УДК: 004.37

INSTALLING THE WEB APPLICATION CREATION ENVIRONMENT

Qiryigitov B.A.[1]
Qiryigitov Baxriddin Abdusattarovich.[1] *Teacher of department of chemistry and phyisics,*
Andijan Branch of Tashkent State Agrarian University

Andijan. Uzbekistan

Annotation: In this article, you need to master the basic structural tags and use the HTML language to create a web page layout. Exploring the installation environment for creating web applications on a personal computer. In computing, a web application or web app is a client–server computer program in which the client (including the user interface and client-side logic) runs in a web browser.

Key words: web application, computer program, web sites, webmail, online, function.

Common web applications include webmail, online retail sales, online auctions, wikis, instant messaging services and many other functions.

The general distinction between a dynamic web page of any kind and a “web application” is unclear. Web sites most likely to be referred to as “web applications” are those which have similar functionality to a desktop software application, or to a mobile app. HTML5 introduced explicit language support for making applications that are loaded as web pages, but can store data locally and continue to function while offline.

Single-page applications are more application-like because they reject the more typical web paradigm of moving between distinct pages with different URLs. Single-page frameworks like Sencha Touch and AngularJS might be used to speed development of such a web app for a mobile platform.

In earlier computing models like client–server, the processing load for the application was shared between code on the server and code installed on each client locally. In other words, an application had its own pre-compiled client program which served as its user interface and had to be separately installed on each user's personal computer. An upgrade to the server-side code of the

application would typically also require an upgrade to the client-side code installed on each user workstation, adding to the support cost and decreasing productivity. In addition, both the client and server components of the application were usually tightly bound to a particular computer architecture and operating system and porting them to others was often prohibitively expensive for all but the largest applications.

In contrast, web applications use web documents written in a standard format such as HTML and JavaScript, which are supported by a variety of web browsers. Web applications can be considered as a specific variant of client–server software where the client software is downloaded to the client machine when visiting the relevant web page, using standard procedures such as HTTP. Client web software updates may happen each time the web page is visited. During the session, the web browser interprets and displays the pages, and acts as the universal client for any web application.

In the early days of the Web, each individual web page was delivered to the client as a static document, but the sequence of pages could still provide an interactive experience, as user input was returned through web form elements embedded in the page markup. However, every significant change to the web page required a round trip back to the server to refresh the entire page.

Applications are usually broken into logical chunks called “tiers”, where every tier is assigned a role. Traditional applications consist only of 1 tier, which resides on the client machine, but web applications lend themselves to an n-tiered approach by nature. Though many variations are possible, the most common structure is the three-tiered application. In its most common form, the three tiers are called presentation, application and storage, in this order. The web browser sends requests to the middle tier, which services them by making queries and updates against the database and generates a user interface.

For more complex applications, a 3-tier solution may fall short, and it may be beneficial to use an n-tiered approach, where the greatest benefit is breaking the business logic, which resides on the application tier, into a more fine-grained

model. Another benefit may be adding an integration tier that separates the data tier from the rest of tiers by providing an easy-to-use interface to access the data.

There are some who view a web application as a two-tier architecture. This can be a “smart” client that performs all the work and queries a “dumb” server, or a “dumb” client that relies on a “smart” server. The client would handle the presentation tier, the server would have the database (storage tier), and the business logic (application tier) would be on one of them or on both. While this increases the scalability of the applications and separates the display and the database, it still doesn't allow for true specialization of layers, so most applications will outgrow this model.

An emerging strategy for application software companies is to provide web access to software previously distributed as local applications. Depending on the type of application, it may require the development of an entirely different browser-based interface, or merely adapting an existing application to use different presentation technology. These programs allow the user to pay a monthly or yearly fee for use of a software application without having to install it on a local hard drive. A company which follows this strategy is known as an application service provider (ASP), and ASPs are currently receiving much attention in the software industry.

Security breaches on these kinds of applications are a major concern because it can involve both enterprise information and private customer data. Protecting these assets is an important part of any web application and there are some key operational

areas that must be included in the development process. This includes processes for authentication, authorization, asset handling, input, and logging and auditing. Building security into the applications from the beginning can be more effective and less disruptive in the long run.

Cloud Computing model web applications are software as a service (SaaS). There are business applications provided as SaaS for enterprises for fixed or usage dependent fee. Other web applications are offered free of charge, often generating income from advertisements shown in web application interface.

Writing a web application is often simplified by open source software such as Django, Ruby on Rails or Symphony called web application frameworks. These frameworks facilitate rapid application development by allowing a development team to focus on the parts of their application which are unique to their goals without having to resolve common development issues such as user management. While many of these frameworks are open source, this is by no means a requirement.

The use of web application frameworks can often reduce the number of errors in a program, both by making the code simpler, and by allowing one team to concentrate on the framework while another focuses on a specified use case. In applications which are exposed to constant hacking attempts on the Internet, security-related problems can be caused by errors in the program. Frameworks can also promote the use of best practices such as GET after POST.

In addition, there is potential for the development of applications on Internet operating systems, although currently there are not many viable platforms that fit this model.

Examples of browser applications are simple office software (word processors, online spreadsheets, and presentation tools), but can also include more advanced applications such as project management, computer-aided design, video editing and point-of-sale.

Litratures

1. Inkova NA, Zaitseva EA, Kuzmina NV, Tolstykh SG The creation of Web-sites: Educational-methodical manual. Part 5. Tambov: Publishing house of Tamb. state. tech. University, 2005.
2. 4. Orlov L. V. Web-site without secrets. L.V. Orlov. - 2 nd ed. - Moscow: Buk-press, 2006.
3. 5. Polonskaya E.L. The HTML language. Self-teacher: - M .: Publishing house “Williams”, 2005.
4. 6. Creation of Web-pages and Web-sites. Textbook: ed. VN Pechnikova. - Moscow: Publishing House Triumph, 2006.
5. 7. Yakushev, L. V. We begin to work on the Internet. Quick Start Guide. - M .: Publishing house “Williams”, 2006.