

Content integration

Opening up the corporate application

Web technologies are enabling organisations to make information held on external systems available to employees and trading partners beyond the corporate network.

MAKING INFORMATION

available to parties external to the organisation may be a chore, but it is also a valuable business opportunity. After all, so-called 'offline' processes tend to be error-prone and can frequently be expensive. Fortunately, by externalising applications using web technologies, companies can remove the majority of system integration barriers that prevent employees, suppliers, and customers beyond the corporate network from accessing the information they need to carry out vital tasks.

There are two reasons why applications bearing web-based interfaces are more straightforward to roll out to users across multiple organisations than traditional desktop-based interfaces. First, the ubiquity of the web browser removes the need for client software installation and maintenance. Second, because most organisational networks are open to HTTP and secure HTTP traffic from the outside world, there is no need for network configuration, such as firewall rule changes, to be carried out on a per-application basis (see figure one).

By Nigel Atkinson

The Process

The process of web-enabling an application may be broken down into three stages: System design, substantiation of the design and implementation of the system (see figure two).

Design

The initial stage of the web-enabling of an application is common to all software system design processes. The goals are to define the requirements of the web based system, to produce a system design.

Business requirements should be gathered through discussion with the system owners and the intended users of the system. Use cases are helpful as a tool in this phase to ensure that the needs of all users are met and to map key processes. Usability requirements are also defined at this point and must take into account the intended system audience. These should also be informed by factors such as target web-browser platforms

and accessibility requirements. Design, of course, should not be browser-specific.

Once the requirements for the proposed system are defined, the next phase is to perform a systems analysis. The purpose of this analysis is to identify all systems that will be affected by the application externalisation. These will include the application itself, existing website functionality and management systems as well as supporting infrastructure, such as networking and access control systems. The analysis should detail the effects on each system identified and their likely impact. Key areas for investigation include security, performance and licensing.

The outputs from the two previous phases provide the information required to produce a system design. The system design details all of the software development, integration work and infrastructure changes necessary to web-enable the application, in sufficient detail to accurately estimate the duration and cost of implementation.

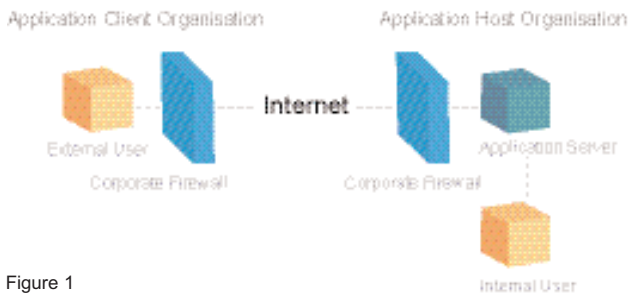


Figure 1



Figure 2

Substantiate

The purpose of the next stage of the process is to ensure the feasibility and ultimate success of the project, or at least to understand and quantify the risks so that the implementation can be planned to mitigate them.

The first phase of this stage of the process is to perform a feasibility study. This may be performed prior to the creation of a full system design; however, it will be far more accurate when performed in light of the full design. The feasibility study should examine the technical feasibility of the solution proposed in the design and should also include a cost-benefit analysis based on the business benefits that the solution is expected to deliver and the cost projection derived from the design.

Secure in the knowledge that the business benefits of the project outweigh the cost of implementation, the next phase is to identify and quantify risks. Because of

the level of integration with existing systems this type of project must be analysed both in terms of risks to the project itself, but also in terms of risks to existing systems and infrastructure. Key areas of potential risk to existing

systems are security, system stability and performance.

Implement

The last phase before the implementation stage is to create a system-implementation plan. This should identify and detail each task that must be performed and set key delivery milestones.

The implementation plan can then be used to schedule resources and plan for interruptions to applications and infrastructure. The plan should include the system implementation, testing and roll out phases that make up the final stage of the project. Managers of complex or high-risk projects may wish to update the project-risk analysis in light of the implementation plan.

Objectives

Broadly speaking, an application externalisation has one of these two objectives:

1. To publish application data for use outside of the organisation;
2. To enable external users to interact with existing internal applications.

While they share many similarities, both of these types of project require a slightly different technical approach and it is important when embarking on a web-enablement project to understand these key technical considerations.

Publishing application data

Publishing application data externally is the most straightforward type of web-enablement project. This is because the flow of data is unidirectional and no application data is updated via the external interface. Typical applications include publishing of financial data, such as customer price lists, and reporting of real-time status information.

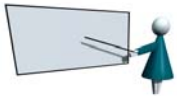
In the simplest example of a direct publishing model, there are three steps to the externalisation process: The first step is to make the data available from within the application, the second is to transform the data into information in a form that can be presented to the audience and the final step is to publish the formatted information (see figure three).

The mechanics of making application data available will be largely dictated by the technical characteristics of the application itself, software platform, data storage system and application architecture. It is, however, important to select an integration point that is appropriate to the business requirements early in the design process. There are three integration options: data-level integration, application programming interface (API) level integration, and service-oriented architecture (SOA) integration (see figure four).

Data-level integrations operate directly on the application's data



Figure 3



store and are often implemented as a software system independent of the application software itself. Integration at this level may offer faster implementation times or performance advantages over integration that operates at the API level. When selecting this approach it is important to understand the implications of doing so as there are several potential drawbacks.

API-level integrations are implemented using interfaces built into the application that are specifically designed for exposing data or functionality for use in external systems. API-level integrations offer the advantages of access to application state and application calculated values, as well as lower maintenance overheads and protection against future changes to the application data schema. Applications often provide more than one integration API at different conceptual levels and it is important to select the most appropriate API layer for use in the implementation.

SOA is a relatively new concept in application integration, although it is often possible to modify existing applications for use in a SOA context. In a service-oriented architecture, an application provides one or more services, similar to traditional public APIs, which are exposed to applications that consume the services. The fundamental difference between

SOA and standard application architectures is that the services are exposed in a standard, platform-neutral form, such as SOAP (the simple object access protocol) and WSDL (web service definition language), enabling applications developed using different technologies and deployed on different platforms to interoperate. The services ultimately rely on traditional APIs exposed by the application for their functionality, so appropriate APIs must be available before services may be exposed.

The complexity of data transformation required will vary depending upon factors such as data processing requirements and the ultimate information presentation formats. In the simplest case, where the application exposes data in the required form, there is no transformation required. In a more complex case the data transformation process may take data from multiple applications, process the data into the required information and transform the information into multiple formats for publishing.

The publishing process may simply involve the transfer of transformed data files to a web server or it may have sub-processes for tasks such as content editing, approval and scheduling. There are many reasons why it may not be desirable to publish transformed data directly. These include regulatory compliance issues, information error checking and editorial control. Complex editorial and publishing requirements are best managed through integration with a content management or web-publishing system that provides support for this kind of workflow. Editorial and approval workflows are complex to design and implement, so leveraging established processes is advisable.

External user interaction

The design of external interfaces to an application should be driven by specific use cases and supported by workflow and notification mechanisms to support users both internally and externally. Wherever possible, the external application interfaces should replicate the offline processes such that interaction with the application becomes a direct replacement for the offline process.

Web-enabling an application for external user interaction follows a similar integration process to that of enabling an application to publish data to the web, but with the additional complexities of managing bi-directional data flow between users and application. The application needs to be able to handle concurrent data requests and updates from multiple web and internal users. Because of the increased complexity in implementations of this type the integration point will usually be service-based or make use of an API layer designed for the purpose. This is because the technical issues of managing concurrent, transactional updates to application data are best managed by the application itself. Data-level integration causes data updates to occur outside of the application context and introduces many potential data integrity and consistency issues. A suitable API or service interface may not already exist, in which case it will be necessary to create one; this is especially true of older 'legacy' systems.

Usability is a crucial consideration when designing this type of system externalisation, and a significant proportion of the development effort involved will be spent implementing user-data validation and error reporting. A well-designed application will validate data prior to storing it in order to maintain integrity and consistency



Figure 4: Application integration points



Figure 5: User input handling



Figure 6

within its data store. While this helps ensure the security of application data it does not offer much help to the user. When creating a new web-based user interface for an application it is crucial to ensure that data entry errors are identified and clearly communicated to the user. One of the greatest challenges in an application with multiple user interfaces is to perform data validation and provide feedback to the user without replicating validation rules across multiple systems (see figure five).

CMS integration

A content management system (CMS) can provide a good platform for both types of application externalisation discussed in this article. When publishing application data to the web a CMS may provide editorial workflow capabilities and manage approval and publishing of information generated by the application. Approved information may also be made available through the CMS for use by content authors and editors.

There are two web-publishing strategies available and most CMS products follow one or the other. When selecting a CMS it is important to understand the difference between them and to understand the implications in terms of performance and security.

In the first strategy, the CMS and the website are combined in a single application. Both the CMS and the website run in the same physical environment and share the same data store. This offers a number of advantages, including quick publishing as there is usually no need to transfer data to the website environment and it usually requires less infrastructure than the alternative.

The disadvantages of this type of CMS are that it must be deployed outside of the corporate network in order to make the website available to external users. This means that if the website is compromised then the CMS and all of the information stored within it will be, too.

The second publishing strategy maintains the CMS and the website independently of each other. They are often in geographically disparate locations and do not share a data store. This strategy has the advantage that the CMS may be installed within the corporate network boundary and data transferred to the website only at publish time. Additionally, should the website become compromised then there is no access to the CMS itself or the information stored within it. Another inherent advantage of this strategy is that CMS maintenance and upgrades may be performed without

affecting the live websites and applications (see figure six).

When designing a new web-based user interface for an application of any complexity, the use of a CMS product as the development platform should be considered. Where applicable a web-publishing platform may also be used to manage the application-implementation process. Enterprise CMS products often provide an environment for the rapid development of user interfaces and supporting capabilities, built from libraries of re-usable components. This can both help to reduce development times and improve the stability, security and maintainability of the user interface and supporting data validation and error-reporting functionality. Version control and approval workflow features available in many CMS products provide support for multiple simultaneous development, testing and live versions of content and functionality.

Developing web-based applications using a CMS solution provides a centralised management infrastructure for all web properties within the organisation. This has many advantages to the organisation as a whole, including reduced IT management costs, a consistent management interface across all web based systems and cost savings during the development process. Selecting the right CMS platform is an important strategic decision, as most platforms provide support for one application development platform and one publishing strategy. ■

Nigel Atkinson is a director of NeoWorks, a software development and consultancy company that has performed website content management and application externalisation projects for clients such as Channel 5 Broadcasting, Hadlow College and Levi's Europe. He can be contacted at nigel@neoworks.com