



SOA REFERENCE ARCHITECTURE: WEB TIER

SOA Blueprint

A structured blog by Yogish Pai

Web Application Tier

The primary requirement for this tier is that all the business systems and solutions be accessible from any supported browser. This tier is the user interface or presentation tier and contains business logic for components such as enterprise infrastructure services and applications.

Packaged Applications

Typically, enterprises license the “best of breed” packaged applications that meet most of their businesses requirements, and then have their IT organizations and systems integrators tailor the packaged applications to meet their needs. Examples of such packaged applications are customer relationship management, enterprise resource planning, or industry-specific application suites.

Most packaged applications are now based on Internet protocols, which means that users can access many of the functions using any supported browser. Some of the latest applications can expose a limited set of functions as discrete callable services or externally controlled business processes.

Some of the best practices for leveraging packaged applications include:

- Limiting the amount of custom development, making it easier and cheaper to maintain and upgrade
- Attempting to achieve one standard implementation worldwide, thereby reducing integration and maintenance costs
- Leveraging the UI and the business process provided by the packaged applications, wherever possible
- Leveraging published application programming interfaces (APIs) rather than directly accessing the database.

Following are recommended approaches for taking the packaged application through the SOA maturity model:

Develop Web Applications

- Deploy the latest version of the application that is accessible by any browser; preferably a version that supports appropriate portal standards such as WSRP.
- Expose application services for consumption by custom applications, preferably as web services. This may require an adapter to enable access the application. Some recent versions of applications provide direct access to the application services through integration gateways or web services.
- Provide seamless user experience by incorporating the enterprise look and feel (templates, skins, skeletons, CSS) as well as integrating with the enterprise single sign-on solution.
- Externalize authentication by integrating to the enterprise identity and access manager (typically LDAP).





Develop Composite Applications

- Identify business objects that could be shared across the enterprise as composite applications
- Send event notifications (triggers) to the composite applications to initiate specific actions
- Modify business processes and user interfaces to enable the composite applications
- Expose additional business services so the composite applications can synchronize with the packaged application.

Automate Business Processes

- Understand and model business processes to identify opportunities for re-engineering
- Identify re-usable portions of business processes that can potentially be automated by a business process engine
- Expand the number of exposed services and business processes
- Reduce and consolidate the number of applications deployed.





Custom Applications

Organizations may choose to develop a custom application, to create a distinct brand and unique experience for their customers and partners. This requires providing a consistent seamless interface to internal and external users. Companies often prefer to develop a custom application rather than customize a packaged application, because:

- Making modifications to the user navigation or user interface for some core transactions is not easy
- As most of the major packaged applications are not based on open or standard technologies, their performance may not scale to the business needs
- Proprietary development model makes it difficult to find resources or rapidly deploy new business capability
- Integration to other technology is not straightforward, resulting in point-to-point integration and possibly poor data quality.

The three options for developing custom applications are:

1. Develop and deploy custom applications on an application server
2. Develop and deploy custom applications by leveraging a portal
3. Develop a thick client either using tools based on open standards or proprietary development tools.


For options 1 and 2, the first step for IT organizations is to determine the approach, infrastructure, and tools for developing custom applications. In addition, IT organizations need to define the governance and organization model to develop the custom solution.

For option 3, thick client custom applications are typically developed using SWING, Visual Studio, or similar tools. Most of these thick clients need to interface with some external systems and the recommended approach would be to leverage open standards such as SOAP, web services, XMPP, or WebDAV instead of directly accessing any external resources. This approach makes it easier for IT organizations to support and upgrade the integration.

Custom Applications Business Requirements

Most enterprises have already deployed external sites as well as multiple internal sites and applications to support the diverse needs of each of the business units. The first step is to standardize the look and feel of these sites and the infrastructure across the enterprise to make it easier for a customer, partner, or an employee to get the information they are seeking.

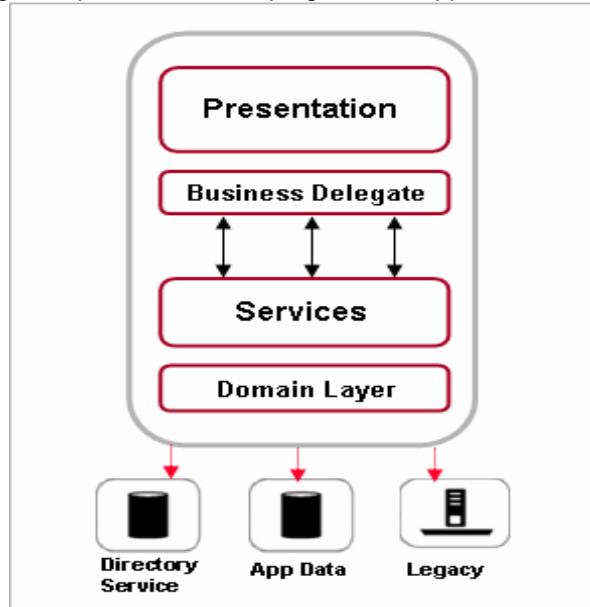
The business requirements for this phase include:

- Unify user experience on the external site, making it easy for potential users, partners, customers and analysts to find information that they are looking for
 - Standardize the look and feel across all internal and external sites, and the process and procedures for publishing content
 - Create one my<company name> site for all employees, contractors, partners, and customers to personalize the services and content
 - Provide secure access to confidential information for all internal and external sites
 - Provide a highly reliable, available, and scalable environment
 - Facilitate branding and accessing multiple applications through a common portal
 - Allow users to log in once and gain access to all their services
 - Personalize service based on roles and responsibility of the user
 - Reduce maintenance cost of maintaining multiple systems and applications by standardizing on one platform or environment
- 

- Standardize on one look and feel to eliminate multiple user training requirements
- Reduce operations and support cost, freeing IT to deploy scarce resources on developing new functionality.

Custom Applications Architecture Approach

As portals provide a proven set of capabilities in support of the presentation layer, most IT organizations have started standardizing on a portal for developing custom applications.



Recommended Custom Application Architecture

This recommended architecture would provide the following benefits:

- Follows SOA principles to promotes re-use at all levels
- Provides capabilities to deliver in weeks not months (once there is a stable framework in place)
- Leverages each product for what it is good at, for example, uses portal for presentation based on entitlements
- Allows business to combine services to deliver new capabilities
- Abstracts the data source and the relationship through a domain layer, thereby minimizing the impact of changes to the source systems
- Loosely couples presentation and the business logic to make it reliable and scalable.



Each of the layers in the proposed architecture plays a specific role:

Presentation layer: a Portal is responsible for handling all presentation services. Portlets drive the user experience where a portlet is a view on an application.

Business delegate layer: business delegates are components responsible for the communication between the presentation and the business layers. They abstract the communication details and complexities involved in making a call to the business layer. This layer includes a model view controller framework that helps users navigate through the web site.

Services layer: the services layer includes the capabilities of the application server. It is composed of stateless functions that expose high-level business functionality. It includes a session facade which is the entry point to the business layer and which abstracts away the details of handling fine-grained business entities from the presentation layers. Most of the business logic can be implemented directly on session facades or on a sub-layer of application objects.

Domain layer: the domain layer also uses the core application server. It is a collection of business entities that defines persistent business concepts. Technologies that handle database storage need to be used in this layer since these components represent persistent states. Entity Beans may be used to implement some of the components of the object model. Alternatively plain old Java objects (POJO) can be used with the help of data access objects (DAO) for persistence. Entity Beans are the preferred mechanism for implementing this layer but a combination of technologies may be required depending on the complexity of the object model.

Custom Applications Framework Components

Custom application framework components extend services that are inherent in the application server platform. Framework components include:

Data services: the persistence layer provided for the applications. The container management is robust enough to leverage CMP for most simple transactions, but DAO should be offered as an option for handling complex transactions.

Logging services: services used by applications to record and trace errors and activity. Types of message to be logged include debug messages to trace any issues, error or fault logging for diagnostic purposes, and activity logging for audit trail and usage analysis. Every enterprise should standardize the logging services used by applications, ideally leveraging the features provided by JDK 1.4 onwards. If the logging service is generic across the enterprise, it will enable the staff to more effectively determine performance or transaction bottlenecks. Logging services should standardize the mechanism, communicate it to the entire development community within the enterprise, and ensure compliance with the standard. No specific code needs to be developed for this service.

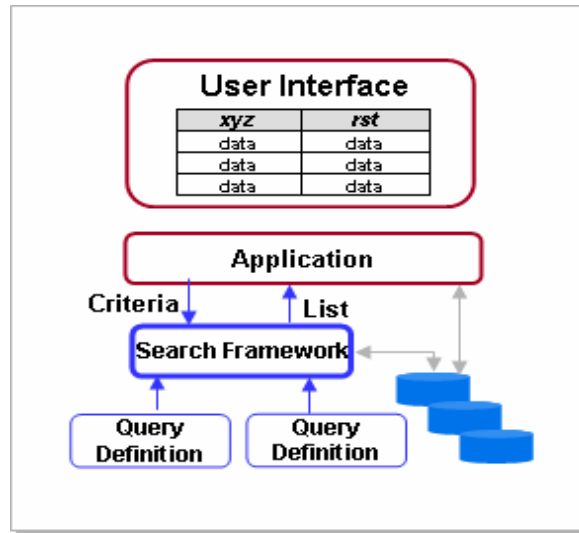
Exception handling: mechanism to manage and communicate exceptions. This is similar to logging services in that the team should leverage standard application server capabilities. The team needs to decide what mechanism to use and communicate it to the entire development community within the enterprise. No specific code needs to be developed for this service but it would be useful if the team provided examples of handling exceptions.

Deployment/Application configuration: document that details configuration. This involves standardizing the mechanism of building and deploying an application in every environment, including development, QA, UAT, staging, and production.



Monitoring: standardization and documentation of monitoring procedures. Since operations staff must monitor platforms and applications and proactively resolve issues, most departments within IT have already identified and deployed a monitoring tool. The challenge is to standardize and integrate their monitoring procedures and technologies to ensure consistent data that encompasses all systems. An enterprise may need to purchase or develop and deploy an additional specialized monitoring tool.

Search framework: shared functionality for searching data. Most portal applications need to present data in a tabular format to the users. Instead of each developer attempting to resolve this problem, a team could develop a “search framework” to be leveraged across applications and portals. The following diagram illustrates an architecture for a search framework.



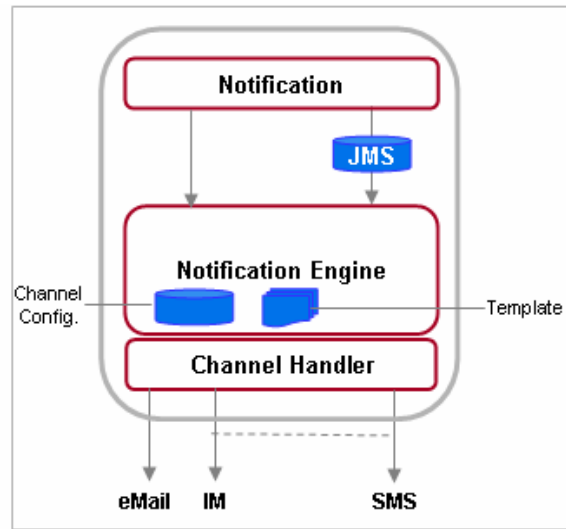
Search Framework

The search framework provides:

- Dynamic query generation based on user input
 - Sort order, joins, etc.
 - Total search results for display purposes
- Consistent mechanism for handling searches
 - Character escaping and wildcard interpretation
 - Pagination
- Abstraction of all database access code from application
 - Criteria used as input
 - Search results required in standards such as java.util.List
- Queries that reside on external files
- Utilities to handle common UI tasks
 - Pagination
 - Criteria persistence.



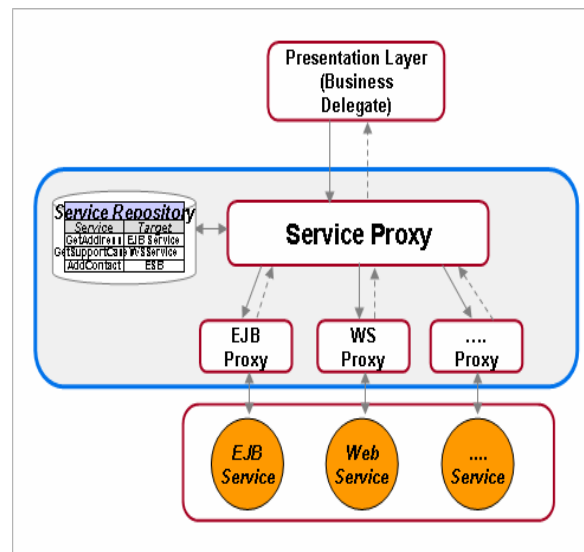
Notification framework: single notification client to all applications. This should support synchronous and asynchronous interface to the notification engine and also provide capability to send notification through multiple channels.



Notification Framework

The interface to the various channels could be developed as required by the business.


Service proxy framework: abstraction of service implementation details. Teams can deploy services either locally or remotely without having to program the calling application with implementation details or location of the service. Instead, the service locator determines the location of the service and calls it in the appropriate fashion. This supports multiple proxies such as EJB, web services, and service bus proxies; additional proxy types can be developed as required. This could also be leveraged by the business delegate to separate the presentation layer from the service layer.



Service Proxy

Security framework: enterprise-wide layer that provides security capabilities. Most application project teams develop their own security layer because current enterprise security solutions do not meet all their business needs. A security framework that supports the client side can reduce, if not





eliminate, the need to develop custom security code for each application. Following are some of the functions a security framework should provide:

- Single sign-on (SSO): capability to log in once and be able to traverse from application to application without having to login again
- Access Control: a set of security features that addresses three main areas:
 - Authentication: determining the identity of the user interacting with the applications
 - Authorization: determining if a user is allowed to perform a particular action
 - Auditing: tracking the actions performed by the users.

Several secondary services are also required such as registration, entitlement granting, and entitlement querying. These features should be provided as a generic framework that can be used and reused by different applications, each with slightly different needs but all having the same basic requirements, including:

- Identity management: Having multiple stores for managing the access control information for a set of applications or services may result in severe management problems. Identity management helps by centralizing the access control management capability, as well as provisioning the users across the enterprise.
- Consolidated user profile: Portals provide this capability to enable the application to extend the base profile. This capability extracts the user profile from multiple data sources, such as the base profile and the application-specific profile from the application-specific repository.
- Registration, delegated administration, provisioning, repository: These are security extensions built on top of access control to meet the application-specific business needs. Alternately, these could be packaged solutions integrated with access control.

Portal products provide most of these capabilities off-the-self. IT organizations will have to develop and support this capability if they do not own a portal for developing custom applications.

Portal Services


Portal services manage the presentation tier of the application. As the presentation is generally based on entitlements, there is a need to support this capability.

Presentation: the portal presentation capability provides the skins, templates, skeletons, and style sheets for each of the application teams. This should also include some sample applications to help jumpstart development and leverage the portal navigation capability, both for vertical navigation bars as well as horizontal tabs.

Personalization: the portal provides personalization services, such as portlet layout and background template selection. This paper will address additional personalization in context of the application in the profile management section of Enterprise Security.

Authentication: all portal products provide this capability. The best practice is to externalize this service. Most enterprises have implemented global directory services (such as LDAP) within the enterprise. The custom application framework should provide an authentication interface and externalize the service.

Single sign-on (SSO): the enterprise should provide a seamless user experience by not requiring multiple logins. This framework component should not only support custom applications, it should also support packaged applications and enterprise services.





Enterprise Infrastructure Services

These are services, based on applications, that could potentially be leveraged by the entire external and internal user community. Most enterprise services are infrastructure components and some of them provide the capability for users to leverage them as an application. Following are some examples of enterprise services.

Directory service: this is the standard directory service provided enterprise wide, generally in conjunction with the e-mail service. Most enterprises implement a meta-directory for managing identity across the enterprise.

Personal information management: this is the standard e-mail, calendar, and address book functionality and includes access to this information from any channel.

Collaboration: this provides capabilities such as white board, conference calling, instant messaging, discussion forums, news groups, and workspace.

Enterprise content management system: this is the infrastructure service for driving custom applications such as knowledge management, asset or contract management, and collaboration. The recommended approach is to leverage the APIs provided by the portal or the content management provider. All the leading content management systems provide the capability to develop templates for uploading and authoring content as well as workflow for managing approval process.

Following are the best practices for implementing an enterprise content management system:

- Define the taxonomy up front, ideally creating one that is enterprise wide.
- Create a single document base or repository, enterprise wide. This may not be practical, but is a good goal.
- Publish all content to one single location in production and configure all applications to retrieve content from that location, for a reduced TCO.
- Train authors and content approvers in using the system.
- Partner closely with the content management system provider by engaging their architects for every project, especially during the design phase of the project.
- Leverage the pre-built portlets to author, review, and manage the content.
- Engage a specialized function person from either your SI or content management system (CMS) provider to map your business processes to CMS workflow.

Search service: this provides capabilities for any external or internal user to find the information they are authorized to access. There are several search solutions:

1. Key word search, which is the standard search capability that most users are accustomed to
2. Natural language search, which is generally targeted towards a non-technical or Internet savvy user who has just been introduced to technology and wants to find information by asking questions using their local language
3. Federated search, which enables search of structured and unstructured data types.

The integration of a search engine is straightforward. The search engine processes XML or HTTP requests and returns the results in the order requested.

The search engine goes hand in hand with the content management system. Following are some of the best practices for getting the most from a search engine.

- Create one taxonomy enterprise-wide for the content management system
- Define meta-tags for the content and leverage them in the portal to present content to the users based on their entitlements
- Use search engines to crawl and create multiple collections and sub-collections as required
- Leverage federated search between various business units, if required
- Leverage portal tags and entitlements to protect secure contents

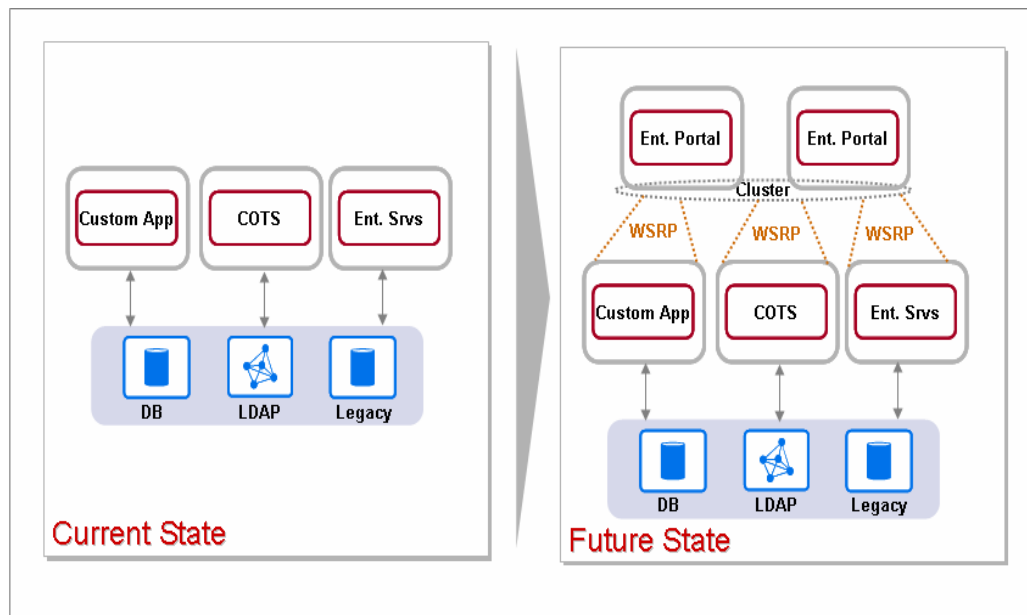


- Store secure content at the application server level.

For large sites, the time taken to crawl the entire content repository can be an issue. Depending on business needs, a company might resolve this by creating multiple collections and including all the collections in the search criteria, performing partial crawls on a periodic basis, setting up multiple search engines, and leveraging federated search. It helps to develop the architecture and the process in collaboration with the search solution provider.

Enterprise (Role-Based) Portal

By implementing the web tier components defined in this document, enterprises would achieve the “current state” as illustrated in the diagram below.



Enterprise (Role Based) Portal

In this state, IT organizations can rapidly deploy business solutions in the form of customer applications, packaged applications, enterprise services, or a combination of these components. The custom application framework enables business to provide an exceptionally good user experience. However, this has the following drawbacks:

- Re-branding the user experience would potentially require changes to all the sites.
- Users still need to know the URLs for each of the sites. By adopting some best practices, this can be reduced but not eliminated.
- This model is likely to result in redundant hardware and software for each of the point solutions. This is because each of the business units would like to schedule their own maintenance windows and the only way to facilitate this is to have dedicated infrastructure for each of the point solutions.



The target state is to leverage the concept of federated portals to create an enterprise-wide role-based portal. The advantages of this approach are as follows:

- Single point of entry for all employees, customers, partners, and other users
- Application (Portlets) access based on the role of the user
- Consolidation of hardware and software infrastructure
- Always-on capability
- Simpler re-branding of sites
- Multi-channel delivery provided by the federated portal by leveraging services.

Copyright

Copyright © 2007.

The author grants a non-exclusive licence to everyone to publish this document in full or part by acknowledging the source. Any other usage is prohibited without the express permission of the authors.

