An Oracle White Paper
May 2011

Oracle Tuxedo: An Enterprise Platform for Dynamic Languages

ORACLE

Introduction

Dynamic languages, also sometimes known as scripting languages, have been in existence for a long while. Recently these languages have grown in significance because of various reasons. Dynamic languages are interpreted languages - no compilation and byte code, unlike C/C++ and/or Java. This makes the development, test and deployment cycle significantly shorter than alternative solutions. Once a change is made in the application code, it is ready for test and deployment. There is no compromise with programming language features either. These languages are feature rich. Many dynamic languages support object oriented programming and include features comparable to compiled programming languages. Also, there are a number of frameworks and libraries available to aid in application development, especially for web applications. For example, there are development frameworks such as Rails for Ruby, Django for Python and Symfony for PHP. These frameworks provide rich constructs for developers and make application development easier. Apart from such frameworks, a lot of sample code is available on the web, which is helpful in expediting the application development. Last, but not least, most of dynamic languages are intuitive and relatively easier to learn, which has resulted in availability of a large pool of programmers with skills in these languages. This increase in skilled pool size further reduces time-to-market and cost for new application development.

The origin of these dynamic languages is in the open source space. Due to this open source nature, these languages have evolved quickly and have a broad installed base. Absence of a standards body can be taken as negative, but there is enough required control mechanism in place for these, now mature, languages to be used in enterprise applications.

There are a large number of dynamic languages used for application development. Out of these dynamic languages, PHP, Python and Ruby have been around for many years and have matured nicely. In this paper, we focus on PHP, Python and Ruby exclusively. These dynamic languages are widely accepted. We have noticed that more and more enterprises are starting to develop new applications in these languages. These applications serve an important business function, which generally fulfill a critical need at the department level. These applications are sometimes called situational applications. Businesses require very quick turnaround time from concept to production-ready and hence these three languages serve well in this space.

Generally speaking, most of the situational application architectures include many components, such as an HTTP server for deployment, a dynamic language framework for development, etc. But there are no application servers around to serve as the container of application logic. This means there is no life cycle management, monitoring, or many other features that an application server is expected to provide. One of the reasons for

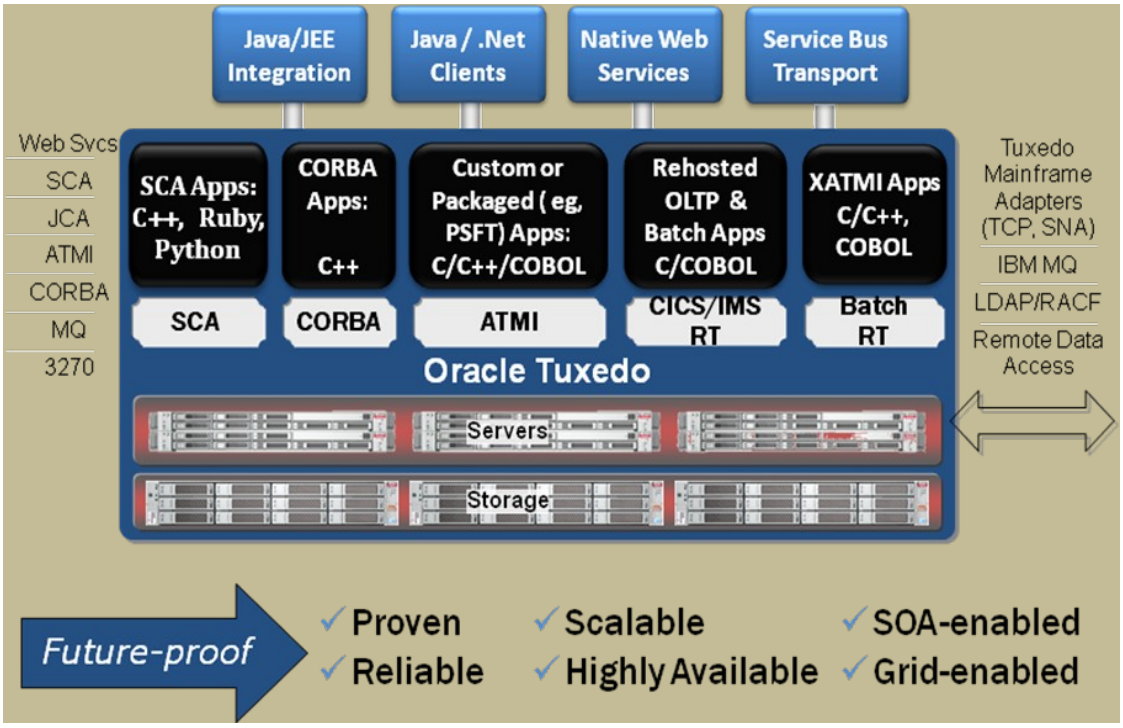
the absence of a good application server platform is that most of the situational applications when initially designed, do not really consider future RASP requirements: reliability, availability, scalability and performance. By definition, these applications are required to provide a quick solution to solve a specific problem and time-to-market is critical. But this does not really mean that situational application do not require scalability and high availability. Many of these situational applications grow with business. They become popular and critical due to their ability to solve users' current problems and ability to be quickly enhanced. Scalability and availability then become critical to preserve investment.

Also, eventually, as usage of these applications grows with the organization, they must communicate with other enterprise applications. These other enterprise applications could be written in Java, C/C++ or even in COBOL in a heterogeneous environment. There has to be a way for situational applications to be good enterprise citizens so that cross communication among various applications, whether situational or enterprise, can take place. So how do we solve this problem? Hold on to that thought for a few minutes while we introduce Oracle Tuxedo.

Oracle Tuxedo Overview

Tuxedo is an application server for developing and deploying applications written in C/C++, COBOL and now PHP, Python and Ruby. Tuxedo provides functionality as is expected of an application server, starting with a container for business logic written in various programming languages. Tuxedo also provides life cycle management for these services, such as starting/stopping, suspend/resume, dynamically starting more instances, restarts, etc. In addition, Tuxedo provides clustering capabilities for high availability and scalability.

Applications can be developed in one of the supported programming languages using one of the many programming models, such as XATMI, SCA or CORBA. Services and applications written in C/C++, COBOL, PHP, Python and Ruby can co-exist in the same application server and have equal access to all infrastructure services that the Tuxedo runtime provides. Obviously, it becomes simple for heterogeneous applications to interoperate with each other when hosted within the Tuxedo runtime. Tuxedo services can be developed using metadata driven application development methodology and without having to worry about any middleware specific APIs or infrastructure details.



Tuxedo can improve the scalability of database centric applications through its resource multiplexing features, while still using standard database drivers and APIs. For example to access Oracle Database, applications will still use cx_oracle for Python, while connections to databases are multiplexed through Tuxedo. In other words, not each and every client has to open a connection to the database. One connection to the database is established for each instance of the application and the same connection is reused for all incoming requests.

Linear Scalability

Tuxedo provides almost linear scalability with each additional hardware node added to the configuration. Applications can be deployed in various topologies to achieve such scalability using Tuxedo domains. A Tuxedo domain is similar to a cluster of nodes and provides location transparency of application services. Services can be located anywhere within the domain and incoming requests are automatically routed to the available server and service. Consumers of the services need not be aware of where its requests are being serviced. Tuxedo can make such decisions dynamically based on the configuration and workload.

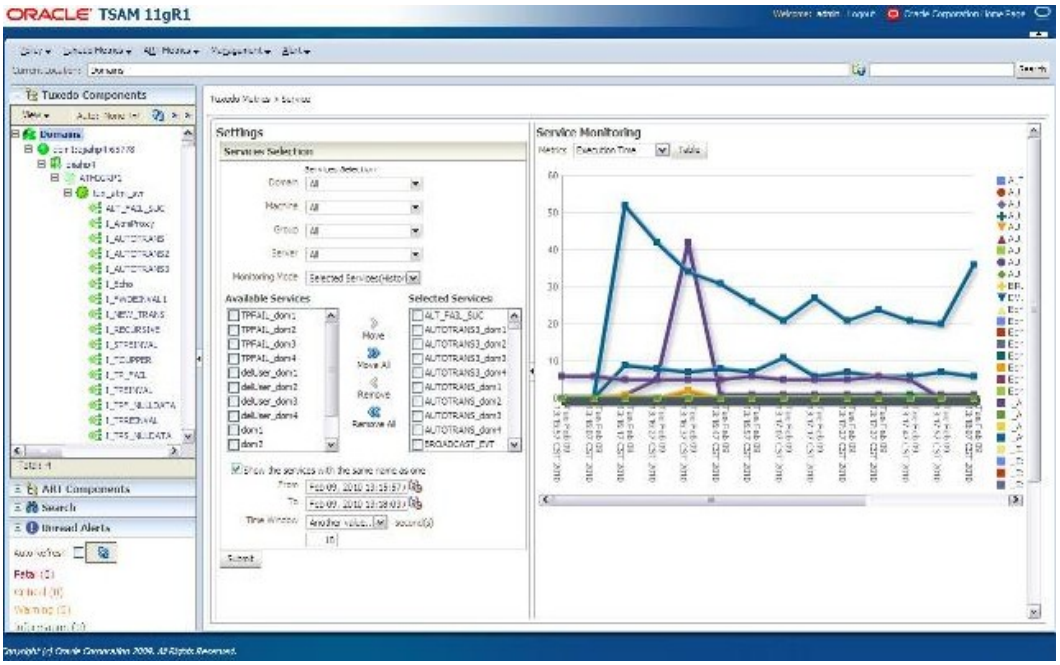
The Tuxedo domain deployment topology not only helps with scaling the application as business grows, but it can also eliminate any single point of failure if multiple instances of the same service are configured across the nodes of a domain or across domains. If

one of the nodes becomes unavailable, requests for a particular service will automatically be routed to any available node offering the service. Many Tuxedo applications provide five 9's of availability and more through such configurations. There are many installations of Tuxedo applications that are deployed using thousands of domains talking to each other.

Such configurations require no planned downtime, even for software and/or hardware upgrades. Nodes can be upgraded one at a time within the domain while rest of the nodes continues to operate normally, thus achieving rolling upgrades. Users of the application continue to receive uninterrupted service and do not see any outage.

Monitoring Tools

Almost all enterprise applications need tools to monitor applications in real time and collect statistics. Tuxedo System and Application Monitor, or TSAM, provides comprehensive monitoring information in real time, with data such as how much time was spent in a particular service call, or how much time it took for a message to reach the business service from its point of origin and so on. TSAM can also collect statistics including CPU usage for a particular service and response time for a service. All of this data can be extremely useful in identifying performance bottleneck or in determining root cause of a production problem in addition to capacity planning. Other features included in TSAM are: time spent by a message on a queue, XA transaction monitoring and service level agreement enforcement. Service level agreements can be set up on some of the services and configured such that an alert is generated or an action is taken automatically every time an SLA is violated. An example of an SLA would be response time of a mission critical service. If service response time exceeds a few milliseconds, TSAM could alert the manager on duty at the time of the occurrence.



In addition to TSAM, Tuxedo also includes tools for dynamic configuration and administration of applications. There is no need to bring the application down in order to make configuration changes. Configuration changes can be done dynamically using MIB or Management Information Base APIs. Tuxedo also provides SNMP interface to these MIB APIs, which allows Tuxedo administration through popular tools such as HP Open View, BMC Patrol etc.

Out-of-the Box Integration

In today's era of SOA and heterogeneity, most applications must be able to access services and applications hosted on totally different platforms. A JEE application must be able to access backend services written in C/C++ and vice-versa. Similarly a web application might need access to mainframe services and so on. Tuxedo provides several options to integrate with applications running on other platforms. Tuxedo includes a JCA Adapter, for bi-directional connectivity between a J2EE and Tuxedo application. Similarly, the Tuxedo Mainframe Adapter provides bi-directional connectivity between IBM mainframe and Tuxedo applications. Tuxedo also includes a native web service stack, which allows any Tuxedo service to be accessed using SOAP over HTTP. There are several other connectivity options that Tuxedo provides including access to Tuxedo services from .Net, MQSeries, and Java Clients.

For example, if a PHP service needs to call a web service, which is hosted in a Java based application server, it can be done by simply importing the WSDL of the web service to be invoked and by editing some of the existing configuration files. There is no

need to write any code. Similarly if a PHP service is to be accessed as a web service so that any standard web service client can access it using SOAP/HTTP, again it can be done using Tuxedo's native web service stack without needing to write any code.

Similarly if a PHP service needs to invoke a transaction running in the mainframe CICS environment, it can be done using Tuxedo's mainframe adapter. In fact, a CICS transaction can invoke a PHP service running within Tuxedo as well.

Where is Tuxedo used?

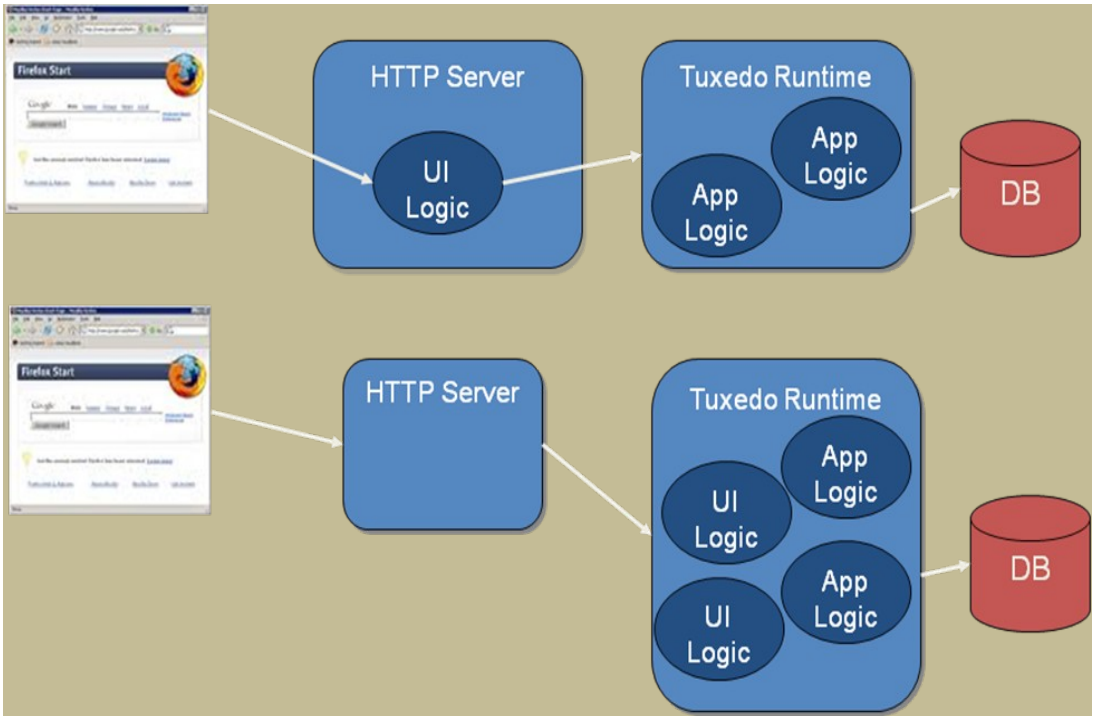
Use of Oracle Tuxedo is very pervasive across many verticals, including telecom, financial services, logistics, transportation, retail and public sector. Most organizations in these verticals reliably run their mission critical applications using Tuxedo infrastructure. For example, one of the largest retail chains in United States has a Tuxedo domain installed in more than 7,000 stores. Similarly, it is very likely that one of your financial transactions at some point or other is using a Tuxedo based application. In addition, there are many ISV applications, which embed Tuxedo. For example, all Oracle PeopleSoft installations run on Tuxedo.

Developing Enterprise Applications using PHP, Python, Ruby

As we discussed in the introduction, many situational applications are written in dynamic languages. Eventually most of these situational applications require the quality of service of an enterprise application. Dynamic languages provide a rapid application development environment. Tuxedo provides the best runtime infrastructure available. Oracle has combined the best of both worlds and created an enterprise platform for PHP, Python, and Ruby applications. Now applications in these dynamic languages can provide the same level of quality-of-service as their counterparts written in Java and C/C++/COBOL have had so far. Essentially Tuxedo provides a container to host business logic written in PHP, Python and Ruby, which can scale as well as C/C++ or Java applications. The application platform is neutral to the development environment – developers can use an IDE or framework of their choice. This platform can be used to develop new web applications and/or backend services with quality of service required by enterprise applications.

With Tuxedo, business logic and UI layers can be deployed together or separately. As depicted in the figure below, UI layer can be left in an HTTP server or deployed along with application logic within Tuxedo. With Tuxedo, there is almost unlimited flexibility in deployment: only a few or hundreds of instances of an application service can be deployed. Or the number of instances can be increased in real time responding to business requirements. All the instances of services, which are written in PHP, Python and Ruby, work in a clustered environment eliminating a single point of failure. The same

applies to UI logic or services which generates HTML code. The UI logic can be an application service and gets all the benefits of Tuxedo run time infrastructure.

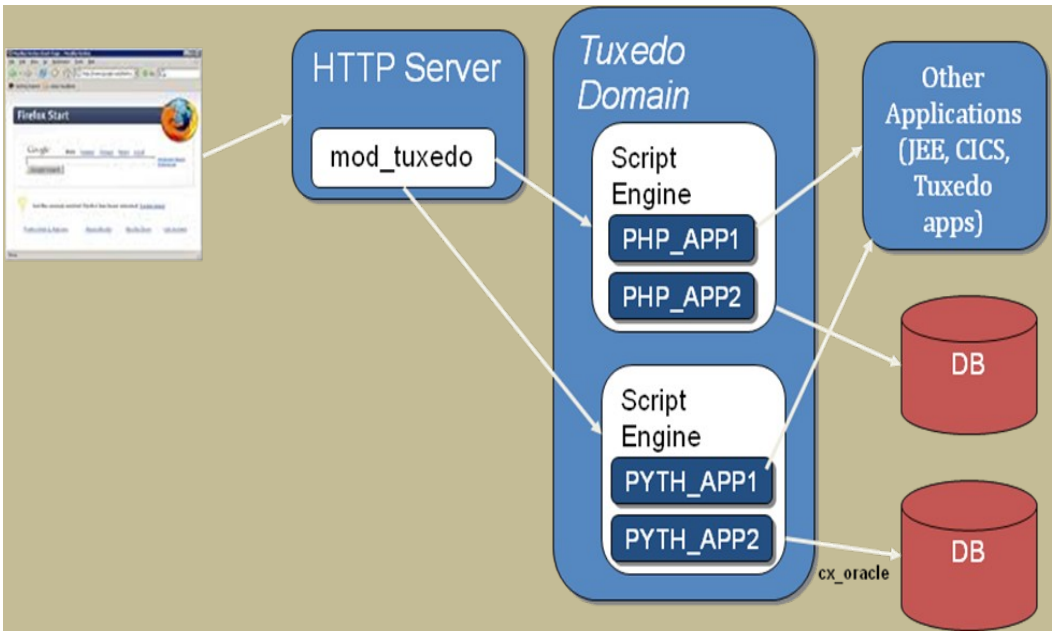


This flexibility in deployment opens up the world for PHP, Python and Ruby applications. These languages can now be used to write business logic for backend services, or alternatively to develop new web applications. And all these benefits of an enterprise runtime environment are available without losing the flexibility to develop applications in a personal choice of development environment.

Use Case 1: Developing new web applications

An organization may have a PHP or Python application, a situational application as discussed earlier. The application may have been developed in hurry – but no problem. Such existing applications can be deployed in Tuxedo's scripting engine as is – without any code changes. No code change is required to deploy existing application written in dynamic languages to run within Tuxedo infrastructure. It does not matter if application was written using one of the frameworks, such as Rails, Django or Symfony. Application may be calling a database directly or calling some other services. That is OK too. All such applications can be hosted with Tuxedo's scripting engine. As proof points, several common PHP open source applications have been deployed successfully in Tuxedo without any change to the application code. One of such applications is mediawiki, written entirely in PHP. Obviously, once this application is in Tuxedo runtime, it

immediately gets all the benefits of Tuxedo. It can now linearly scale as business grows and its availability improved. Also, it automatically becomes ready for integration with other applications either running in Tuxedo or in other application servers.



HTTP requests are passed to mod_tuxedo in the Apache HTTP server. Then mod_tuxedo directs the requests to Tuxedo's scripting engine, which in turn invokes the application. Tuxedo's scripting engine takes care of hiding details of any differences in runtime environments, if any.

Mod_tuxedo and script engine are two components of the solution – let's look at each of these one by one.

Mod_tuxedo:

Mod_tuxedo is an add-on module for Oracle HTTP Server, Apache HTTP server and Oracle iPlanet Web Server. It prepares and transforms an HTTP payload for delivery to applications written in PHP, Python and Ruby. It is a lightweight solution, intercepting HTTP calls and transforming those into Tuxedo service calls. Mod_tuxedo is configured in the same way as any other module for Apache, i.e. using httpd.conf. Mod_tuxedo also supports Apache processing modules, such as prefork, worker, etc.

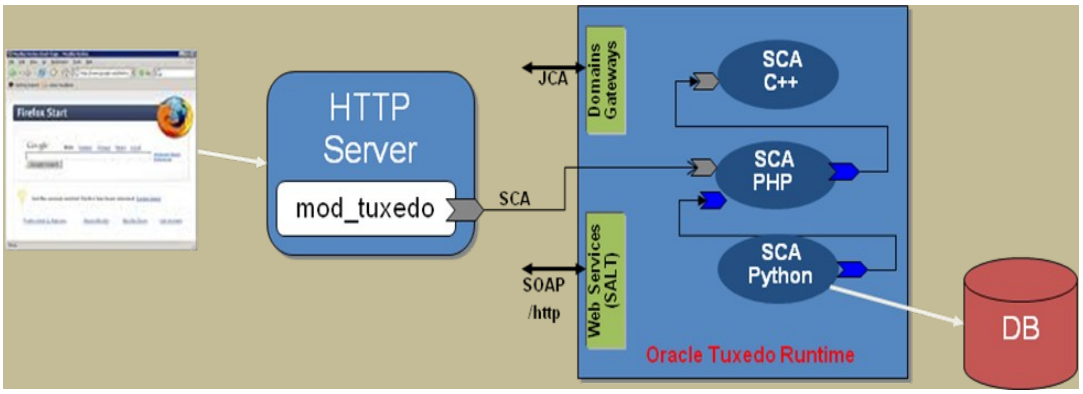
Script Engine:

Tuxedo runtime consists of a number of system servers; each system server providing various runtime services such as life cycle management, request routing, load balancing, transaction management, etc. The script engine is a system server to host applications based on FastCGI and written in PHP, Python and Ruby. Script engines can host one or

more applications and provide life cycle management services, such as start and stop, dynamic reload of applications, application caching, etc. Script engine inherits all the benefits of Tuxedo's runtime infrastructure. Script engine can be configured to have more than one instances located on the same node or distributed across multiple nodes for scalability and high availability. In addition, PHP, Python and Ruby applications can now be monitored through Tuxedo tools to collect various statistics and comprehensive performance data.

Use Case 2: Developing Business Services

An organization may plan to develop new business services using PHP. Such services could solve a specific business problem, for example recording sale of an item in a web store, doing credit card verification, etc. Or a business service might access a database on the back end and generate HTML code in return. How to develop such business services for Tuxedo deployments? Tuxedo provides a simple to use programming model, which is based on the Service Component Architecture (SCA) standard. Methodology to develop new services in PHP is the same as is for C++ and is metadata driven. Developer focuses on business logic and is Tuxedo unaware, i.e. there is no Tuxedo specific API that developer must learn and use. Business logic would look almost the same as if it were a standalone program.

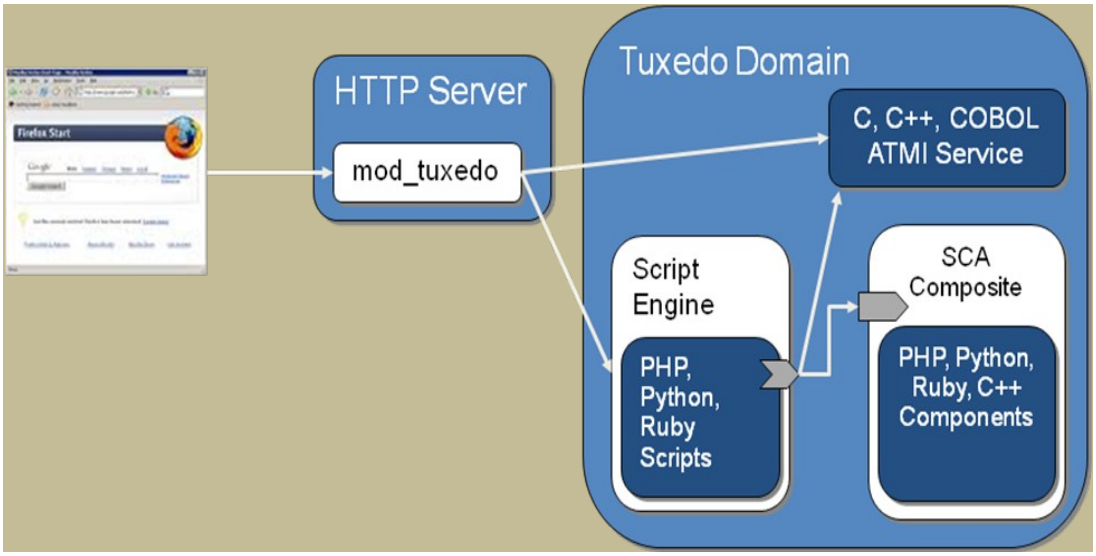


Once a business service is deployed it can be accessed using one of the many possible methods provided by Tuxedo, including access from an HTTP server or as a web service. This service, written in PHP, becomes accessible to services written in other languages, such as C/C++, COBOL, etc. Developers do not have to know the details of integration - all the infrastructure details are taken care of by Tuxedo.

Use case 3: Exposing Existing Tuxedo Services

An organization may have Tuxedo business services that have been working well as part of a large application. Most likely, these services are developed in C/C++ or COBOL.

Now, due to business reasons, these services must be accessed from a web browser. So how to connect a browser based HTML application to Tuxedo? Many applications have used a JSP/Servlet engine in front of Tuxedo. Now, a web application can be created in PHP, Python or Ruby, which talks to Tuxedo services on the back end and generates HTML code. Existing Tuxedo application does not need to change. As discussed in the previous use cases, HTTP requests are intercepted by mod_tuxedo and then forwarded to the script engine. The script engine in turn executes the web application and which makes calls to the existing Tuxedo services. Also by hosting the UI in the same container as back end services, IT operators and administrators will have one less application server to manage and monitor.



Conclusion

Oracle Tuxedo is the only industrial strength platform for PHP, Python and Ruby enterprise applications. With Tuxedo, PHP, Python and Ruby applications can provide same level of quality of service as applications written in Java, C/C++ etc. Tuxedo's proven reliability, availability, scalability and performance simplifies application development and deployment as developers focus on business logic and do not worry about runtime details. Moreover, these applications do not have to operate in a silo anymore and can serve as critical business needs as any other enterprise applications, while still maintaining the agility and all other benefits associated with dynamic languages.

Oracle Tuxedo: An Enterprise Platform for Dynamic Languages
May 2011
Author: Deepak Goel

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

SOFTWARE. HARDWARE. COMPLETE.