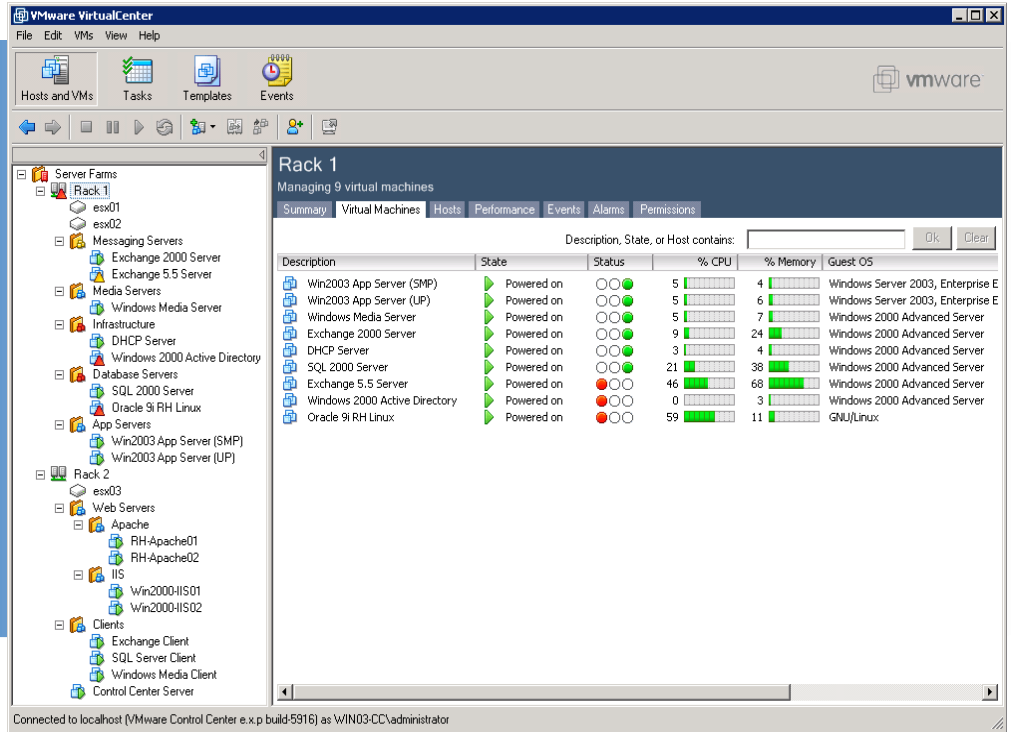
Centralized Management : vCenter

VMware vCenter





SHARED
STORAGE



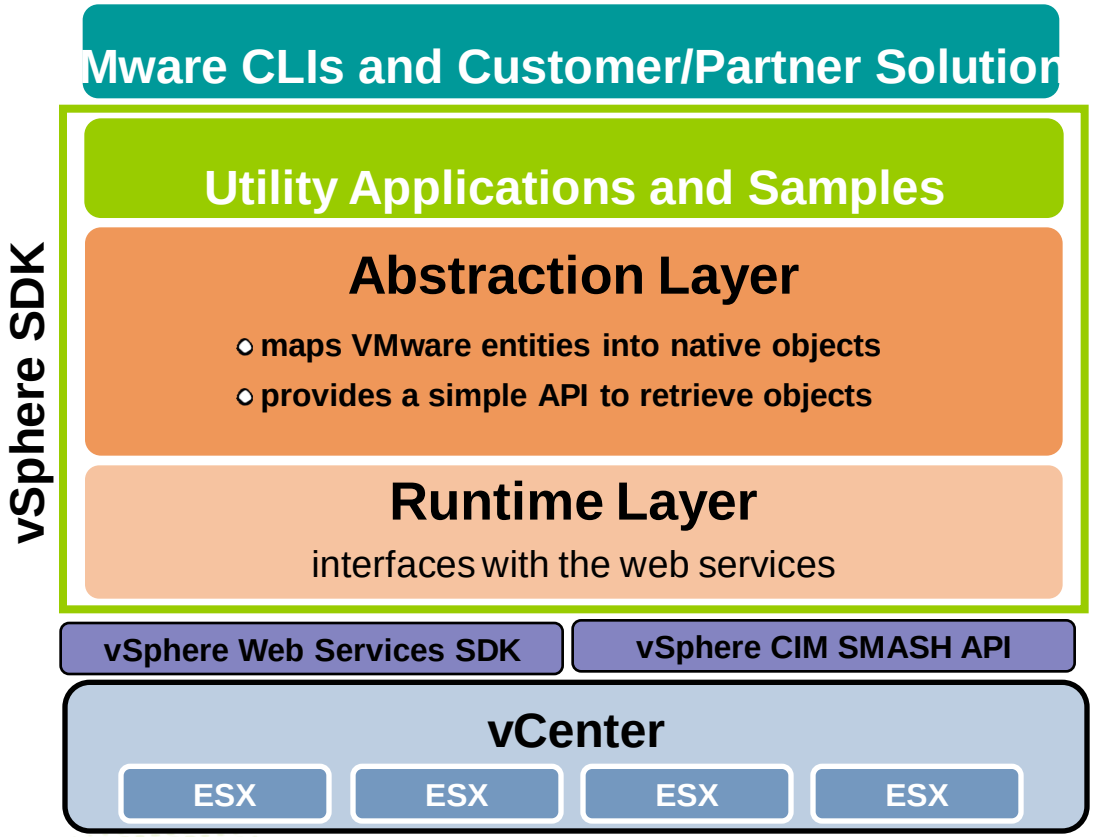
Agenda

- Virtualization 101
- VMware APIs
- VI Java Overview
- Architecture and Implementation
- New in 2.0
- How to Use the API?
- Use Case Samples
- Jython Scripting

VMware APIs/Tools Overview

- vSphere Web Service SDK (a.k.a. VIM API)
  - VI Perl; vSphere CLI; PowerCLI; VI SDK for Java
- vCloud API
- CIM SDK (SMASH, SMI-S)
- vSphere Guest SDK
- VIX API
- VDDK
- vProbes
- VMware Studio
- Others

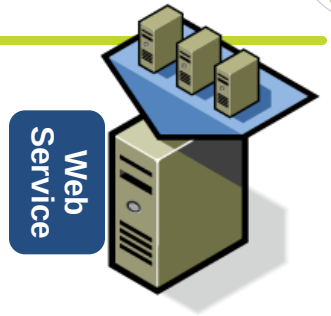
vSphere SDK Architecture



vSphere Web Services SDK

- ESX, ESXi, vCenter and VMware Server 2.0
- Comprehensive interfaces for vSphere virtualization management
- Authentication credentials must be provided to access the interfaces
- Updated each time the vSphere includes new features – typically during major and minor vSphere releases
- Maintain backward compatibility for 2 releases

vSphere Web Services SDK



- Provides interfaces to do the following:
  - Inventory retrieval (Hosts, VMs, Virtual Devices)
  - Virtual Machine management
  - Host configuration (storage, networking)
  - Performance monitoring and management
  - Event/Alarm management
- Sample utilities are included with the vSphere SDK package to demonstrate use of the APIs.

Agenda

- Virtualization 101
- VMware APIs
- VI Java Overview
- Architecture and Implementation
- New in 2.0
- How to Use the API?
- Use Case Samples
- Jython Scripting

History & Milestones

- May 22, 2008 – 1.0 Alpha
- Oct 18, 2008 – 1.0 GA
- Dec. 29, 2008 – first 1,000 downloads
- Jan 26, 2009 – 1.0 U1
- Feb 05, 2009 – 2.0 Beta
- April 4, 2009 – 2,000 downloads
- June 25, 2009 – 2.0 GA
- July 15, 2009 – 3,000 downloads
- Oct. 19, 2009 – 4,000 downloads

Key Statistics

- Downloads (4,040 as Oct 21,2009)
- Bugs (2 open/71 total)
- Developers on project: 11
- Discussion forums (public): 4, containing 377 messages
- Feature Requests (13 open/25 total)

Developer Testimonials

"We switched to using the VI Java API the day of its 1st alpha release in the Hyperic HQ plugin for VI3 and haven't looked back. **Using the VI Java API is concise and intuitive, yet powerful and solved all the challenges we'd seen previously when using the SDK. We're also thrilled with the performance improvements and removal of dependency jars after upgrading to VI Java 2.0**. Looking forward to tapping into the new vSphere features as well. A big thank you to Steve and the VI Java team!"

Doug MacEachern, CTO, Hyperic

Many more at: http://vijava.sourceforge.net/testimonial.php

What was the Motivation?

- Managed Object types exist in the VI SDK API reference, not in WSDL stubs. Instead ManagedObjectReference is used to represent all managed object types.
  - No type information for error catching at compile time
  - Interface signature can be confusing at the first time. See the _this parameter defined in all the methods.
- Web service APIs are all flattened out, not easy to choose one method from 300+ methods.
- PropertyCollector is powerful, but not easy to use. Also code to get a property is too long for its purpose.

What was it like with Web Services?

```
public class VMEventHistoryCollectorMonitor {
   private static AppUtil cb = null;
   private VimPortType _service;          // All webservice methods
   private ServiceContent _sic;
   private ManagedObjectReference _propCol; // PropertyCollector Reference
   private ManagedObjectReference _searchIndex;

   // EventManager and EventHistoryCollector References
   private ManagedObjectReference _eventManager;
   private ManagedObjectReference _eventHistoryCollector;
   /**
    * Initialize the necessary Managed Object References needed here
    */
   private void initialize() {
   …
   // 100 lines of code and hard to read
```

What is it like with VI Java API?

```
public class EventHistoryCollectorMonitor
{
  public static void main(String[] args) throws Exception    {
    ServiceInstance si = new ServiceInstance(new URL(args[0]),
                                    args[1], args[2], true);
    EventManager evtMgr = si.getEventManager();
    if(evtMgr!=null) {
      EventFilterSpec eventFilter = new EventFilterSpec();
      EventHistoryCollector ehc = evtMgr.createCollectorForEvents(eventFilter);
      Event[] events = ehc.getLatestPage();
      for (int i = 0; i < events.length; i++) {
        System.out.println("Event: " + events[i].getClass().getName());
      }
    }
    si.getServerConnection().logout();
   }
} // 17 lines of code and doing the same thing
```

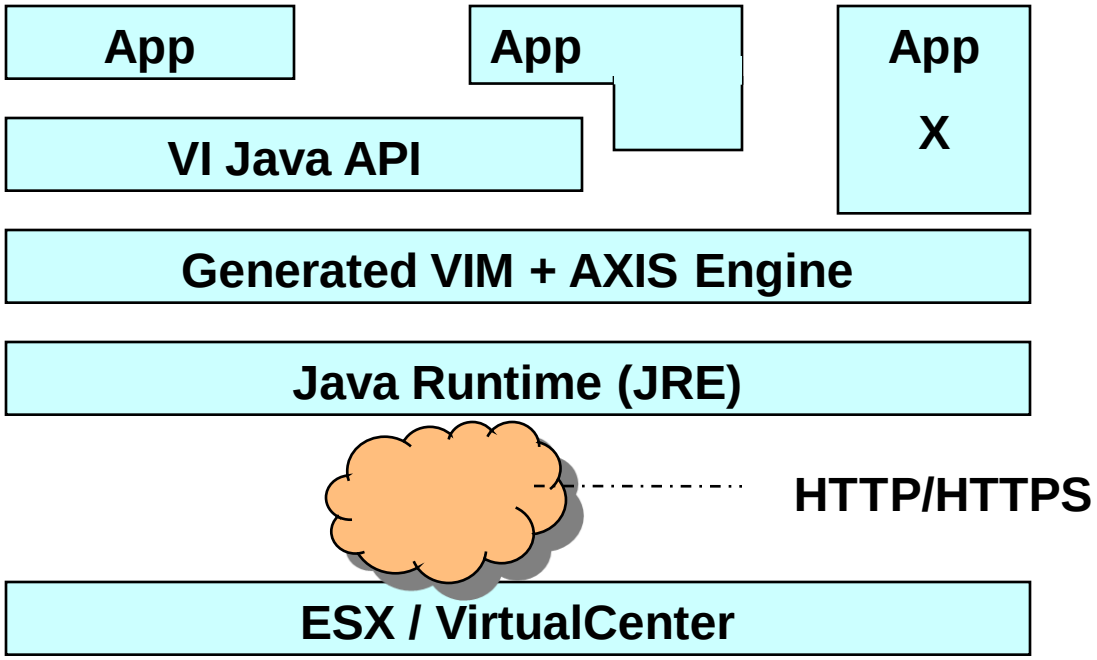
Agenda

- Virtualization 101
- VMware APIs
- VI Java Overview
- Architecture and Implementation
- New in 2.0
- How to Use the API?
- Use Case Samples
- Jython Scripting

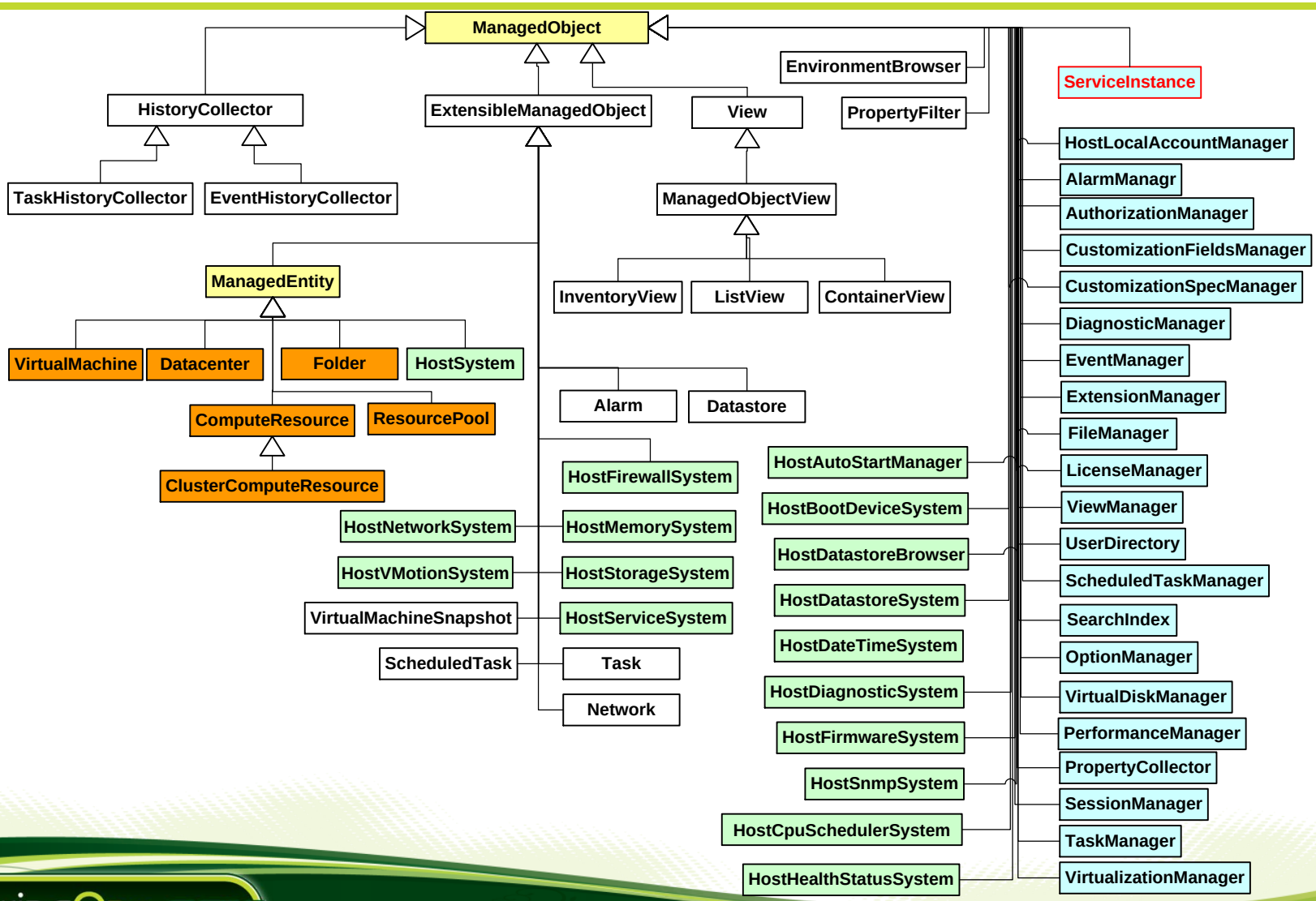
Design Goals

- Simple, but no simpler
- Easy to learn and use
- No tweak/change on WSDL & generated stubs
  - It won't cause support issue or confusion
- Built on top of web service interface, but doesn't hide it completely. Apps can switch between Java API and Web Services interface easily.

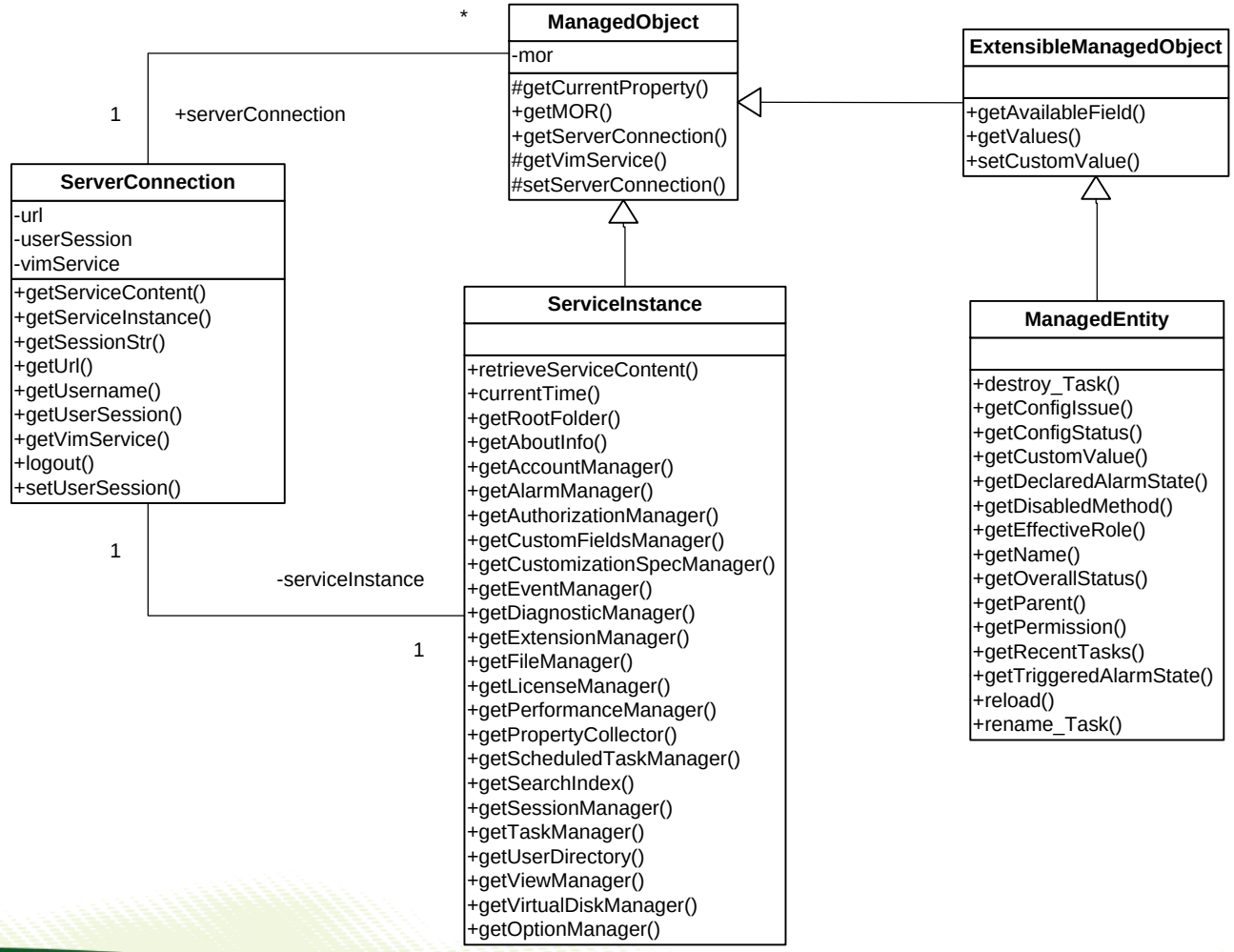
Architecture Overview (1.0)



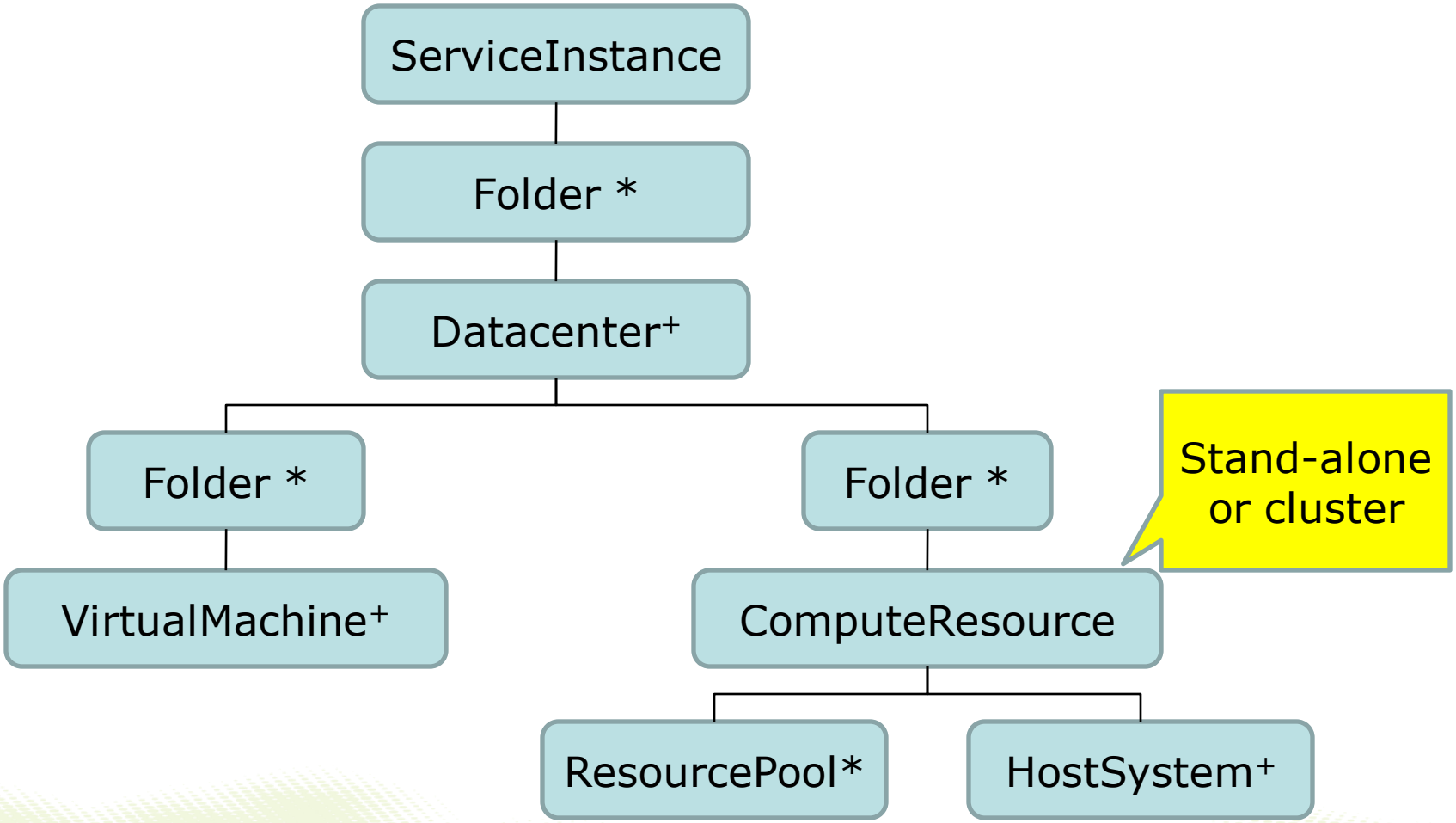
All Managed Types (in VI SDK 2.5)



Key Managed Types (Structurally)



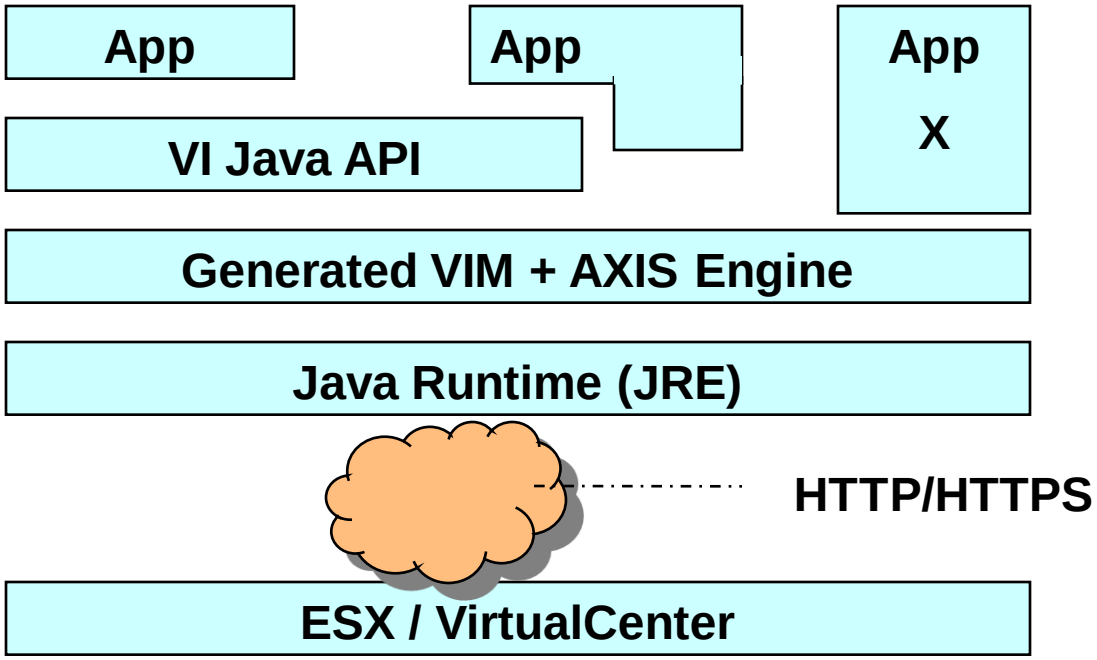
Inventory Hierarchy



Agenda

- Virtualization 101
- VMware APIs
- VI Java Overview
- Architecture and Implementation
- New in 2.0
  - WS engine, Client side REST, Caching Framework
- How to Use the API?
- Use Case Samples
- Jython Scripting

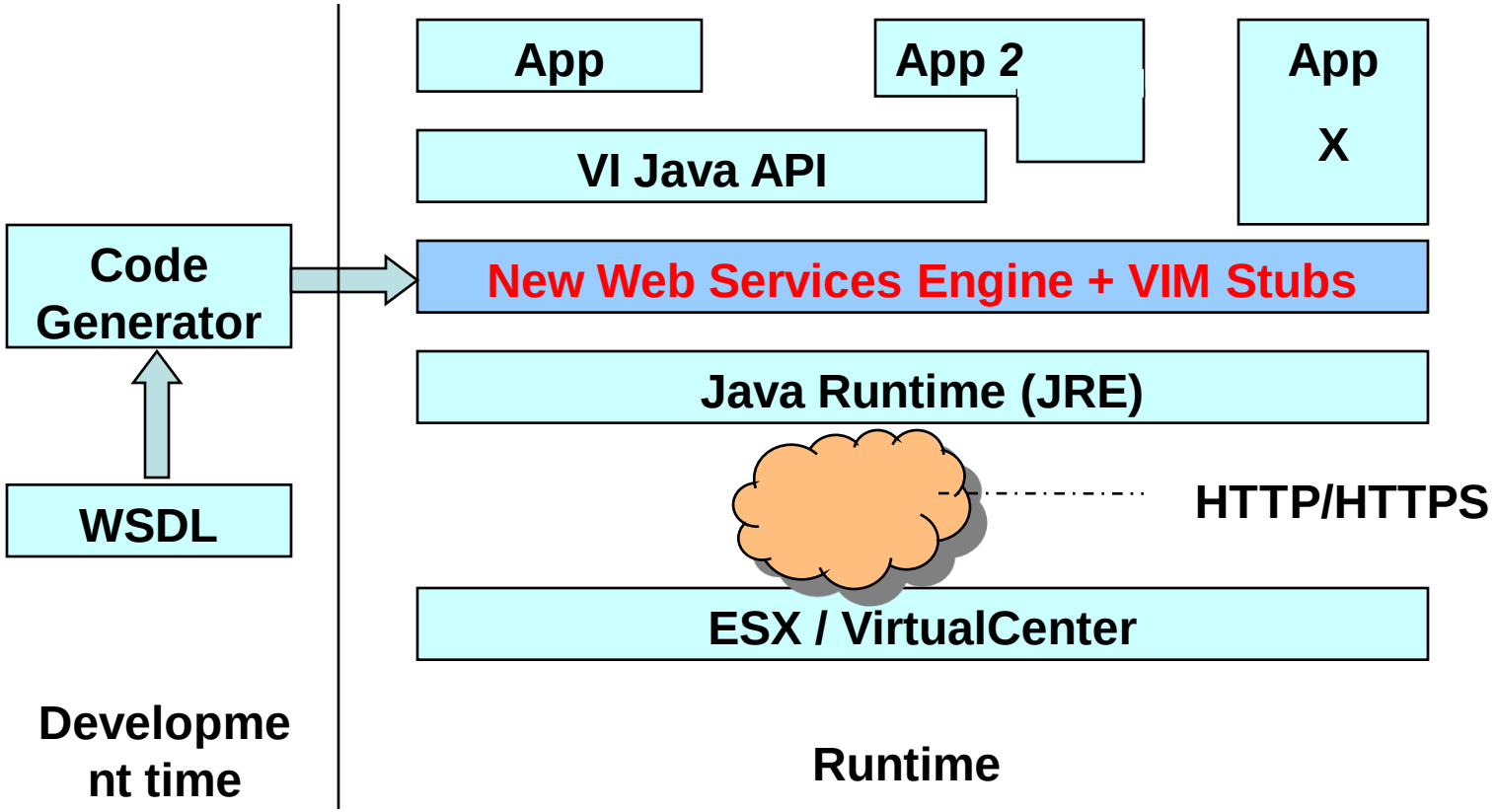
Current Software Stack



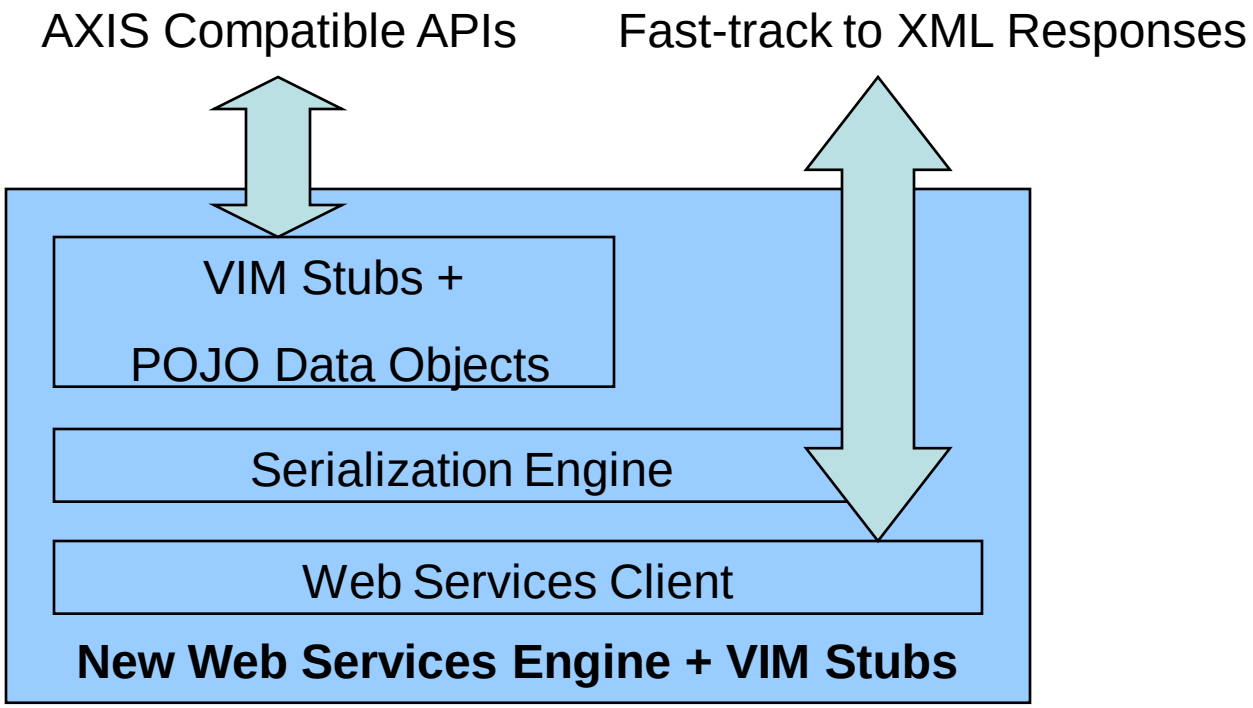
Motivation: The Problems

- Slow initial loading time
  - 3 to 6 seconds depending on the client machine
- Low performance in serialization & de-serialization
- Data objects are not clean
- Not flexible enough to get XML response directly
- Incompatible version due to namespace
- Dependency on libraries that VMware cannot re-distribute and have complications for end users
  - mailapi.jar, saaj.jar

The Solution



Architecture Overview



Data Object -- AXIS

```
package com.vmware.vim25;

public class AlarmAction  extends com.vmware.vim25.DynamicData  implements java.io.Serializable {
    public AlarmAction() {
    }

    public AlarmAction(
           java.lang.String dynamicType,
           com.vmware.vim25.DynamicProperty[] dynamicProperty) {
        super(
            dynamicType,
            dynamicProperty);
    }

    private java.lang.Object __equalsCalc = null;
    public synchronized boolean equals(java.lang.Object obj) {
        if (!(obj instanceof AlarmAction)) return false;
        AlarmAction other = (AlarmAction) obj;
        if (obj == null) return false;
        if (this == obj) return true;
        if (__equalsCalc != null) {
            return (__equalsCalc == obj);
        }
        __equalsCalc = obj;
        boolean _equals;
        _equals = super.equals(obj);
        __equalsCalc = null;
        return _equals;
    }
```

50 lines more

Data Object– VI Java API

```
package com.vmware.vim25;

/**
@author Steve Jin (sjin@vmware.com)
*/

public class AlarmAction extends DynamicData
{
}


// All POJOs, Enums
```

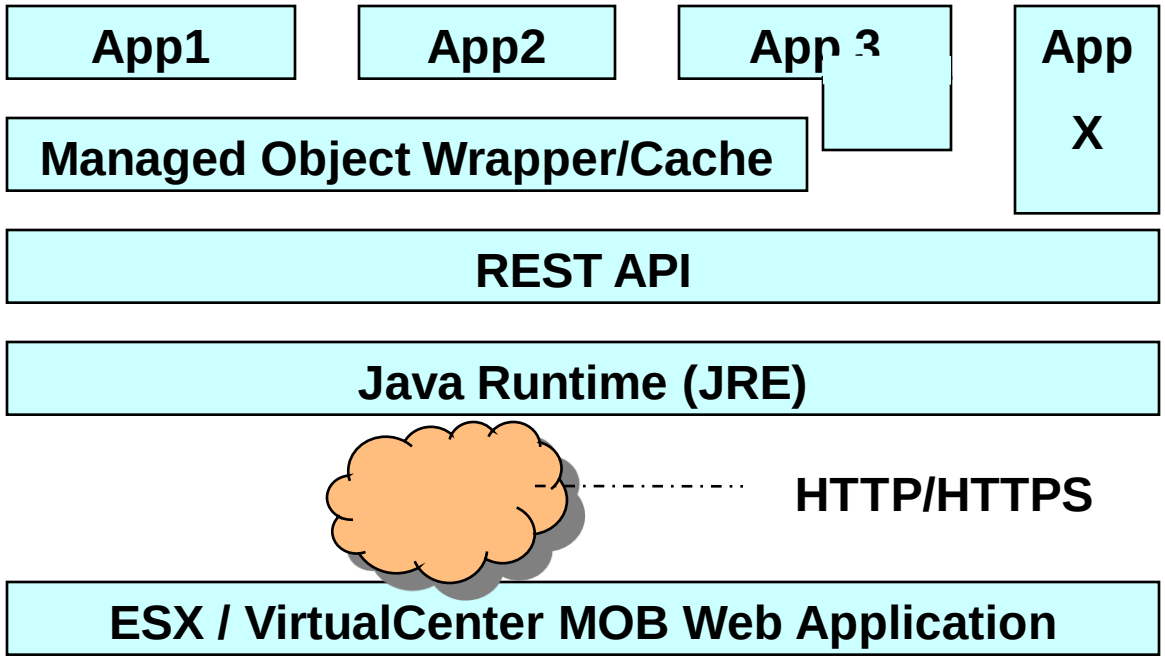
Key Results

- Super fast loading time compared with AXIS
  - 0.188s vs. 3.094s
- Much light-weighted library
  - 1.3 M vs. 5M in all jar files needed
- POJOs + Java Enum
- Up to 15 times faster in API calls
- Easy access to XML response for demanding apps
- No dependency on any other library other than J2SE and dom4j
- Version neutral: 2.0, 2.5, and 4.0

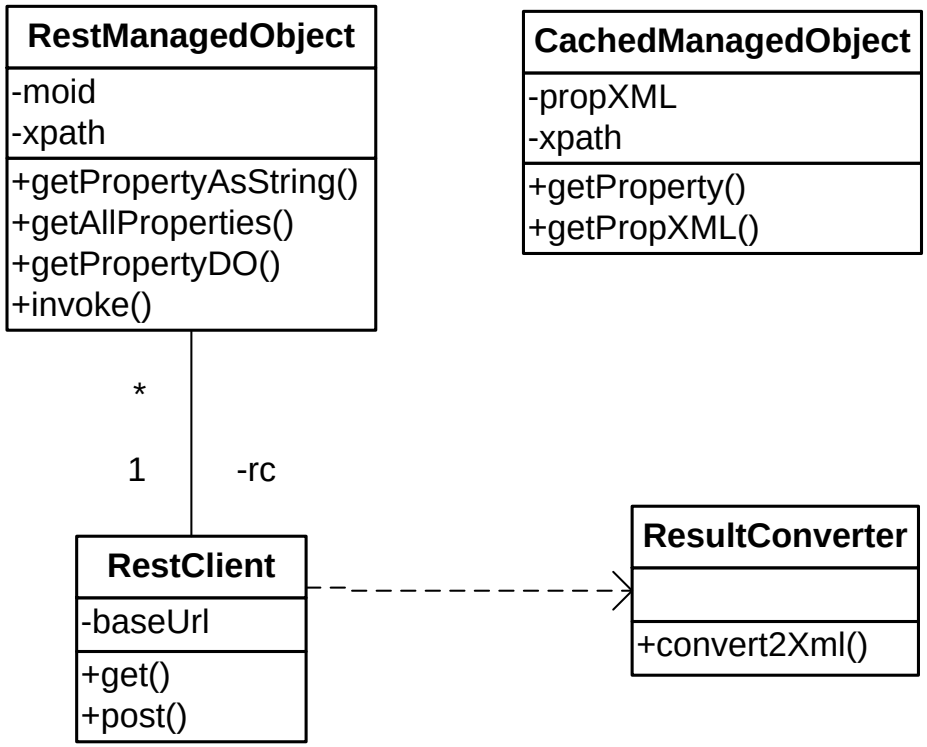
Agenda

- Virtualization 101
- VMware APIs
- VI Java Overview
- Architecture and Implementation
- New in 2.0
  - WS Engine, Client side REST, Caching Framework
- How to Use the API?
- Use Case Samples
- Jython Scripting

Architecture Overview



Object Model Design



Sample – Rest Level

```
RestClient rc = new RestClient("https://8.8.8.8", "root", "pass");
String contentXml = rc.get("moid=ServiceInstance&doPath=content");
System.out.println(contentXml);

XPath xpath = XPathFactory.newInstance().newXPath();
String emMOR = xpath.evaluate("//eventManager/text()",
        new InputSource(new StringReader(contentXml)));
System.out.println("view:" + emMOR);
String lastEventXml = rc.get("moid=" + emMOR + "&doPath=latestEvent");
xpath.reset();
String eMessage = xpath.evaluate("//fullFormattedMessage/text()", new
InputSource(new StringReader(lastEventXml)));
System.out.println("Latest Event: " + eMessage);
```

Sample – MO Level

```
RestClient rc = new RestClient("https://8.8.8.8", "root", "pass");
RestManagedObject si = new RestManagedObject(rc, "ServiceInstance");

System.out.println("name:" + si.getPropertyAsString("content.about.fullName"));
System.out.println(si.invoke("currentTime"));
System.out.println(si.invoke("retrieveContent"));

String allXml = si.getAllProperties();
System.out.println(allXml);
CachedManagedObject csi = new CachedManagedObject(allXml);
System.out.println("name:" + csi.getProperty("content.about.fullName"));
System.out.println("root:" + csi.getProperty("content.rootFolder"));
System.out.println("cap:" + si.getPropertyAsString(
        "capability.multiHostSupported"));
```

Key Benefits of Client REST

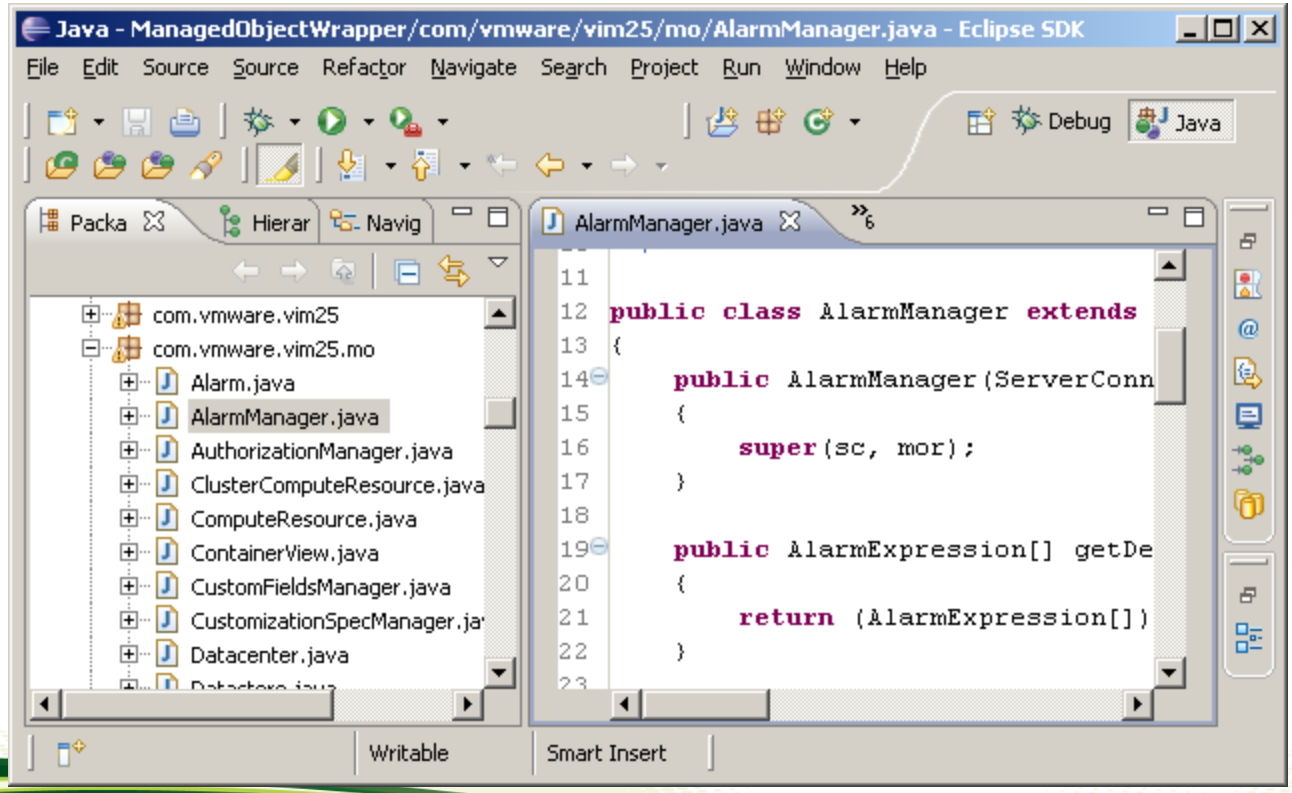
- Super fast loading time compared with VI SDK
  - 0.01 second vs. 6 seconds
- Super simple and light-weighted
  - Only 4 key Java classes
  - No dependency on any other library other than J2SE
- XML centric
  - Ready to apply XPath, XSLT, XQuery, etc.
- Version neutral
  - Works with 2.0, 2.5, and 4.0
- Can be easily ported to other languages

Agenda

- Virtualization 101
- VMware APIs
- VI Java Overview
- Architecture and Implementation
- New in 2.0
- How to Use the API?
- Use Case Samples
- Jython Scripting

Before You Get Started

- Download the VI Java 2.0 binary and include the two jars inside; or download the source and dom4j (www.dom4j.org), and include them. You are done.



Typical Flow using the APIs

1. Always starts with a ServiceInstance with URL/username/password, or URL/sessionID:

```
ServiceInstance si = new ServiceInstance(new URL(urlStr),
username, password, true);
```

2. From the ServiceInstance object, you can:
   - Get its properties or call its methods:

```
Capability cap = si.getCapability();
Calendar time = si.currentTime();
```

   - Get root folder object of the inventory tree.

```
Folder rootFolder = si.getRootFolder();
```

   - Get all different "manager" object or SearchIndex object:

```
SearchIndex searchIndex = si.getSearchIndex();
EventManager em = si.getEventManager();
```

Inventory Tree Navigation

- From the root folder, you can navigate all the inventory tree:

```
rootFolder.getChildEntity();
VirtualMachine vm = (VirtualMachine) new
InventoryNavigator(rootFolder).searchManagedEntities
("VirtualMachine")[0];
```

- All the items you see in the inventory tree is of type ManagedEntity.
- You can test the exact type of a ManagedEntity:

```
if(me instanceof VirtualMachine){ }
```

- You can call the methods defined on the exact subtypes of the ManagedEntity.

```
Task task = ((VirtualMachine)me).powerOffVM_Task();
```

What If You Have to Use M.O.R?

```
…
String sesssionStr = "vmware_soap_session=\"" + session_id + "\"";
ServiceInstance si = new ServiceInstance(new
        URL(urlStr), sesssionStr , true);


…
ManagedObjectReference mor = new ManagedObjectReference();
mor.setType(type);
mor.set_value(value);

ManagedEntity me = MorUtil.createExactManagedEntity(
            si.getServerConnection(),  mor);
…
```

Getting Help…

- For example, a method in the API:

  public Task powerOnVM_Task(HostSystem host) throws VmConfigFault, TaskInProgress, FileFault, InvalidState, InsufficientResourcesFault, RuntimeFault, RemoteException {}

- Description in VI SDK API Reference holds

PowerOnVM_Task

Powers on this virtual machine. If the virtual machine is suspended, this method resumes execution from the suspend point. When powering on a virtual machine in a cluster, the system might implicitly or due to the host argument, do an implicit relocation of the virtual machine to another host. Hence, errors related to this relocation can be thrown.

Required Privileges

VirtualMachine.Interact.PowerOn

Parameters

NAME	TYPE	DESCRIPTION
_this	ManagedObjectReference	A reference to the VirtualMachine used to make the method call.
host*	ManagedObjectReference to a HostSystem	(optional) The host where the virtual machine is to be powered on. If no host is specified, the current associated host is used. This field must specify a host that is part of the same compute resource that the virtual machine is currently associated with. If this host is not compatible, the current host association is used.

*Need not be set

Return Value

TYPE	DESCRIPTION
ManagedObjectReference to a Task	This method returns a Task object with which to monitor the operation.

Quick Comparison

	Java APIs	Web Services
Managed objects	**Yes**	**No**
OO programming	**Yes**	**Kind of**
Compile time type checking	**Yes**	**No**
Line of code	**≈30%***	**100%**
Application code readability	**Great**	**OK**
Productivity	**High**	**Normal**

- Note: * based on experience converting more than 10+ samples in VI SDK.

Agenda

- Virtualization 101
- VMware APIs
- VI Java Overview
- Architecture and Implementation
- New in 2.0
  - JEWSE, Client side REST, Caching Framework
- How to Use the API?
- Use Case Samples
- Jython Scripting

Use Case: Configure HostSystem

```
…
HostDateTimeSystem hdts = host.getHostDateTimeSystem();
HostDateTimeInfo info = hdts.getDateTimeInfo();

HostDateTimeSystemTimeZone tz = info.getTimeZone();
System.out.println("Name:" + tz.getName());

Calendar curTime = si.currentTime();
//roll back one hour
curTime.roll(Calendar.HOUR, false);
hdts.updateDateTime(curTime);
…
```

Use Case: Provision New VMs

```
…
VirtualMachineCloneSpec cloneSpec = new VirtualMachineCloneSpec();
cloneSpec.setLocation(new VirtualMachineRelocateSpec());
cloneSpec.setPowerOn(false);
cloneSpec.setTemplate(false);

Task task = vm.cloneVM_Task((Folder) vm.getParent(),
    cloneName, cloneSpec);
System.out.println("Launching the VM clone task. " +  "Please wait ...");

//alternatively, use Folder.createVM_Task() or Folder.registerVM_Task()
```

Use Case: Manage VM Lifecycle

```
// power on VM
Task task = vm.powerOnVM_Task(null);

// power off VM
Task task = vm.powerOffVM_Task();

//other operations include reboot, standby, shutdown, reset, suspend

// unregister VM
vm.unregisterVM ();

// destroy VM
Task task = vm.destroy_Task();
```

Use Case: Monitor System Performance

```
…
  PerformanceManager perfMgr = si.getPerformanceManager();
…
  PerfQuerySpec qSpec = createPerfQuerySpec(vm, pmis, 3, refreshRate);

  while(true) {
      PerfEntityMetricBase[] pValues = perfMgr.queryPerf(
        new PerfQuerySpec[] {qSpec});
      if(pValues != null) {
        displayValues(pValues);
      }
      System.out.println("Sleeping 60 seconds...");
      Thread.sleep(60*1000);
    }
```

Use Case: Automate System Mgmt

```
ScheduledTaskManager stm = si.getScheduledTaskManager();
…
// Note: the time should be fetched from server,
ScheduledTaskSpec oneSpec = createOneTimeSchedulerSpec(
        "ViMaster_OneTime", si.currentTime());
ScheduledTaskSpec weekSpec = createWeeklySchedulerSpec(
        "ViMaster_Weekly");

ScheduledTask st = stm.createScheduledTask(vm, oneSpec);
ScheduledTask st1 = stm.createScheduledTask(vm, weekSpec);
// sleep two minutes before deleting the one time scheduled task.
Thread.sleep(2*60*1000);
st.removeScheduledTask();
…
```

Agenda

- Virtualization 101
- VMware APIs
- VI Java Overview
- Architecture and Implementation
- New in 2.0
- How to Use the API?
- Use Case Samples
- Jython Scripting

What is Jython?

- "Jython is an implementation of the high-level, dynamic, object-oriented language Python written in 100% Pure Java, and seamlessly integrated with the Java platform. It thus allows you to run Python on any Java platform." http://www.jython.org/Project/index.html

- It looks like Python, and integrates well with Java.

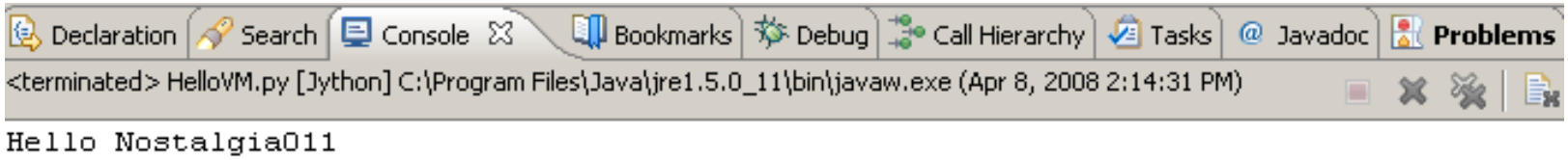
What do I need?

- JRE and Jython
  - http://www.jython.org/Project/download.html
  - FAQ: http://www.jython.org/Project/userfaq.html
- Jython IDE (JyDT plugin for Eclipse -- optional)
  - http://www.redrobinsoftware.net/jydt/
- VI Java API
- Or Jython Virtual Appliance (http://vijava.sf.net)

HelloVM.py

```
1 from java.net import *
2 from com.vmware.vim25.mo import *
3
4 si = ServiceInstance(URL("https://10.17.218.174/sdk"), "root", "password", True)
5 rootFolder = si.getRootFolder()
6 vms = InventoryNavigator(rootFolder).searchManagedEntities("VirtualMachine")
7
8 print "Hello " + vms[0].getName()
9
10 si.getServerConnection().logout()
```

- Output:



Calling Java from Jython

- First, include the Java libs in the Jython classpath
- Import the Java class in Jython code:

```
import java.net.URL as URL
import com.vmware.vim25.mo.ServiceInstance as ServiceInstance
from com.vmware.vim25.mo import *
```

- Use the Java object in your code:

```
si = ServiceInstance(URL("https://10.17.218.174/sdk"), "root", "vmware", True)
```

- More info @ http://www.jython.org/Project/userguide.html

Summary

- VI Java API provides a high level abstraction that makes the application development much easier and faster
- With 2.0, the included Just Enough Web Service Engine also boost your application performance up to 15 times
- Client REST API enables you to develop tiny apps
- Caching framework get you maintained caching easily
- The way I wrap up the VI Web Service interfaces can be used in any other APIs to be built on top of Web Services

Q&A

- http://vijava.sf.net
- Steve's E-mail: sjin@vmware.com