Service Oriented Architecture

Situation

The idea of Service Oriented Architecture (SOA) as well as the concepts behind it are often confusing to both Java developers and WebLogic administrators. Vendors such as Oracle sell a plethora of products tagged with "SOA" but, simply put, often fail to motivate how clients should use them.

At the same time it is well understood that time-to-market is too long for classic, monolithic application silos. Clients having to be decoupled from service implementations, that business logic shouldn't been hardcoded etc. - these experiences are now relegated to the past.

Solution

I would like to provide a deliberately simple, no-nonsense picture of an SOA here: A ‘product independent’ look at the different layers that constitute an SOA, from a system architecture perspective.

If you are only interested in Oracle product specifics, just treat this introduction as a bonus recipe (or skip it entirely and go for a swim or a run).

Service Oriented Architecture

Service Oriented Architecture (SOA) is an IT strategy that aims to narrow the gap between business requirements and IT solutions. SOA promises more agile business processes, faster implementations of IT solutions, and reduced long-term costs due to service reuse.

The key idea behind an SOA is to organize the discrete functions of enterprise applications into reusable and interoperable services. Services are first class citizens in an SOA representing the solution logic.

Workflow engines orchestrate the services; they contain the business process steps, and the sequence in which they are executed. These steps are graphically modeled by an executable workflow that can be both quickly changed as well as easily monitored.

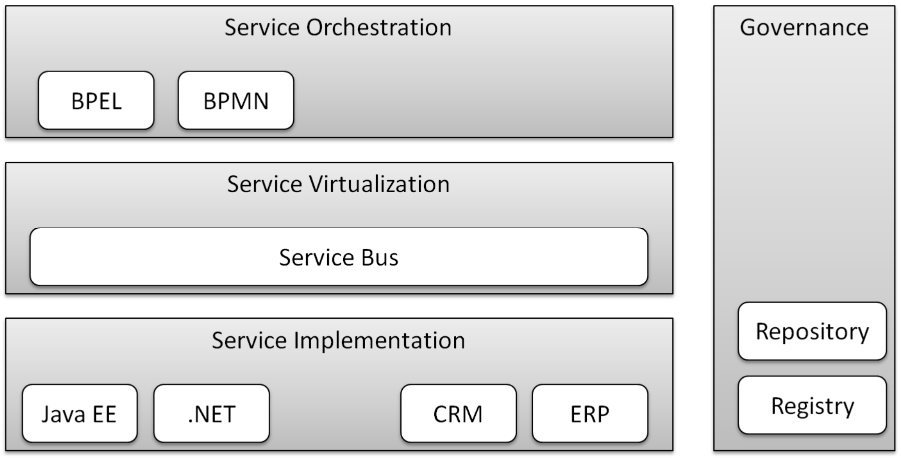
The Big Picture in Layers

If you are interested in a more academic view of SOA, I would recommend the OASIS model. However, please note: the OASIS model is an abstract model which can be used for a range of service oriented architectures and their analysis. It is difficult to understand, and surprisingly, it doesn’t contain a single diagram showing the layers of an SOA.

A full-blown service oriented design is often based on a surprisingly high number of middleware infrastructure components, which generally map to a vendor’s products. In real projects, you will likely find a subset of these products.

In the following paragraphs, the more popular components will be introduced and then mapped to concrete products in the following recipes.

Let's take a more detailed look at these layers, starting with the bottom of the stack. A simple SOA stack consists of a service implementation layer, a service virtualization layer, and a service orchestration layer, as shown in the following diagram:



CRM and ERP are used as examples of external systems providing services that can be used for their integration.

Service Implementation

The service implementation layer provides all services. Services are often implemented using either a Java EE application server or a .NET framework.

It sometimes seems as if all services in an SOA world have to be web services using SOAP over HTTP. However, the service implementation’s technical interface can be *any* transport mechanism, e.g. RMI for Enterprise Java Beans (EJB), JMS, email, or even FTP for legacy systems.

Also, the service implementation itself is not limited to Java EE and .NET. The service can be implemented using any language and protocol: starting from REST based web service implemented in Ruby or Groovy, to a legacy billing system written in C++ with an ftp interface expecting a flat file - anything is possible.

Service Virtualization

In the world of SOA, the service virtualization layer often replaces the traditional enterprise application integration (EAIs) system. A service bus is the principal performer here. Its tasks are to reduce the complexity of service integration, i.e., to bridge transport protocols and perform the message routing and message transformation. A service bus can also aggregate messages or callout to other services - but will never orchestrate services (service orchestration occurs in the orchestration layer).

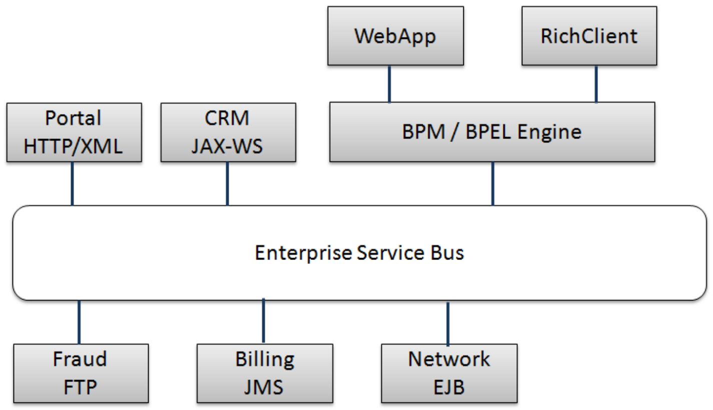
Configuration Driven

Now, the key feature of a service bus is that it is *configuration driven* and optimized for stateless routing with high throughput. Looking at the tasks listed above, you will realize that they can be achieved using an EAI broker. But in that case, you would have to build and deploy your changes. Whereas with a service bus, only a configuration change is required, and that can be done online (with a running bus).

Reduced Complexity

Since all clients talk to the service bus and the service bus forwards incoming service invocations to the services, the overall complexity of the distributed system is reduced. The service bus reduces a square number of interfaces between n instances (clients and services) to a linear number of interfaces.

The following figure is an illustration of this point using a telecommunications scenario. Typical components such as fraud management, billing, and network provisioning are shown with their (hypothetical) access mechanisms such as FTP, JMS or EJB call (RMI).



Decoupling

The service bus decouples the caller on the client side (which could be a BPEL or a BPMN engine) from the physical location, the data format, and the transport protocol of the service implementation.

Canonical Data Model

Larger companies, in particular, should consider putting a canonical data model (CDN) in place; this is also referred to as common data model (CDM). For example, every major car company probably has a unique data type representation of a car's tire throughout the company. As another example, a large telecommunications company would benefit from a unique data type representation of a SIM card or a customer address record that can be used in all subsidiaries across the globe.

The canonical data model is implemented in the virtualization layer. Only non-conforming client data formats (e.g. a proprietary client, which cannot be changed and has to be integrated after an acquisition) have to be transformed to the canonical data format of the service virtualization layer. In addition, the CDN of the service bus is transformed to the proprietary format of the service implementation, if necessary.

Service Orchestration

Service orchestration occurs in the top of the three layers. In this layer, multi-step business logic is executed by a Business Process Modeling (BPM) engine, or a Business Process Execution Language (BPEL) engine. Service orchestration is never performed by the service bus.

Instead of invoking the service implementation directly, the BPM system always invokes services which are exposed on the service bus and it is the service bus that decides where to route these calls.

A BPM engine, sometimes referred to as a workflow system, is well suited to long running processes (stretching over many years), human interactions (including web based forms), and the versioning of services.

A key benefit of using a BPM tool is that the business architect can model the workflow (usually with the support of a technical architect). The modeled workflow is then executed in the workflow engine.

This means the workflow is a visually easy to understand contract of the business model, which at the same time is executable. The workflow also serves as documentation for business processes – so the documentation is never out of synchronization with the business process.

When only orchestration is required on a technical level for web services, a BPEL engine is perhaps preferable to a BPM system. With BPEL, there is no human interaction (and no state is persisted between the individual steps of a workflow).

BPEL4People is an extension of BPEL that includes asynchronous communication with humans and supports four eye scenarios and escalations.

By orchestrating and reusing existing services in BPM or BPEL engines or other clients, we expect to reduce the gap between business requirements and IT solutions. This is because changes in the business logic do not require the service implementation to be rebuilt, tested, and deployed.

Governance

Governance is the process that ensures your SOA project is carried out according to best practices, architectural principles, legal and industry regulations etc.

OASIS definition

I'd like to give you a taste of the official OASIS definition, which is a touch more formal: "Governance is the prescribing of conditions and constraints consistent with satisfying common goals and the structures and processes needed to define and respond to actions taken towards realizing those goals."

These days, the importance of governance for SOA is widely understood. An SOA strategy can only be successful if governance is defined and enforced.

Governance is about service lifecycle management and service portfolio management. One governance issue pertains to determining what production ready service means: what metadata does a production ready service need? Who decides when it is in production? Where can you find the service? Whose number will be called if the service is not functioning properly?

Anne Thomas Manes has written an article about governance "The Elephant Has Left the Building", which I recommend you read, (my informal definition of governance is derived from this article).

Governance certainly involves a lot of documents, a lot of processes and a lot of communication. Registries and repositories are the middleware components that assist to enforce governance.

Registry

A registry is an electronic listing with metadata of your services. The key idea is that you somehow have to be able to browse and lookup your services, since you can only use or reuse a service if you know that it exists.

The standard that defines how to talk to a registry is called Universal Description, Discovery, and Integration, or UDDI. UDDI uses SOAP messages to look up and submit service listings to a registry.

There is an ongoing discussion about the usefulness of registries and where to position them. The idea of having a global UDDI registry failed to materialize when IBM, SAP and Microsoft announced their discontinued support.

In a classical SOA world with SOAP based web services, registries are often positioned as a runtime mechanism. I personally don't share this viewpoint, since too many unanswered questions remain: what if the registry cannot resolve a service at runtime? Can you afford to change your infrastructure so that every client does a service lookup in the registry at first and all available service implementations automatically register to the registry? What if your services are JMS, file or ftp based and haven't got a WSDL?

Best practice is to leave the runtime resolution of a service request as a task for the service bus. The service bus in turn uses a registry to import the service interfaces of the service implementations and to export its proxy services for the service bus clients.

Repository

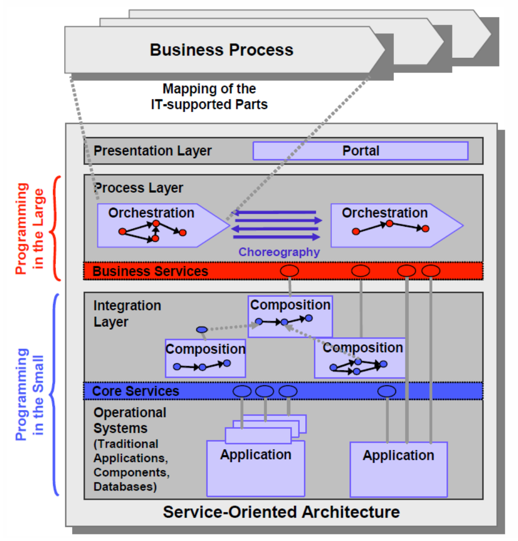
A repository is used at design time. It either stores files directly, or as a cross reference back to a document management system. Repositories become handy when resources, such as XSD files, are required at different locations; in order to ensure that everything is consistent, you want to guarantee that only one copy of a file exists.

The repository is responsible for all SOA assets and their relationships. It supports the lifecycle management and versioning, dependency management and impact analysis for change processes, as well as enforcing service reuse.

Terms

I guess it is time to start distinguishing between composition, orchestration, and choreography. These terms are often used interchangeably, and marketing brochures tell you that a service bus orchestrates services.

A good place to start would be taking a look at the work of C. Emig, who introduced the SOA layers in an early publication and uses composition for the integration layer (where an enterprise service bus would be be placed nowadays), orchestration for the process layer, and choreography between the processes.



Source: C. Emig et al.

More?

Links

OASIS SOA Model:

http://docs.oasis-open.org/soa-rm/soa-ra/v1.0/soa-ra-cd-02.pdf

Anne Thomas Manes "The Elephant Has Left the Building"

http://www.informationweek.com/software/business-intelligence/the-elephant-has-left-the-building/164301126

BPEL or BPMN?

http://redstack.wordpress.com/2012/02/13/choosing-bpmn-or-bpel-to-model-your-processes/

S. Abeck, SOA Layers

http://www.cm-tm.uka.de/CM-Web/05.Publikationen/2006/[EL+06]_The_SOAs_Layers.pdf

Oracle Fusion Middleware

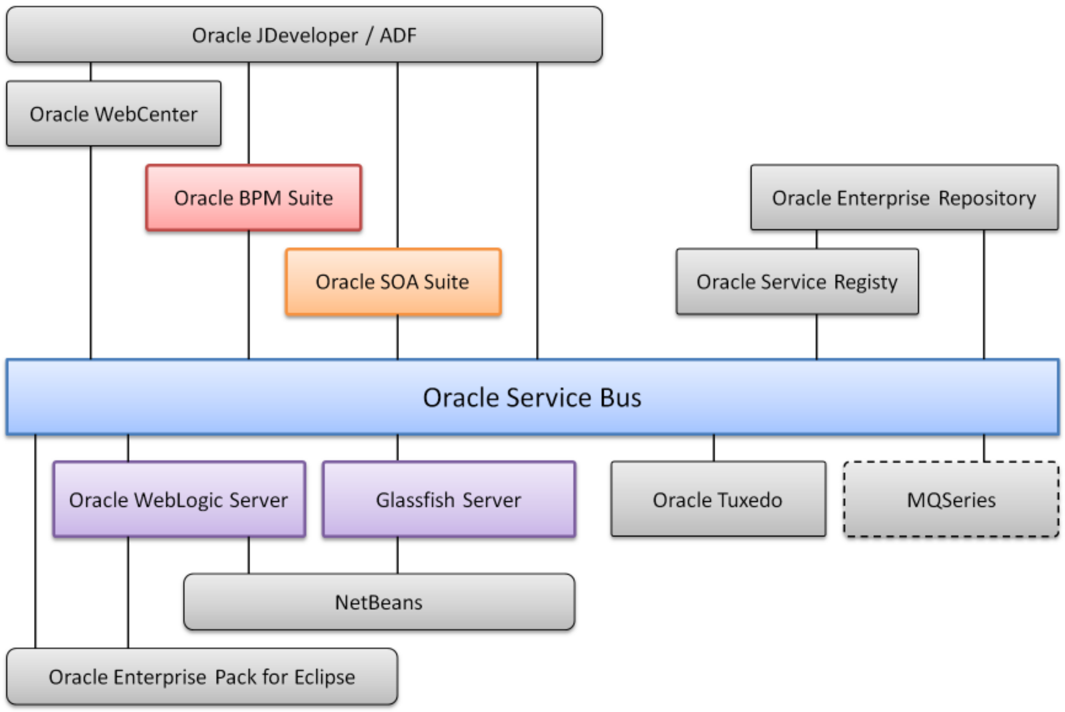
Oracle Fusion Middleware

Oracle Fusion Middleware (OFM) is certainly one of the most important and expansive middleware offerings currently available.

At the time of writing, Oracle Fusion Middleware comprises more than 70 individual products and bundles, most of which are beyond the scope of this introduction. We will therefore be dealing purely with those products that you would most likely use in your own environment, and the ones that you would encounter working as a consultant across industries.

The following list shows the Oracle products covered in detail in this book:

- ✓ WebLogic
- ✓ Oracle Service Bus
- ✓ Oracle SOA Suite
- ✓ Oracle BPM Suite
- ✓ Oracle Repository
- ✓ Oracle Registry

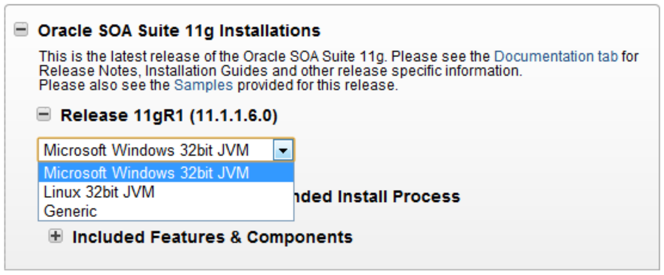


The figure above illustrates and OSB centric view of Oracle Fusion Middleware (MQ Series is from IBM and serves as an example for integration to third party middleware.)

Download and Installation

You can download Oracle Fusion Middleware from the Oracle download site. For the installation of an OFM product such as SOA Suite, several different components are necessary. Oracle made the selection of the products with matching versions and the download process really comfortable.

After choosing a product, first of all select the target operating system.



Oracle SOA Suite 11g Installations

This is the latest release of the Oracle SOA Suite 11g. Please see the Documentation tab for Release Notes, Installation Guides and other release specific information.
Please also see the Samples provided for this release.

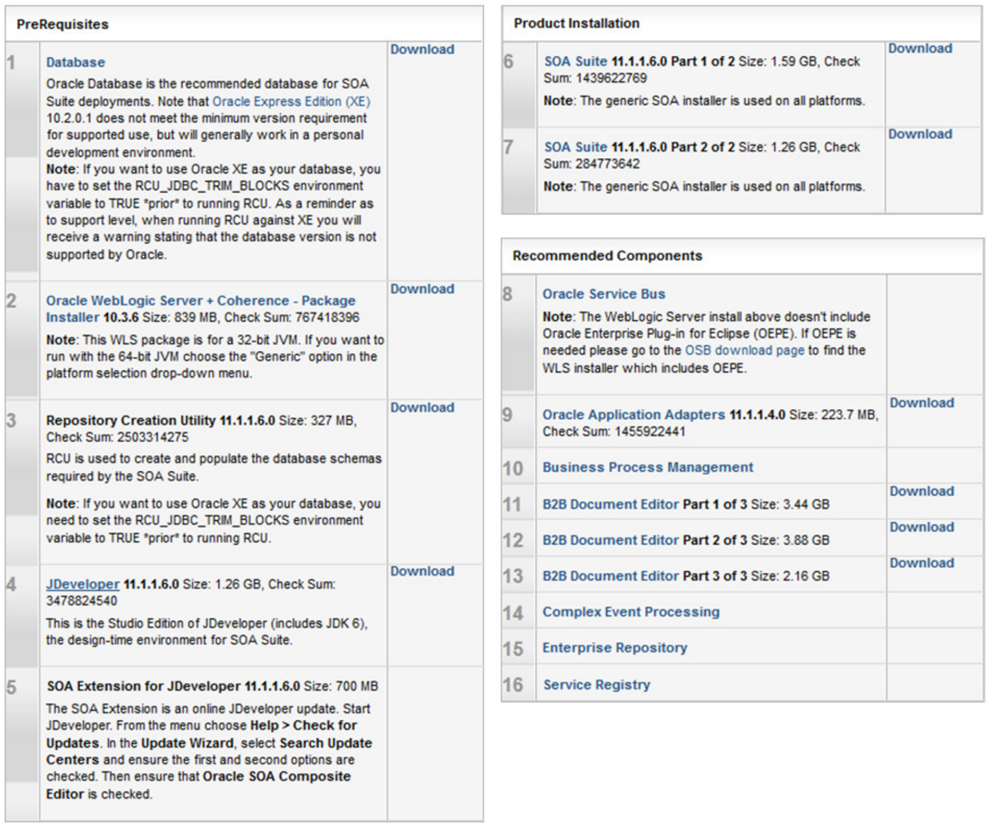
Release 11gR1 (11.1.1.6.0)

Microsoft Windows 32bit JVM
Microsoft Windows 32bit JVM
Linux 32bit JVM
Generic

nded Install Process

Included Features & Components

Then the matching version of all prerequisites, product installers, and recommended components will be displayed.



PreRequisites		
1	**Database** Oracle Database is the recommended database for SOA Suite deployments. Note that Oracle Express Edition (XE) 10.2.0.1 does not meet the minimum version requirement for supported use, but will generally work in a personal development environment. **Note**: If you want to use Oracle XE as your database, you have to set the RCU_JDBC_TRIM_BLOCKS environment variable to TRUE *prior* to running RCU. As a reminder as to support level, when running RCU against XE you will receive a warning stating that the database version is not supported by Oracle.	Download
2	**Oracle WebLogic Server + Coherence - Package Installer 10.3.6** Size: 839 MB, Check Sum: 767418396 **Note**: This WLS package is for a 32-bit JVM. If you want to run with the 64-bit JVM choose the "Generic" option in the platform selection drop-down menu.	Download
3	**Repository Creation Utility 11.1.1.6.0** Size: 327 MB, Check Sum: 2503314275 RCU is used to create and populate the database schemas required by the SOA Suite. **Note**: If you want to use Oracle XE as your database, you need to set the RCU_JDBC_TRIM_BLOCKS environment variable to TRUE *prior* to running RCU.	Download
4	JDeveloper **11.1.1.6.0** Size: 1.26 GB, Check Sum: 3478824540 This is the Studio Edition of JDeveloper (includes JDK 6), the design-time environment for SOA Suite.	Download
5	**SOA Extension for JDeveloper 11.1.1.6.0** Size: 700 MB The SOA Extension is an online JDeveloper update. Start JDeveloper. From the menu choose **Help > Check for Updates**. In the **Update Wizard**, select **Search Update Centers** and ensure the first and second options are checked. Then ensure that **Oracle SOA Composite Editor** is checked.	

Product Installation		
6	SOA Suite **11.1.1.6.0 Part 1 of 2** Size: 1.59 GB, Check Sum: 1439622769 **Note**: The generic SOA installer is used on all platforms.	Download
7	SOA Suite **11.1.1.6.0 Part 2 of 2** Size: 1.26 GB, Check Sum: 284773642 **Note**: The generic SOA installer is used on all platforms.	Download

Recommended Components		
8	**Oracle Service Bus** **Note**: The WebLogic Server install above doesn't include Oracle Enterprise Plug-in for Eclipse (OEPE). If OEPE is needed please go to the OSB download page to find the WLS installer which includes OEPE.	
9	**Oracle Application Adapters 11.1.1.4.0** Size: 223.7 MB, Check Sum: 1455922441	Download
10	**Business Process Management**	
11	B2B Document Editor **Part 1 of 3** Size: 3.44 GB	Download
12	B2B Document Editor **Part 2 of 3** Size: 3.88 GB	Download
13	B2B Document Editor **Part 3 of 3** Size: 2.16 GB	Download
14	**Complex Event Processing**	
15	**Enterprise Repository**	
16	**Service Registry**	

OFM Domains

For classical web and EJB development with WebLogic server, the easy and recommended approach is to deploy everything homogenously on every server. Therefore a classic WebLogic domain often comprises several almost identical managed servers (possibly with singleton services assigned to exactly one of them).

With Oracle Fusion Middleware the design of domains has changed. SOA Suite is often used together with Business Activity Monitoring (BAM) and Service Bus - then of course the question arises which *product* should be deployed on which server.

The following recipes explain domain layouts for the respective products.

Management

With every OFM release, management becomes more integrated with Oracle Enterprise Manager. Whereas Oracle Service Bus 11g is still used independently from OEM 11g, typical SOA Suite tasks such as deployment, monitoring and testing of components are fully integrated with OEM.

Oracle Fusion Middleware 12c

At the time of this writing, only WebLogic 12c and Enterprise Manager 12c are available. The full Fusion Middleware 12c suite is still to come. All OFM products are currently running on WebLogic 11g. For example OEM 11g is used for managing Oracle SOA Suite 11g.

Development Environments

There is a long history of integrated development environments (IDEs) for OFM.

Workshop

Starting with WebLogic 9, in which an Eclipsed based version of Workshop was bundled with WLS, which became increasingly useful and popular. Before WebLogic 9 there was another IDE named Workshop, which wasn't Eclipse based, and to put it politely, was no fun to work with at all.

Oracle Enterprise Pack for Eclipse

With WebLogic 10.3 - which was the first version of WebLogic released by Oracle - things changed. Oracle is now offering an Oracle Enterprise Pack for Eclipse (OEPE) that contains the necessary WebLogic plugins that you simply add on top of a bare Eclipse installation. The newest version of OEPE also supports the Glassfish application server.

Expect that in the future OEPE will only be supported for pure Java development. Eclipse is still the development environment for Oracle Service Bus 11g, however this is expected to change. (Actually, it has been expected to change now for almost 2 years...)

JDeveloper

The main IDE for Oracle Fusion Middleware is Oracle's JDeveloper. JDeveloper will be covering all development from Java to Fusion Middleware. At the moment it supports Oracle BPM and SOA Suite, but not yet Service Bus.

Looking at Java only, you can still develop with NetBeans, IntelliJ or Eclipse.

NetBeans

NetBeans made a big leap forwards. It has certainly become an excellent IDE for Java and Java EE development, and it provides many out of the box features that require installation plugins when trying to achieve the same with Eclipse.

NetBeans supports WebLogic but does not offer any support for OFM.

More?

This recipe was just the start. The following recipes describe Oracle Service Bus and Oracle SOA Suite in more detail.

Links

Oracle Fusion Middleware Overview, Downloads, and Documentation:

http://www.oracle.com/technetwork/middleware/fusion-middleware/overview/index.html

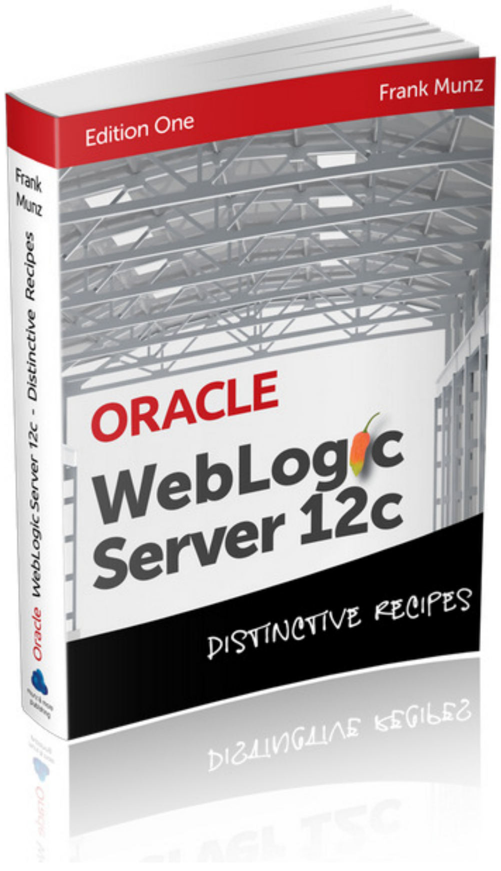
Books

Oracle Fusion Middleware Architecture and Management

http://www.amazon.com/Oracle-Fusion-Middleware-Architecture-Management/dp/0071754172/

Oracle WebLogic Server 12c - Distinctive Recipes (Architecture, Development and Administration)

Homepage:	http://wls12book.munzandmore.com
Amazon:	http://amazon.com/dp/0980798019
Webcast channel:	http://youtube.com/WeblogicBook
Book on Facebook:	http://facebook.com/WebLogicBook



Frank's details:

Twitter: @frankmunz

Blog: http://www.munzandmore.com/blog