CHAPTER

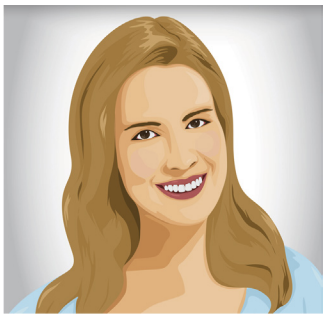
MANAGING WEB-BASED APPLICATIONS

7

First we thought the PC was a calculator. Then we found out how to turn numbers into letters with ASCII—and we thought it was a typewriter. Then we discovered graphics, and we thought it was a television. With the World Wide Web, we've realized it's a brochure.
Douglas Adams

People tend to think of the web as a way to get information or perhaps as a place to carry out ecommerce. But really, the web is about accessing applications. Think of each website as an application, and every single click, every single interaction with that site, is an opportunity to be on the very latest version of that application.
Marc Andreessen

A Day in the Life of Acme Manufacturing: Meeting Customer Needs.



Meet Cassie. She is an executive relationship manager at Acme Manufacturing. Cassie's job is to sell Acme's products to a dozen or so very large customers, address their continuing needs, and maintain a good relationship with each customer so they can be used as references for other potential customers. To date, Acme has not had a web-based application that will enable Cassie and her counterparts at Acme to efficiently and effectively monitor customer interactions with Acme product development and support personnel. But as of today, things have changed. A team of Acme IT developers just released a sophisticated web-based application to make Cassie's life much easier and hopefully increase Acme's customer satisfaction ratings.

Application Performance Management (APM) in the Digital Enterprise. http://dx.doi.org/10.1016/B978-0-12-804018-8.00007-3

Using the web-based customer relationship management application, customers can now use their desktop or mobile devices to send any issues or concerns to product development and managed services and receive a quick response consistent with the service-level agreement (SLA) with Acme. Cassie will be kept in the loop so that she can monitor, control, and resolve any unanswered requests. The new application had only been in service a few days when Cassie really saw its value.

Meet Henry. Henry is the CEO of XYZ, Inc. Henry is a "problem customer" who has always taken up an inordinate amount of Cassie's time. Despite the fact that Henry's company, XYZ Inc., cannot afford many of Acme's premium managed services, Henry continually peppers Cassie with long drawn-out telephone queries to try to secure free demos and trial subscriptions to managed services that are clearly beyond the needs and financial reach of XYZ. He also continually harasses the Acme help desk. With the deployment of the customer relationship management application, all of Henry's interactions are systematically recorded and his pattern of unorthodox behavior is monitored. Now Cassie has the objective evidence she needs to bring the issue to the attention of her supervisor, Bryan. When Bryan steps in to put a stop to Henry's antics, Cassie is freed up to more effectively manage a greater number of Acme's more profitable customers and the help desk is grateful for her intervention.

Fast forward to the end of Acme's fiscal year, and it's clear that the web-based customer relationship application has enabled Cassie to achieve a much higher level of customer engagement overall and stay in much closer communication with Acme's product developers and managed services. On a recent survey of customer satisfaction, Cassie scored top marks for all her accounts, and levels of customer satisfaction with Acme and its products are at an all-time high.

INTRODUCTION TO WEB-BASED APPLICATIONS

Basically, a **web-based application** is a software program that is stored on a remote server and uses web technologies (e.g., Flash, Silverlight, JavaScript, HTML, CSS) and **web browsers** to deliver one or more functions for the end user over a network through a browser client. The varying levels of usefulness provided by websites have caused some debate as to what qualifies as a web application. The determining factor is whether the website provides a **service** or function. If it performs a task, no matter how insignificant, it is a web application and should be managed as such. A web application can be as simple as Google's search engine, the basic concept of which is to act as a phone directory to search names, locations, and numbers, or as complex as a word processor that enables users to store information and download the document to their personal hard drive. For example, the first mainstream web **applications** were relatively simple, but by the late 1990s there was a push toward more complex web applications, such as TurboTax, that are used by millions of Americans to prepare and file their income tax returns on the web.

It is useful, at this point, to distinguish between web-based applications and the cloud applications discussed in Chapter 4 (Fig. 7.1).

Although similar in many ways, there are some obvious differences between web-based applications and cloud applications that impact the style in which each is managed. While cloud applications can be construed as web applications in the sense that they can be used through web browsers, not all web applications qualify as cloud applications for the following reasons:

- Web Applications
  - Almost exclusively designed to be used from a web browser.

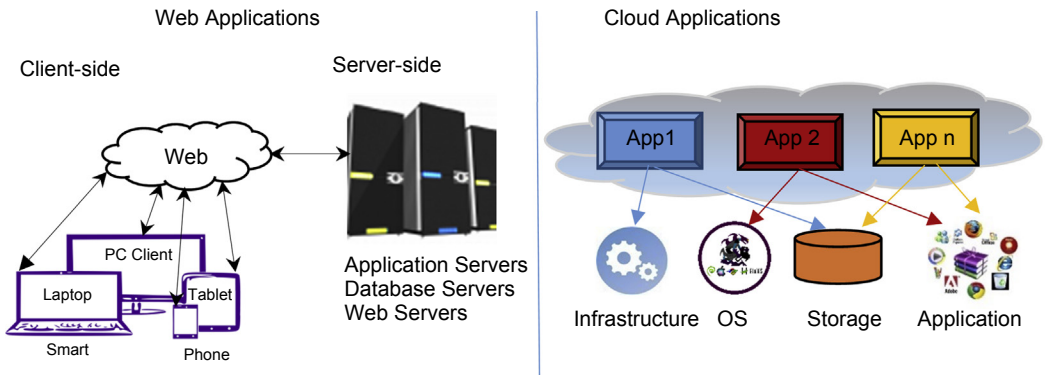


FIGURE 7.1

Comparison of web versus cloud.

  - Typically use a mix of server-side script, such as ASP or PHP, and client-side script, such as HTML5 or JavaScript. The server-side script manages how the information is stored and retrieved while the client-side script manages the presentation of the information.
  - The web browser (thin client) relies on the web server to provide its core functional **web services.**
  - Exclusively web-based with limited options for consumer customization.
  - Offer more limited functionality than **cloud** applications.
- Cloud Applications
  - Data are stored on a cloud or cloud-like infrastructure.
  - Data can be cached locally for full offline mode.
  - Support different use requirements, including data backup, data compression, and **security.**
  - Can be used from a web browser and/or customer-built apps installed on the web-connected devices, such as desktops and **mobile devices.**
  - Can be used to access a wider range of services, such as on-demand computing, storage, and application **development** platforms.
  - Depend on cloud technology but are available offline, if users choose.
  - Offer rich functionality and customization.
  - Support **virtualization.**

Web-based applications can be developed internally by an organization and then self-hosted, cloud-hosted, or provisioned as **software as a service (SaaS)**. Internally developed and hosted applications evolve from legacy applications with added web-based front ends to reduce the technical challenges of maintaining and supporting thick clients. This approach also accommodates endpoint evolutionary trends, including **mobile applications** and **bring your own device (BYOD)** policies.

As alternatives evolve with regard to where and how web-based applications are hosted, organizations are adding SaaS and cloud-hosted applications to their mix of web-based application offerings. SaaS is a particularly viable and popular alternative for those applications that need to be accessed by a mobile workforce. Similarly, where organizations experience variable levels of use and load, hosting applications in a third-party cloud either in whole or on a dynamic, as-needed basis is gaining in popularity.

Over the years, web-based applications have become more sophisticated and powerful thanks in large part to **AJAX**, a programming model used to create more responsive web applications such as Gmail and Yahoo mail clients. As a result, present-day web applications are better able to handle the day-to-day tasks of users and the lines between desktop applications and web applications have blurred.

Most recently, many existing web-based applications were recast into mobile web applications (i.e., Internet-enabled applications), such as Facebook and Google Maps, which have specific functionality for mobile devices.

WHY WEB-BASED APPLICATIONS?

It is difficult to imagine any organization existing today without web-based applications. An obvious advantage of web-based applications is that they are accessible from anywhere via a web browser. The main selling point of web applications is reduced support costs. Web applications eliminate the need to

install software on a client, to routinely upgrade the software, and to maintain the client's **operating system**. To further reduce support costs, a web application can run on a PC or a Mac and on multiple browsers such Internet Explorer, Chrome, or Firefox. Consequently, web applications eliminate the need for developers to build a client for a specific type of computer or a specific operating system.

MANAGING WEB-BASED APPLICATIONS

Unlike **native applications** that are downloaded and installed on a specific device, web-based applications require a very different **management** approach to their development, accessibility, efficiency, and revenue-generation capabilities. The following sections discuss the different elements of **application lifecycle management** that specifically relate to the management of web-based applications.

HIRE CROSS-FUNCTIONAL PERSONNEL

The growing need for information technology (IT) to deliver web services, along with deploying and support infrastructure, has resulted in the need for processes and skills to be much more cross-functional. Traditionally, applications were the domain of the developers; however, as organizations move to web-based applications the line between **development and operations** blurs. Consequently, staffing, skills, and product requirements have changed drastically, resulting in a rise in IT support costs and a growing interest in best practices, such as the **IT Infrastructure Library (ITIL)** framework and the emergence of centers of excellence in large organizations.

To effectively manage this complex environment of web-based applications, it is also important to hire IT personnel with higher level cross-functional skills, such as knowing which server supports which application(s) or which database begins to underperform when it hits a given number of connections.

CAPITALIZE ON DATABASE DESIGN AND QUERY OPTIMIZATION

The database is the most important component of a web-based application. It is essential that web developers are familiar with these critical aspects in a web-based application. A high level of SQL knowledge is essential to optimizing and rewriting queries to save milliseconds on response time. Basic SQL queries can be problematic in the web-based production environment. Their fill-in-the-blank nature can cause problems since it will be the part of the web-based application that developers are least familiar with. Instead, they should be replaced with custom-created SQL queries that focus on the needs of the database being used by a specific web-based application. Another necessary skill is that of formulating contingencies about how the app can grow and the associated database problems that growth will create.

THINK LIKE A SERVER

Web-based applications are dependent on servers that can interpret code differently from the way it was written. When managing web-based applications, it is important to establish acceptable usage patterns, identify exceptions that may result in problems, and track and monitor all application activities to identify the way in which different resources are used.

CONSIDER THE SOURCE

Web-based applications are typically used through web browsers. Each browser is slightly different and displays web pages in different ways. As a result, programming of web-based applications needs to be specialized to accommodate the ways that browsers interact with different web languages, such as HTML, **XML**, Flash, Perl, ASP, and PHP. This variability in browsers, along with the likelihood of their use by potential users, must be considered when designing web-based applications. To address this issue and maximize exposure for a consumer-based application, open source code can be used to develop a web-based application.

SIMPLIFY DEVELOPMENT WITH WEB APPLICATION PROGRAMMING INTERFACES

Web **application programming interfaces (APIs)** are quickly becoming another critical web **application management** technique to assist developers in efficiently creating web applications. As public and private organizations are pressured to deliver products and services to more consumers as inexpensively and conveniently than ever before, organizations have no choice but to adapt or risk being left behind in the era of web APIs. APIs provide a framework that developers can use to quickly build web services or an application from multiple services to leverage, advertise, and combine corporate assets for widespread consumption. Using traditional **service-oriented architecture (SOA)**, organizations can create web services from diverse data sources, including databases and legacy systems, to extend strategic organizational assets such as product catalogs, phone listings, and order status data beyond conventional boundaries. **XML** or **JSON** formats provide internal and external developers with the ability to easily create readily accessible Web 2.0 applications. To create web applications that support a wide range of industry and **line of business** functions that lead to an increased share in new and existing markets, organizations need to develop a carefully constructed web API strategy. To achieve this, organizations should complete the following steps:

1. Develop a clear understanding of the underlying organizational objectives for creating web applications, such as:
   a. Rebrand the organization
   b. Educate customers about a new offering
   c. Explore new market channels to increase revenue
2. Choose which assets to expose and to whom
3. Determine Web API availability—internal developers only, or offer it for public consumption
4. Tap best developers to support organizational objectives through effective partnering
5. Understand legal implications of exposing organizational assets through a web API
6. Establish metrics to measure API success, such as tracking number of page visits or revenue generated

The world's largest API repository is the Programmable Web's API Directory (http://www.programmableweb.com/apis/directory), which currently lists approximately 15,000 public APIs that developers can use to build new products and business workflows. The repository is searchable by category and/or protocol/format.

DETERMINE BEST LEVEL OF TESTING

Getting a web-based application into use is sometimes far more important than spending an inordinate amount of time on testing. The goal is to test the application as much as possible without delaying its release. Testing can be accomplished by peer review, unit testing, ad-hoc testing, load testing, or **quality assurance** processes such as weekly reviews of error logs. Many development mistakes can be avoided by using a combination of these testing techniques. The more obscure mistakes can be left to the application users to report in real-time feedback. Take advantage of this commonly accepted, iterative process that emerged in the online software culture to manage the amount of testing that is performed in web-based application development.

PERFORM REAL-TIME MONITORING

In today's rapidly changing business landscape, IT managers and application developers are being called on to deliver web applications and web services without delay and maximize their end-user expectations by adeptly navigating the increased complexity and potential points of failure of their web applications. This requires organizations to perform **real-time monitoring** of the **user experience**, volume of traffic, and health and **performance** of web applications, including slow database queries. Fortunately, a number of vendors now offer tools (e.g., BMC's TrueSight App Visibility Manager) to accomplish these daunting tasks.

Another important aspect of a web-based application is its response time. If the application is well designed, elegantly coded, and technically correct but not fast enough, users will reject it. Solutions for these problems should include installing an accelerator on the servers and incorporating load testing to benchmark response times.

MEASURE APPLICATION PERFORMANCE

As web-based applications become an increasingly important component of conducting business, their performance becomes more tightly coupled with an organization's bottom line and ultimately its mission. Consequently, managing **application performance** is particularly essential in a web-based environment. However, the complexity of diverse, web-based application capabilities requires more than organizational knowledge and manual processes to ensure business continuity and organizational productivity.

For example, application performance differences of fractions of a second in online financial transactions can translate into losses of millions of dollars per hour in a large organization. Similarly, in the multibillion dollar online retail industry, a customer's perception of brand quality can adversely be affected by slow response times or a less than seamless online shopping experience.

Robust web-based **application performance management (APM)** is critical when delivering high-quality business services. Inefficiencies and points of failure in the communication, server, or data layers can lead to subpar performance or failure. Unfortunately, since these exhaustion points often are not discovered until user expectations are not met, **service-level agreement (SLA)** thresholds are missed and/or business is lost. Proactive monitoring of availability and performance is beneficial in these situations. This combination ensures that organizations consistently meet or exceed user

expectations and enables companies to reap benefits of best practice implementation to successfully deliver against SLAs, lower overall resource costs, improve infrastructure utilization, and satisfying demanding customers.

APM tools such as New Relic, AppDynamics, Foglight, and BMC Software APM are particularly useful in addressing these complexities and mitigating risk to optimize costs, customer satisfaction, and retention, and in achieving a higher probability of meeting or exceeding service levels. In addition, the implementation of APMs frees up skilled personnel from lower level day-to-day support tasks to concentrate on developing new web-based applications to support higher level business goals and objectives.

CHALLENGES OF MANAGING WEB-BASED APPLICATIONS

While web-based applications offer higher levels of cost efficiency and flexibility, they also introduce new and unique visibility and control challenges.

REQUIRE A BILATERAL MANAGEMENT APPROACH

To meet performance and user experience expectations in addition to basic availability, IT teams must adopt a bilateral approach that combines performance visibility with proactive optimization technologies.

ABSENCE OF SOFTWARE DEVELOPMENT KITS

Another challenge is that while native applications have their own unique development process, web applications do not and **software development kits (SDKs)** that exist to provide developers with a suite of tools to efficiently develop native applications are not available to assist in the development of web-based applications.

API RELIABILITY

APIs, as discussed, are a mixed blessing when managing web-based applications. While APIs reduce development time and cost by allowing applications to share information, such as in Yelp where nearby restaurants can be displayed on a Google Map within Yelp, they have their drawbacks. For example, APIs can unexpectedly become unavailable when the companies that create them shut down their API services, restrict their accessibility, or go out of business, causing web applications to lose the services they depend on from the API.

SECURITY

Despite their many advantages, web-based applications raise a number of security concerns stemming from improper coding. A significant amount of confidential information is stolen every day from vulnerable web application servers that are publicly accessible and tied into backend database servers that store a wealth of corporate information. For example, an attacker can input an SQL query into the search field of a web form that can then be accepted by the web-based application and passed to the back-end database where read/write access is granted from the application to the database server. Then, the query is

executed. This allows the attacker to view and/or delete the contents of the database. Another common security concern in web-based applications is **cross-site scripting (XSS)**, a common technique used by hackers to maliciously inject code into a legitimate web-based application to deceive users and redirect them toward phishing sites. Attackers do not directly target a user but instead exploit a vulnerability in a website or web-based application that the user visits to deliver a malicious script to the user's browser. A common example of this is when an attacker obtains the user's session cookie to impersonate that user.

The **Open Web Application Security Project (OWASP)** is a 501(c)(3) worldwide not-for-profit charitable organization dedicated to enabling organizations develop, purchase, and maintain applications that can be trusted. In its Top 10 project, OWASP raises awareness about web application security by identifying some of the most critical risks facing organizations (Table 7.1).

Table 7.1 OWASP Top 10 Web Application Security Issues

No.	Security Risk	Cause	Consequence
1	Injection	SQL, OS, and LDAP injection can occur when untrusted data is sent to an interpreter as part of a command or query.	Attackers' hostile data can trick the interpreter into executing unintended commands or accessing unauthorized data.
2	Cross-site scripting (XSS)	Application takes untrusted data and sends it to a web browser without proper validation.	Attacker can execute scripts in user's browser to hijack user session, deface website, or redirect user to phishing sites.
3	Broken authentication and session management	Functions related to authentication and session management are improperly implemented.	Attackers can compromise keys or session tokens, and exploit other implementation flaws to assume other users' identities.
4	Insecure direct object references	Developer exposes reference to an internal implementation file directory or database key.	Attackers can manipulate references to access unauthorized data.
5	Cross-site request forgery (CSRF)	Forces logged-on user's browser to send forged **HTTP** request, including session cookie and other automatically included authentication information.	Attacker forces user's browser to generate requests that vulnerable application recognizes as legitimate user requests.
6	Security misconfiguration	Many **configuration** settings are not shipped with secure defaults and settings are undefined.	Software, including all code libraries used by applications, are not kept up to date.
7	Insecure cryptographic storage	Sensitive data such as credit cards, SSNs, and authentication credentials are not properly encrypted or hashed.	Attackers may steal or modify weakly protected data to conduct identity theft, credit card fraud, and other crimes.
8	Failure to restrict URL access	Access-controlled checks are not performed each time protected links and buttons are accessed.	Attackers can forge URLs to access hidden pages.
9	Insufficient transport layer protection	Confidentiality and integrity of sensitive network traffic is not protected through authentication or **encryption**.	Applications support weak algorithms, use expired or invalid certifications, or use them incorrectly.
10	Invalidated redirects and forwards	Users are redirected or forwarded to other pages and websites. Untrusted data is used to determination the destination pages.	Attackers can redirect users to phishing or malware sites or use forwards to access unauthorized pages.

Adapted from OWASP (www.owasp.org/index.php/Top_10_2013-Top_10).

However, it is not sufficient to just be aware of potential risks or concentrate on preventative measures; it is also imperative that organizations understand how and why these security breaches occur. To guard against potential security breaches, organizations should conduct a thorough audit of their web applications on a regular basis, understand how web application attacks work, and consistently and faithfully apply the following principles:

- Appoint a visible executive advocate or management sponsor to encourage and demonstrate leadership backing to promote **collaboration** between business leaders, IT leaders, development, operations, and security teams.
- Integrate application security throughout the application development process and enforce key milestones.
- Raise awareness of sources of potential security flaws in developer and server administrator training.
- Develop a threat model so developers can anticipate authentication, encryption, data storage, and system **integration** security issues and design a secure application from the start.
- Employ manual static (**SAST**) and automated dynamic (**DAST**) application security testing tools to respectively spot check and repeatedly test the application's behavior. Cloud-based DAST tools are particularly useful because they can test large numbers of application in a short period of time without incurring the cost of setting up on-premises software.
- Use **web application firewalls (WAFs)** to block certain web application attacks like cross-site scripting. WAFs can also limit access from undesirable or suspicious networks, alter the way in which browsers and applications communicate, and protect an organization against vulnerabilities in third-party applications while vendors are working on fixes.
- Conduct a thorough audit of all web applications on a regular basis.

SUMMARY

Web-based applications have increasingly become a necessary part of conducting business. They enable organizations to significantly expand the geographical reach of their customer base and increase their market share. IT requirements for web-based applications are fewer than for native applications, and system-wide updates and patches are handled by the provider and implemented instantly. Add to this the **interoperability** and ease of use of web-based applications and it is clear to see that they are a powerful business tool.

Managing web-based applications requires a different approach to their development, accessibility, efficiency, and revenue-generation capabilities. The many facets of web application management that must be addressed span problem definitions, from operations and maintenance, with a particular emphasis on security in every phase of the system development lifecycle.

KEY TAKEAWAYS

- Organizations cannot effectively conduct business without one or more web-based applications.
- Web applications require a different application lifecycle management approach from the one required by native applications.

- Greater cross-functional skills are required to effectively develop, operate, and manage web-based applications.
- Website safety and security are crucial to establishing positive public perception and maintaining an organization's image, as well as managing potential financial and legal risks.
- The Open Web Application Security Project (OWASP) is a nonprofit organization dedicated to supporting and protecting organizations from web-based application security breaches.

Examples of vendors with products in this space:

Apica
AppDynamics
AppFirst
Appnomic
ASG
Aternity
BlazeMeter
BMC
CA Technologies
Dell Quest
Dyn
Dynatrace
EMC
ExtraHop
HP
IBM
Idera
ManageEngine
Nastel
NetScout
Netuitive
New Relic
Push Technology
Riverbed
SmartBear
SOASTA
Splunk
Stackify
Sumo Logic