LAGUNA STATE POLYTECHNIC UNIVERSITY
COLLEGE OF COMPUTER STUDIES

Web and Database Integration

ITST 303

TABLE OF CONTENTS

CHAPTER

1

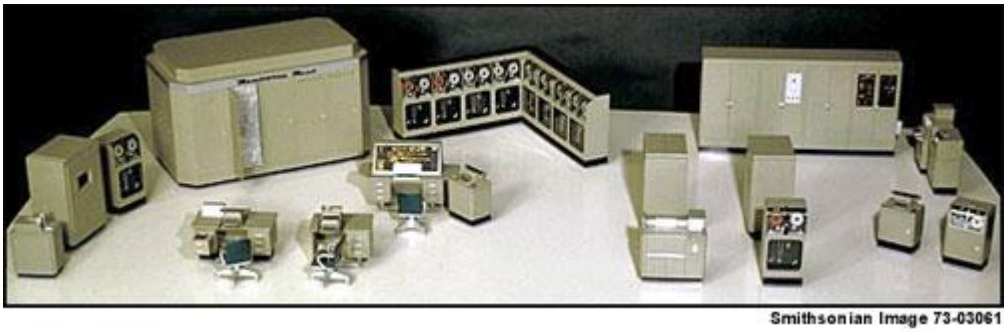
THE WEB

HISTORY OF THE INTERNET

The Internet started in the 1960s as a way for government researchers to share information. Computers in the '60s were large and immobile and in order to make use of information stored in any one computer, one had to either travel to the site of the computer or have magnetic computer tapes sent through the conventional postal system.

Another catalyst in the formation of the Internet was the heating up of the Cold War. The Soviet Union's launch of the Sputnik satellite spurred the U.S. Defense Department to consider ways information could still be disseminated even after a nuclear attack. This eventually led to the formation of the ARPANET (Advanced Research Projects Agency Network), the network that ultimately evolved into what we now know as the Internet. ARPANET was a great success but membership was limited to certain academic and research organizations who had contracts with the Defense Department. In response to this, other networks were created to provide information sharing.

January 1, 1983 is considered the official birthday of the Internet. Prior to this, the various computer networks did not have a standard way to communicate with each other. A new communications protocol was established called Transfer Control Protocol/Internetwork Protocol (TCP/IP). This allowed different kinds of computers on different networks to "talk" to each other. ARPANET and the Defense Data Network officially changed to the TCP/IP standard on January 1, 1983, hence the birth of the Internet. All networks could now be connected by a universal language.



Smithsonian Image 73-03061

The image above is a scale model of the UNIVAC I (the name stood for Universal Automatic Computer) which was delivered to the Census Bureau in 1951. It weighed some 16,000 pounds, used 5,000 vacuum tubes, and could perform about 1,000 calculations per second. It was the first American commercial computer, as well as the first computer designed for business use. (Business computers like the UNIVAC processed data more slowly than the IAS-type machines, but were designed for fast input and output.) The first few sales were to government agencies, the A.C. Nielsen Company, and the Prudential Insurance Company. The first UNIVAC for business applications was installed at the General Electric Appliance Division, to do payroll, in 1954. By 1957 Remington-Rand (which had purchased the Eckert-Mauchly Computer Corporation in 1950) had sold forty-six machines.

18 Famous Internet Milestones

In 1971, Ray Tomlinson sent the world's first email, a message passed from one computer to another placed directly beside it. Now, more than 50 years later, we send and receive approximately 306 billion email messages each and every day.

Email is just one example of the explosive growth experienced by the Internet and its features.

Since its development in the late 20th century, the Internet has quickly evolved into a necessity for our daily lives. Not long ago, Internet access was considered a luxury, and it certainly wasn't fast. In our graphical representation of the Internet, we'll explore a series of famous Internet milestones you may not have known about, including the first spam email, the first item purchased securely online, and the creation of cryptocurrency.

1. The birth of the Internet

There's not really one main date you can pin down as the creation of the Internet. Some sources say the Internet started in 1983 with the establishment of a communication protocol known as Transfer Control Protocol/Internetwork Protocol (TCP/IP). This protocol made it possible for computers on different networks to communicate.

Other sources trace the Internet back to the 1960s with the development of ARPANET. ARPA was the U.S.

Defense Department's Advanced Research Projects Agency, and the purpose of their network was to connect different computers so information could pass between them.

2. The first email

You can thank engineer Ray Tomlinson for your email inbox. In 1971, Tomlinson figured out how to send a message from one computer to another, leading to email as we know it today.

In interviews, he shares that the contents of the first email were probably a random selection of letters and does not recall exactly what he sent from one computer to another.

Today, Statista estimates that we send and receive 306 billion emails around the world each day. We also have around four billion email users around the world.

Those numbers will likely continue to grow.

3. The first spam email

Around 400 people received the first spam email in 1978. Now, around 89 billion spam emails are sent daily. The good news, however, is that the number of spam emails is decreasing.

Gary Thuerk, the "father of spam," sent out the first unsolicited email to an audience of about 400 on ARPANET in 1978. The message was meant to promote Digital Equipment Company's (DEC's) new product demo on the west coast of the United States.

As ARPANET was, at the time, a channel with limited usability and serious restrictions, this resulted in some backlash.

4. The first registered domain name

The first registered domain name ever was symbolics.com in March of 1985. It was purchased by the Symbolics Computer Corporation, a computer development company. They retained the domain in 2009, until it was sold to an investment group that continues to preserve its legacy online.

As of Q4 2021, Verisign reports a total of 341.7 million registered domain names.

The number of registered domains increased by 1.6 million — or 0.5% — from Q4 2020. The top-level domains (TLDs) .com and .net saw am 8.2 million — or 5.0% — increase, reaching 173.4 million registrations.

5. Invention of the World Wide Web

Tim Berners-Lee changed the course of Internet history in 1989 when he invented the World Wide Web.

Berners-Lee created the World Wide Web while working at CERN (the European Council for Nuclear Research) to help scientists and universities share information all over the world on devices that use the Internet.

As written on the first website, the World Wide Web is a "wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents." Basically, the web uses a bunch of interconnected links on the Internet to help you access websites, videos, and anything else you need online.

6. The first website

Not long after Tim Berners-Lee proposed the World Wide Web, we also got the first website.

In 1991, a website explaining CERN's World Wide Web project launched.

Fun fact: You can still view the website to this day (or at least at the time of making this video). It's definitely not as complex as the websites you're used to know, but it was revolutionary when it was created.

Today, you can find about 1.7 billion websites on the Internet, with more than 500,000 new websites launched each day.

7. The first photo on the Internet

You're going to hear the name Tim Berners-Lee again. The man behind the World Wide Web is also responsible for uploading the first photo to the Internet in the early 1990s.

The image features a group of singers associated with CERN, known as Les Horribles Cernettes, photoshopped onto a blue background.

Berners-Lee got the photo from his colleague Silvano de Gennaro, who had no idea that his image would be a piece of history.

8. The invention of Wi-Fi

What's the Wi-Fi password?

Wi-Fi uses an Internet-connected transmitter to send radio signals to different devices, which then allows you to stream movies, share memes, and spend your paycheck on things you don't need.

Many smaller milestones led to Wi-Fi's popularity today. In fact, a very early contributor to the technology that makes Wi-Fi possible was actress Hedy Lamarr, who worked with fellow creative George Antheil to develop a "frequency hopping" system. This system allowed transmitters and receivers to move to new frequencies in a way that prevented others from intercepting radio waves.

That was back in the 1940s.

Fast forward to the 1990s, and a team at CSIRO invents the wireless local area network (WLAN), which lets your devices connect to a network using radio waves.

In 1997, the Institute of Electrical and Electronics Engineers (IEEE) approved the 802.11 standard for how Wi-Fi should work, marking the official creation of Wi-Fi. This standardization led to more Wi-Fi-compatible devices, including the one you're probably using to read this blog post.

9. The first search engine

If you thought Google was the first search engine, think again. In the 1990s, just like today, people wanted a way to search for specific files. Enter Archie, the first search engine.

A student at McGill University in Montreal named Alan Emtage created the Archie index for a school project in 1990. That's some project.

Many different search engines began to pop up after Archie. You had Gopher in 1991, Excite in 1993, and Yahoo! and WebCrawler in 1994. A few years later, in 1998, the search engine you know as Google came to be.

Today, Statcounter reports that Google holds about 92% of the search engine market share. According to Internet live stats, there are more than 3.5 billion searches every day.

10. The first banner ad posted

AT&T sponsored the first online ad in 1994.

Joe McCambley, then a creative director for an interactive agency named Modem Media, pitched the idea of a banner ad — then a brand-new concept — to one of the agency's clients, AT&T. The agency and AT&T worked together to create an ad that was hosted on Hotwired.com.

Instead of promoting AT&T, the ad urged viewers to click on the banner (using the clever call to action "you will") to see a total of seven museums containing great works of art from around the world.

According to Modem Media founder GM O'Connell, the first banner ad had a 44% clickthrough rate. Although the ad didn't reach that many people — the Internet was still in its infancy in 1994 — it received a lot of press attention, and helped AT&T secure its position as an industry innovator.

11. The first secure ecommerce transaction

While there's some debate as to who sold the first item securely on the Internet — securely meaning they used encryption technology to protect credit card information — this ecommerce transaction occurred in 1994.

Some say that the first item sold securely online was a Sting CD, at the cost of $12.48. Dan Kohn created NetMarket, the website that sold the CD.

Another company also lays claim to the secure ecommerce throne. The team at the Internet Shopping Network says they used the Internet to sell computer equipment a month before NetMarket sold its CD.

No matter who sold what first, both NetMarket and Internet Shopping Network paved the way for online marketplaces like Shopify, Alibaba, and Amazon to succeed.

Ecommerce as a whole keep growing, with one report sharing more than 27% growth globally.

12. The first item sold on Amazon

Founded in Jeff Bezos' garage in Washington in 1994, Amazon.com sold its first item — a book — on April 3, 1995. The book was Douglas R. Hofstadter's Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought. The book is still for sale on Amazon.com and has a 4.5-star rating.

From September 1, 2020 to August 31, 2021, Amazon selling partners in the U.S.

sold 7,400 products every minute. Not bad for a company originally started in a garage selling a few books!

13. The first social media platform

Before TikTok. Before Facebook. Before Tumblr.

Before YouTube. Before even MySpace… we had Six Degrees.

Referred to as the first social media platform, Six Degrees operated on the same basic principles as modern social media companies. You would create a profile, connect with friends, and communicate with people in your network.

This platform was founded in 1996 by Andrew Weinreich and had millions of registered users at its peak.

Now, Facebook holds the title of most popular social network with 2.9 billion users worldwide.

14. The first person on Facebook (who wasn't a co-founder)

The first few accounts created on Facebook were reserved for testing, Mark Zuckerberg, and his co-founders.

But the first real user to join the site was Arie Hasit, a friend of the co-founders and roommate of Mark Zuckerberg, who joined the site in early 2004.

Hasit, whose account ID is 7, currently lives in Israel and works as a rabbi.

Currently, Facebook's parent company — Meta — boasts 3.64 billion monthly active people using its family of products, including Facebook, Messenger, Instagram, and Whatsapp.

15. The first Tweet ever Tweeted

Twitter co-founder Jack Dorsey made the first tweet on March 21, 2006, when the site was still known as Twttr:

> *"just setting up my twttr" — Jack (@jack) March 21, 2006*

According to Twitter's earnings release, the company had 217 million monetizable daily active users (mDAU) in Q4 2021.

16. The first YouTube video uploaded

YouTube co-founder Jawed Karim uploaded the first video to the site on Saturday, April 23, 2005. The video was a 19 second clip of Karim at the San Diego Zoo talking about elephants.

Today, more than 2 billion logged-in users watch content on YouTube every month. Not only that, but people watch more than one billion hours of video each day.

17. The first Instagram photo posted

Instagram co-founder Kevin Systrom posted the first photo on Instagram on July 16, 2010, a few months before the app's initial launch on iOS in October. The photo was of a golden retriever.

According to Hootsuite, 1.22 billion people use Instagram every month, and it's the second most-downloaded app worldwide, falling right behind TikTok.

18. The first commercial transaction using Bitcoin

Strap in. This section covers the rollercoaster that is cryptocurrency.

In 2010, the first commercial transaction using Bitcoin occurred when a man in Florida, USA bought two pizzas from Papa John's for 10,000 BTC (Bitcoins). This transaction assigned a value to Bitcoins.

Bitcoin entered the scene in 2008 when the domain name bitcoin.org came to be.

A white paper on Bitcoin was distributed to a cryptography mailing list shortly after, published by either a group or a single person using the pseudonym Satoshi Nakamoto. No one really knows who actually published the white paper to this day.

Bitcoin may be an important part of Internet history, but it wasn't the first form of cryptocurrency.

Computer scientist and cryptographer David Chaum used his own privacy-focused formula to create DigiCash in 1995, which is regarded as one of the first versions of digital money. Other versions of decentralized currency came into existence after DigiCash, but none became as popular as Bitcoin.

Bitcoin's value has seen its highs and lows since it gained popularity, and many other forms of cryptocurrency have come into existence.

You've got your Ethereum, Litecoin, and — of course — Dogecoin.

There are many other options out there for investors, but we're not going to give you investment advice.

EVOLUTION OF THE INTERNET

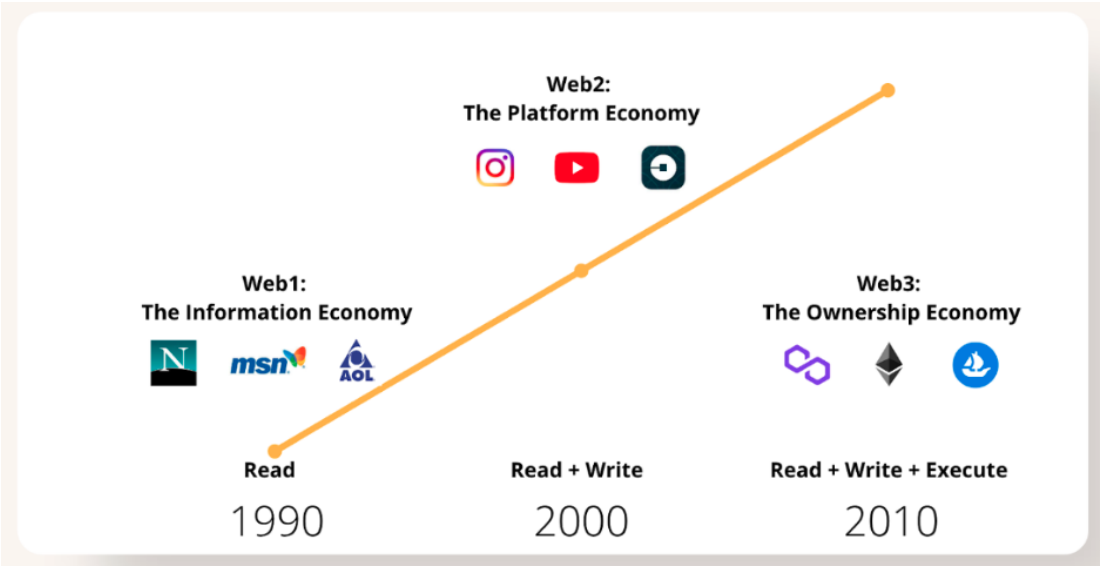
Evolution of the Internet - from web1.0 to web3

The World Wide Web is the key tool used by billions of people to distribute information, read and create it, and connect with others over the internet. The web has evolved considerably over time, and its contemporary uses are almost unrecognizable from its inception. The development of the web is often classified into three stages: Web 1.0, Web 2.0, and Web 3.0.

What is Web 1.0?

Web 1.0 was the very first version of the internet. Consider the read-only or syntactic web to be Web 1.0. The majority of participants were content consumers, whereas the creators were mostly web developers who produced websites with predominantly textual or visual content. Web 1.0 existed approximately between 1991 and 2004.

In Web 1.0, sites supplied static content rather than dynamic hypertext mark-up language (HTML) content. The data and content came from a static file system rather than a database, and there was limited interactivity on the web pages.



What is Web 2.0?

Most of us have only seen the web in its present form, which is also known as Web 2.0, the interactive read-write web, and the social web. You don't have to be a developer to participate in the Web 2.0 creation process. Many applications are built in such a manner that anybody can create content.

You may think and share your ideas with the rest of the world. In Web 2.0, you may also submit a video and make it accessible for millions of people to view, engage with, and comment on. Web 2.0 apps include YouTube, Facebook, Flickr, Instagram, Twitter, and other social media platforms.

Web technologies like HTML5, CSS3, and Javascript frameworks like ReactJs, AngularJs, VueJs, and others enable businesses to create new concepts that allow users to contribute more to the social web. As a consequence, since Web 2.0 is designed around people, developers just need to provide a system to empower and engage users.

Consider how popular applications like Instagram, Twitter, LinkedIn, and YouTube were in their early days compared to how they are now. All of these businesses generally go through the following steps:

- The organization releases an app.
- It tries to enroll as many individuals as possible.
- The user generates content and engagement.
- Then it profits from its user base.

When a developer or organization develops a successful app, the user experience is often exceedingly smooth, particularly as the program's popularity rises. This is why they were able to get momentum so quickly in the first place. Many software companies were unconcerned about monetization at first. Instead, they were primarily concerned with acquiring and maintaining new customers, although they would ultimately start profiting.

However, the constraints of accepting venture capital often affect the life cycle and, ultimately, the user experience of many of the services we use today. When a company seeks venture money to build an application, for example, its investors often anticipate a return on investment in the tens or hundreds of times that they put in. This implies that, rather than following a long-term, organic development plan, the firm is typically driven down one of two paths: marketing or data sales.

More data implies more targeted marketing for several Web 2.0 businesses, including Google, Facebook, and Twitter. This leads to more clicks and, as a consequence, more ad revenue. The use and centralization of user data are critical to the operation of the web as we know and use it today. As a consequence, data breaches are fairly frequent in Web 2.0 apps. There are even websites devoted to monitoring data breaches like https://haveibeenpwned.com/ and notifying you when your personal information has been compromised.

In Web 2.0, you have no control over your data or how it is kept. In reality, corporations regularly monitor and keep user data without their knowledge or consent. All of this data is then owned and managed by the companies in charge of these platforms. Furthermore, when governments feel someone is expressing a viewpoint that opposes their propaganda, servers are routinely taken down or bank accounts are seized. Using centralized servers, governments may easily intervene, control, or shut down programs.

Governments routinely meddle in banks since they are also computerized and centralized. They may, however, suspend bank accounts or limit access to money during times of extreme volatility, excessive inflation, or other political upheavals. Many of these issues will be addressed by Web 3.0, which seeks to fundamentally rethink how we build and interact with apps.

What is Web 3.0?

Web 3.0, also known as Semantic Web or read-write-execute, is the phase (starting in 2010) that suggests the future of the web. Artificial Intelligence (AI) and Machine Learning (ML) allow computers to evaluate data in the same manner that people do, allowing for the intelligent development and dissemination of useful information based on a user's individual requirements.

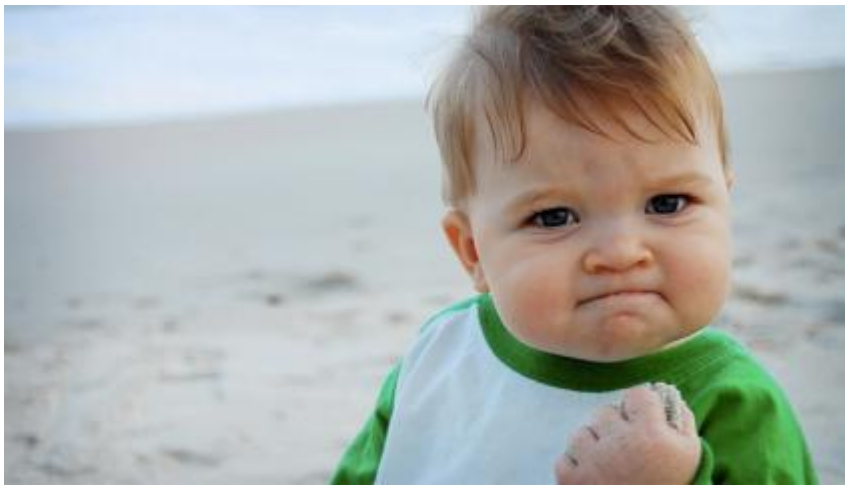
Although there are several major differences between Web 2.0 and Web 3.0, decentralization is the defining parameter. Web 3.0 developers almost never design and deploy programs that operate on a single server or store data in a single database (usually hosted on and managed by a single cloud provider).

Web 3.0 applications are based on blockchains, which are decentralized networks of many peer-to-peer nodes (servers), or a combination of the two. These applications are known as decentralized apps (DApps), and the term is often used in the Web 3.0 ecosystem. Participants in the network (developers) are rewarded for providing the best quality services in order to maintain a robust and secure decentralized network.

Many leading companies are presently conceptualizing and developing Web3, with Ethereum standing out in terms of early user acceptance and scope. While Web3's underlying architecture has not yet been determined, its decentralized nature is a key component of its planned design.

INTERNET CULTURE

Memes and Their Impact on Society



You probably see 'Demotivators' (for example 'Never give up!'?) or pictures of a cat with amusing captions every day on social media; you may even have shared images like this with others. These are examples of memes – virally transmitted remixed images with words that poke fun at cultural or social norms and that can change meaning multiple times through further sharing and remixing. In doing so they may not change the world, but they do act as sensitive indicators to public opinion.

Finding their origins in the likes of puns and cartoons, internet slang and geek forums, memes are sarcastic and have the ability to turn a message on its head. For example, think about a demotivator that mocks a motivational poster and says 'Never give up! Never stop trying to exceed your limits. We need the entertainment.' Memes mock their message and their format at the same time.

Once invented, memes spread across the internet on 4Chan, 9GAG and social media and are used to express thoughts on personal life, Kim Kardashian, fashion, politics and gender roles – pretty much everything. We see and understand images much faster than text and we respond faster to them. But memes are not just simple visual gags – they always have a hidden joke. To comprehend why it is supposed to be funny, a meme requires thought. That is its real superpower: visibility, transmission and mental application and thought.

Memes are used in many ways:

Everyday slang and folklore. Nowadays, people use memes to communicate on all topics. Think about 'grumpy cat' or 'Keanu Reeves' memes. They are used to express surprise, disappointment, excitement, skepticism. What distinguishes memes from other forms of

image-based artefacts is the strong emotion embedded in them. There is always some kind of feeling that the meme directs you towards – like disgust or happiness. Memes are subjective.

Political mindbombs. The founder of Greenpeace coined the idiom 'mindbombs'. He understood that once seen, a striking image cannot be unseen and so sent expensive ships with photographers to follow whale trawlers. These ships were not there to stop the slaughter of the whales but to record it. The Greenpeace activists were able to take heartbreaking pictures of suffering animals which were then circulated to journalists and the world's press. In public opinion the issue of whale hunting is now often connected to the images of suffering – all thanks to a 'mindbomb'. Another example of a mindbomb that affected a political career is that of Ed Miliband, ex-leader of the Labour party who was photographed struggling to eat a bacon sarnie. The memes that mocked this breakfast fiasco became criticism of Miliband for being out of touch with working class people. Was this ethical? Was this viral shame deserved? This and many other cases make us question how ethical memes can be and how memes can damage reputations.

Fast-food media. Due to their high visibility and humour, memes serve as fast-food media. They are like cheeseburgers – highly tempting in colour, smell and texture, but low in nutritional value. They feed you up a bit (on the news agenda), but you really need a decent meal (or reading a respected newspaper) to nurture your body and mind.

Memes have this cosy quality of inside jokes and community chat; however, they cannot unite people into communities if other ties don't exist. Memes do not age well – most of them evaporate within days, but the points they make can leave a lasting effect on society and politics. In countries with censorship, memes can obtain an even bigger value – as people throw them into public discourse to send resistant messages and oppose corruption. But what if sinister forces utilise memes too? These forces can use memes to manipulate people and promote ugly ideas. At what point does a cozy inside joke become a powerful political message? What is clear is that memes are an influential communication tool of our society and that they are here to stay.

E-COMMERCE AND ONLINE BUSINESS

The Effects of E-Commerce in Reshaping Customer Shopping Habits

The biggest impact ecommerce has had on consumer shopping habits is that consumers can shop from anywhere, anytime. They no longer have to wait until store hours to make a purchase. While the ability to research and shop online has been around for a while, mobile has taken ecommerce to the next level because shoppers can use the device at any point during the sales cycle.

By 2025, mobile commerce sales are projected to reach $728.28 billion and make up 44.2% of retail commerce sales in the US. Consumers use mobile in a variety of different ways throughout the sale cycle, notes Nels Stromborg, the North America managing director at Retale. These use cases include:

- To discover new products

- To locate products and compare prices
- To create and manage shopping lists
- To make purchases
- To review purchases

The rise of mobile shopping has blurred the line between the physical store and the online experience. Rather than having two distinct channels, both channels can be used in conjunction to optimize the shopping experience. Though some of the legacy brick-and-mortar brands have had trouble keeping up with the growth of ecommerce, it isn't the kiss of death to physical stores. In fact, big companies like Amazon and Alibaba have opened up brick-and-mortar locations.

The kiss of death comes when companies are not able to create a seamless experience between online and offline shopping, explains Tom Popomaronis, senior director of product innovation and business development at the Hawkins Group. The companies that have been able to make the transition have created apps, optimized their ecommerce stores, and started selling products through their social media channels.

By doing this, they have given consumers the choice of where, when, and how to shop. A shopper can purchase a product online at midnight, receive it the next day, and then return it to a physical store if unhappy with the product. That's the power of mobile ecommerce — the ability to create a more seamless, omnichannel shopping experience. It is an experience that customers have come to expect.

Customers Expect More Personalized Experiences

The progression of ecommerce has advanced the customer expectations of the companies they buy from. So, what *do* customers expect? They expect a seamless shopping experience that is personalized to them — one that is consistent no matter what device they are using for their shopping or what stage of the buying process they are in. In addition, people are 40% more likely to spend more than they'd planned if their experience is personalized.

Richard Kestenbaum, partner at Triangle Capital LLC, says the challenge for retailers is they have to offer better experiences than they have in the past to motivate customers to come in or make a purchase. Companies are doing this by creating omnichannel, personalized experiences with content that "resonates, engages and delights consumers" at every stage of the buying process, says marketing consultant Andy Betts.

Take, for example, GOAT, the mobile sneaker marketplace that lets users create wish lists. Then, when those sneakers go on sale, or the price drops into the shopper's target price range, the app sends them a push notification. The company has created a personalized experience that is driving business, as the company now has more than 7 million users worldwide.

Shopping Has Become a Social Activity

When companies like GOAT create a great shopping experience, people want to share that experience with others. Digital marketing has facilitated that sharing and turned shopping into a social activity. What's more, consumers today rely on the opinions of others to guide their purchase decisions, and they have immediate access to those reviews. Anyone on social media can be an influencer for a brand. Social platforms and online review sites have opened the floodgates for word-of-mouth advertising via product reviews.

Today, 95% of shoppers read reviews before making a purchase. And it does not matter to consumers that these reviews are from complete strangers. They trust the reviews more than they trust what brands themselves are saying. That's why, consumers, not brands, are more responsible now for shaping the perception of a brand, says Chris Campbell, CEO of ReviewTrackers. These online reviews have become so important that 94 percent of people have avoided a business because of a negative online review, the company's research shows.

Retailers have recognized the power of these channels to shape shoppers' opinions and have begun engaging with their customers on social media and online review platforms. That engagement has played a big role in facilitating customers' desires for more information before making purchases. A side-effect of that engagement is consumers are more informed than ever before about the products they are buying and the companies they are giving their money to.

Shoppers Are Becoming Their Own Salespeople

In addition to online reviews, consumers can access product and company information that they can read and analyze before buying. These better-informed customers are changing the role of salespeople in companies. These customers' expectations are higher, and companies are having to change their approach to meet those expectations.

Before digital media, customers relied on salespeople to guide them on their path to making the best purchase. Now customers enter stores, online and offline, armed with the information they need to make a purchase.

Shoppers Have Greater Access to Goods

Today's consumers have access to more information, and ecommerce has given them access to products from around the world. New trade agreements and advances in ecommerce technology has opened the door for brands to sell outside their domestic markets and customers are bought in. Global cross-border sales are expected to hit more than $4 trillion by 2027.

While shoppers are willing to purchase products from outside their home country, they still have high expectations for their online shopping experiences. Cross-border shoppers expect merchant websites to appear in their native languages and accept local currency and payment methods.

The Evolution Continues

All customers have similar basic expectations when they shop. They want the products they want when they want them, and they don't want to pay too much for them. This is why ecommerce has grown to be the preferred shopping method for consumers.

Ecommerce gives consumers access to information, the ability to shop on different devices and the option to share their experiences with others, which has completely altered their expectations and the way they shop. Customer shopping habits will continue to evolve with technology, and companies will have to continue to adapt to maintain relevance.

The Future of Online Marketplaces

The world of online marketplaces is evolving at an unprecedented pace, and it's crucial for business owners and consumers alike to stay ahead of the curve. In this edition of Safecastle Connects, we delve into the future of online marketplaces, exploring the latest trends, emerging technologies, and strategies to navigate this dynamic landscape.

Marketplace Growth Statistics: A Snapshot of 2024

The online marketplace is set to soar, with projections indicating a staggering growth of $3,567.00 billion in 2024 (source: Comestri). Forbes estimates that B2C marketplaces will reach $3.5 trillion in sales by the same year. As the industry becomes increasingly competitive, it's essential to understand the transformative trends shaping the future.

Why Online Marketplaces Are Booming

1. **Convenience for Customers:** With the rise of online marketplaces like Amazon, consumers can access a vast array of products without leaving their homes, leading to unparalleled convenience.
2. **Global Expansion:** Digital marketplaces eliminate geographical boundaries, allowing businesses to expand globally and target specific customer segments across regions.
3. **Rigid Marketplace Policies:** Established marketplaces, such as eBay and Amazon, have instilled trust through strict policies on refunds, authenticity, and delivery timelines.
4. **Lucrative Offers and Discounts:** Online platforms attract customers with compelling deals, especially during events like Black Friday and Cyber Monday.
5. **No Physical Presence Required:** Virtual stores eliminate the need for a physical storefront, significantly reducing costs for sellers.
6. **Lower Marketing Costs:** Digital marketing has made branding more affordable, enabling businesses to reach a broader audience quickly.

7. **Easy Marketplace Setup:** Building an online marketplace is now more accessible than ever, with user-friendly platforms like Dokan simplifying the process.

The Future of Online Marketplaces in 2024 & Beyond

Upcoming Trends:

1. **Voice Command Searching:** Approximately 72% of people use voice commands to search for products, emphasizing the growing significance of voice search in online marketplaces.
2. **Inclusion of Vertical Marketplaces:** Niche-based platforms, known as Vertical Marketplaces, are gaining popularity for catering to specific genres and providing specialized services.
3. **Brick and Click Policy:** The synergy of online and offline sales, known as the Brick and Click policy, allows physical stores to maintain a presence in both realms.
4. **Introduction of Recommerce:** The trend of reusing products for sustainability, known as Recommerce, is on the rise, offering consumers affordable options for second-hand goods.
5. **Mobile Shopping's Popularity:** With over 70% of people using smartphones for shopping, mobile commerce is becoming increasingly popular, although desktop users still play a significant role in larger orders.
6. **AI Technology for Constant Service:** Artificial Intelligence, especially chatbots, is becoming integral to providing continuous and responsive customer service in online marketplaces.
7. **Omni-Channel Shopping:** Shoppers often start their journey on one platform and end it on another, emphasizing the importance of a seamless experience across various channels, known as Omni-Channel shopping.

Upcoming Technologies in Virtual Marketplaces:

1. **Internet of Things (IoT):** IoT ensures granular customer service by gathering comprehensive data on individuals.
2. **Facial Recognition:** Enhances the shopping experience by identifying user preferences through facial recognition technology.
3. **Augmented Reality:** Allows users to experience products in a virtual environment, addressing the lack of realism in the online marketplace.
4. **Lifting Gear for Better Maintenance:** Robotic solutions aid in transporting products efficiently within warehouses, streamlining logistics.

Drawbacks to Be Cautious About:

1. **Ghost Charges:** Some platforms impose hidden charges on orders, leading to customer dissatisfaction.
2. **Unfriendly Service Charges:** Fraudulent sellers and unfriendly service charges can damage the credibility of online marketplaces.
3. **Lack of Transparency:** Some retailers offer enticing deals but fail to deliver on service quality, highlighting the importance of transparency.

THE SURFACE WEB

The Surface Web, also known as the Visible Web or Indexed Web, refers to the portion of the internet that is accessible and indexed by standard search engines like Google, Bing, and Yahoo. This part of the web includes publicly available websites and content that can be easily found through typical search queries. It encompasses a wide range of resources such as commercial websites, blogs, news sites, and social media platforms.

Additionally, the Surface Web includes public databases, educational resources from institutions like universities, and information from government websites and official public documents.

In contrast, the Deep Web contains parts of the web not indexed by search engines, including private databases and password-protected sites, while the Dark Web is a subset of the Deep Web that requires specific software to access and often hosts illicit activities.

Importance and Usage of the Surface Web

The Surface Web is the most familiar and frequently used part of the internet, playing a crucial role in everyday life and various industries. Here are some key aspects of its importance and usage:

1. Accessibility and Information Availability

The Surface Web provides easy access to a vast amount of information on almost any topic imaginable. It includes websites, blogs, news articles, and public databases that are accessible to anyone with an internet connection. This accessibility is crucial for education, research, and staying informed about current events.

2. E-commerce and Business Operations

Businesses rely heavily on the Surface Web for their operations. E-commerce platforms like Amazon, eBay, and countless online stores allow consumers to purchase goods and services conveniently. Companies use websites to showcase their products, offer customer support, and engage with their audience, driving the global economy.

3. Communication and Social Interaction

Social media platforms such as Facebook, Twitter, and Instagram are part of the Surface Web, enabling billions of people to connect, share information, and communicate instantly. These platforms have become essential tools for personal interaction, marketing, and even political movements.

4. Educational Resources

The Surface Web is a treasure trove of educational content. Websites of educational institutions, online courses, and open-access journals provide students and lifelong learners with valuable resources. Platforms like Coursera, Khan Academy, and educational YouTube channels offer free or affordable learning opportunities.

5. News and Media

News websites from reputable media organizations like CNN, BBC, and The New York Times are accessible through the Surface Web. These sites provide timely news updates, in-depth analysis, and a variety of viewpoints, helping people stay informed about global and local events.

6. Government and Public Services

Government websites are part of the Surface Web, offering citizens access to important information and services. These sites provide resources such as tax information, public records, and services like renewing licenses or applying for permits. They ensure transparency and ease of access to public information.

7. Entertainment and Leisure

The Surface Web offers endless entertainment options, including streaming services, gaming sites, and online communities. Platforms like Netflix, YouTube, and Twitch cater to diverse interests and provide countless hours of entertainment.

How to Access the Surface Web?

Accessing the Surface Web is straightforward and involves using standard web browsers and search engines. To begin, you need an internet connection and a web browser such as Google Chrome, Mozilla Firefox, Microsoft Edge, or Safari.

Once your browser is open, you can enter a URL directly into the address bar if you know the specific website you want to visit, such as www.example.com. For general searches, you can use search engines like Google, Bing, or Yahoo.

Simply type your query into the search bar, and the search engine will provide a list of indexed websites relevant to your keywords.

These websites are part of the Surface Web, meaning they are publicly accessible and have been indexed for easy discovery. Utilizing bookmarks and browsing history features in your web browser can enhance your browsing experience by allowing quick access to frequently visited sites.

For additional security, ensure your browser is updated and consider using browser extensions that enhance privacy and block malicious content. By following these steps, you can effectively navigate and utilize the vast resources available on the Surface Web

THE DEEP WEB

Unlike the dark web, which is deliberately obscured by layers of technology, the deep web exists just below the surface web. The majority of the deep web consists of regular websites that require users to create an account before they can be accessed.

If the surface web is the visible part of an iceberg above water, the deep web is the part submerged beneath – much larger but hidden from plain view. Some people use the terms 'dark web' and 'deep web' interchangeably; however, significant portions of the deep web are completely legal and safe to use.

A majority of the web consists of databases and intranets. The former includes privately protected file collections that one cannot access without the correct credentials. The latter includes internal networks for governments, educational facilities, and corporations. The 'members only' parts of public platforms are also a part of the deep web.

All web pages not indexed by web crawlers are considered to be a part of the deep web. The content is generally more secure and clean than that of the surface web. This is because deep web content is usually well-maintained. Security tools such as firewalls help in this endeavor.

Simple examples of deep web content include financial data, social security databases, email inboxes, social media, medical documentation, legal files, blog posts that are pending review and web page redesigns that are in progress. These pages are mostly obscured from the surface web to secure user data and privacy rather than any nefarious purpose.

However, the deep web is not entirely devoid of danger. Some portions of the deep web allow users to overcome legal restrictions to access content that is not lawfully available in their geographical location. It is even possible to illegally download movies, music, and other digital media without paying for it. Naturally, these lawless segments of the deep web are rife with malware and other cyber threats.

While the content on the dark web has the potential to be more dangerous, this content is usually walled off from regular users. However, it is entirely possible for regular users to accidentally come across harmful content while browsing the deep web, which is much more easily accessible. This makes deep web security important for individuals and enterprises alike.

Surface Web vs. Deep Web

FEATURE	SURFACE WEB	DEEP WEB
Accessibility	Publicly accessible and indexed by search engines	Not indexed by search engines; requires specific access methods
Content Types	Public websites, blogs, news sites, social media	Private databases, password-protected sites, internal networks
Searchability	Easily searchable through standard search engines	Not searchable through standard search engines
Examples	Google, Wikipedia, Amazon, CNN, Facebook	Online banking sites, academic databases, medical records, private forums
Security	Generally, less secure; open to public	More secure; restricted access
Size	Smaller compared to the Deep Web	Significantly larger than the Surface Web
Usage	Everyday browsing, e-commerce, social interaction	Sensitive information, private communication, research data
User Access	Anyone with internet access	Requires specific permissions, credentials, or network access

THE DARK WEB

The dark web is a subsection of the deep web including websites that one can only access through purpose-built web browsers. Some estimates mark the extent of the dark web as being much smaller than that of the surface web (which is the regular internet we use every day). However, it is challenging to measure how widespread the dark web is due to its decentralized and obscure nature.

Very few regular users will ever interact with the dark web. While the deep web usually requires just a username and password to be accessed (think of your email inbox), the dark web cannot be reached without the right software or hardware.

The architecture of the dark web features some key elements that make it an anonymous sanctuary for censured activities.

Due to their unique registry operator, websites on the dark web are inaccessible by regular web browsers such as Google Chrome and Mozilla Firefox.

Web crawlers cannot index the content hosted on the dark web. Network security tools such as encryption and firewalls prevent these search tools from discovering these websites.

Access to dark web content is restricted with the help of virtual traffic tunnels through randomized network architecture.

The layman associates the dark web with illegal content and criminal activities such as trading illicit products and services. While this is mainly true, this online framework also has applications for legitimate entities. For instance, journalists in autocratic nations can use the dark web to pass information to their colleagues in other parts of the world.

The dark web was originally a domain exclusively used by cybercriminals and governments. Today, this is no longer true–advanced encryption technologies have made the anonymization of web surfing a hassle-free affair for regular users. The Tor browser is a popular tool for accessing the dark web.

Most enterprises do not have a direct use case for the dark web. However, cybersecurity organizations could benefit from indirect participation in discussions regarding hacking and exploit trading that regularly takes place on the dark web. This would give them insights into yet-unknown vulnerabilities and allow them to gain knowledge of exploits before they can turn into widespread cyber threats.

Surface Web and the Dark Web

Here is a tabular comparison between the Surface Web and the Dark Web:

FEATURE	SURFACE WEB	DARK WEB
Accessibility	Publicly accessible and indexed by search engines	Requires special software (e.g., Tor) for access
Content Visibility	Visible to and searchable by standard search engines	Not indexed by standard search engines; hidden
Types of Content	Legitimate websites, blogs, news sites, social media	Often contains illicit activities, black markets, forums
Examples	Google, Facebook, Amazon, BBC, Wikipedia	Darknet markets, Silk Road, private forums
Security	Generally secure but subject to standard cyber threats	Higher anonymity, often used for illegal activities
Legal Status	Legal and regulated	Contains both legal and illegal activities
Usage	Everyday browsing, e-commerce, social interaction	Anonymous browsing, purchasing illegal goods/services
Searchability	Easily searchable with search engines	Requires specific URLs or access points

THE ROLE OF AI ON THE WEB

With the dawn of Artificial Intelligence, humans are at the threshold of a new era. Each passing innovation is blurring the difference between natural and artificial. Artificial Intelligence has not only the capability to mimic human intelligence but can also revolutionize every facet of our life.

The goal of AI-driven Innovations is to perform difficult tasks instead of humans with accuracy and precision.

What Is Artificial Intelligence?

Artificial Intelligence is the use of computers and machines to perform human based tasks. These computers have characteristics similar to the human brain. These machines can think critically, increase productivity and help in Data-driven decision making. In the year 2024, there is no one who doesn't know about ChatGPT. However, the tool only represents a small portion of how AI is being used in today's fast and digital world.

AI Intelligence vs Human Intelligence

Before continuing the topic of AI in web development lets discuss the difference between AI Intelligence vs Human Intelligence to know more about artificial intelligence. Following is some of the main differences:

Aspect	Artificial Intelligence (AI)	Human Intelligence
Learning Process	Based on smart algorithms and data processing	Complex cognitive processes and experiences
Adaptability	Adapts based on programmed rules and patterns	Adapts through learning, experience, and emotional intelligence
Creativity	Limited ability for creativity and originality	Demonstrates creativity, imagination, and innovation
Emotional Response	Lacks emotional understanding and empathy	Displays emotions, empathy, and interpersonal skills
Processing Speed	Processes data at incredibly high speeds	Processing speed varies; slower than AI in some tasks
Contextual Understanding	May struggle with context and nuances	Understands context, ambiguity, and implicit meaning
Problem-solving	Excellent at solving specific tasks with defined rules	Can solve a wide range of problems, including novel and complex situations
Consciousness	Lacks self-awareness and consciousness	Exhibits self-awareness and consciousness
Intuition	Lacks intuition and gut feeling	Relies on intuition for decision-making in ambiguous situations

CHAPTER

2

CYBERSECURITY

COMMON CYBER THREATS AND HOW TO PROTECT YOURSELF

Cybersecurity threats can be daunting, but understanding and preparing against them is essential for today's digital citizens, business leaders, and organizations. This article demystifies current cybersecurity threats, identifies who is perpetrating these attacks, and presents practical defense strategies to prevent a data breach.

Cyber-Threats

A cyber threat is a harmful activity committed with the intent of destroying, stealing, or disrupting data, critical systems, and digital life in general. Computer viruses, malware attacks, data breaches, and Denial of Service (DoS) assaults are examples of these risks.

Cybersecurity threats come from a variety of places, people, and contexts. Malicious cyber threat actors can include:

- **Criminal organizations**
  Organized groups of hackers aim to break into organizations for financial gain. These cyber threat actors use phishing, spam, spyware, and malware for extortion, theft of private information, and online scams that are run like corporations, with large numbers of employees developing attack vectors and executing attacks
- **Nation-states**
  Hostile countries can launch cyber-attacks against local companies and institutions to interfere with communications, cause disorder, and inflict damage.
- **Terrorist organization**
  Terrorists conduct cyber-attacks aimed at destroying or abusing critical infrastructure, threatening national security, disrupting economies, and causing bodily harm to citizens.
- **Rogue Insiders**
  Employees with legitimate access to company assets abuse their privileges to steal information or damage electronic assets for economic or personal gain. This insider threats may be the target organization's employees, contractors, suppliers, or partners.

Types Of Cyber Threats

Cybersecurity attacks take various forms, each with its own set of techniques and objectives. We have put together a list of the *Top 9 cyber threats that could negatively impact your business*.

1. Malware Attacks

Malware is a program inserted into a system intending to compromise data confidentiality, integrity, or availability.

Attackers may embed malware in app downloads, mobile websites, or phishing emails and text messages. Once compromised, a mobile device can give the malicious actor access to personal information, location data, financial accounts, and more.

Here are some common types of malwares:

- **Virus:** This type of malware attaches itself to clean files and spreads throughout a computer system as those files are executed. It can quickly corrupt or delete data on a device.
- **Worm:** Worms infect entire networks of devices either by local networks or through the internet. They operate by exploiting vulnerabilities in operating systems.
- **Trojan:** Unlike viruses, Trojans don't replicate themselves, but they can be just as destructive. They disguise themselves as legitimate software but act maliciously once inside the device.
- **Spyware:** As its name implies, this type of malware spies on users. It can gather data like user habits, logins, credit card information, and other personal details.
- **Ransomware:** This malware locks or encrypts data on a victim's device and demands payment (ransom) to restore access.
- **Cryptojacking:** attackers deploy software on a victim's device, and begin using their computing resources to generate cryptocurrency, without their knowledge.
- **Adware:** While not always malicious in intent, adware presents unwanted advertisements to the user, which can lead to other types of malwares being installed.
- **Rootkit:** These are designed to gain administrative access to a device. Once they do, they become deeply embedded and are difficult to detect and remove.
- **Botnet:** This is a network of compromised devices that are controlled remotely by an attacker, usually to carry out large-scale attacks or to send spam.
- **Fileless Malware:** Unlike traditional malware that relies on files, fileless malware resides in a system's RAM and exploits legitimate programs to infect a computer.
- **Mobile Malware:** As mentioned in the article snippet you provided, this targets mobile devices and can include a range of malicious code types tailored for these devices.

2. Social Engineering Attacks

Social engineering remains one of the most dangerous hacking techniques cybercriminals employ, largely because it relies on human error rather than technical vulnerabilities. The victim provides sensitive information or unwittingly installs malware on their device because the attacker poses as a legitimate actor.

Types of Social Engineering Attacks:

- **Phishing:** Phishing attacks use emails to trick the recipient into disclosing confidential information or downloading malware by clicking on a hyperlink in the message.

- **Spear Phishing:** A more sophisticated form of phishing where the attacker learns about the victim and impersonates someone he or she knows and trusts.
- **Vishing (voice phishing):** the imposter uses the phone to trick the target into disclosing sensitive data or grant access to the target system. Vishing typically targets older individuals but can be employed against anyone.
- **Smishing (SMS phishing):** the attacker uses text messages as a means of deceiving the victim.
- **Baiting:** the attacker lures a user into a social engineering trap, usually with a promise of something attractive like a free gift card. The victim provides sensitive information such as credentials to the attacker.

3. Supply Chain Attacks

These cybersecurity attacks mainly aim to infect genuine applications and spread malware through source code, build processes, or software update mechanisms.

Attackers search for insecure network protocols, server infrastructure, and coding techniques to steal data and compromise build and update processes. They modify source code and hide malicious content, making it difficult to detect the threat.

In a software supply chain attack, the software vendor is unaware that their applications or updates are infected with malware. Malicious code runs with the same trust and privileges as the compromised application.

There are different types of intended targets of supply chain attacks:

- Compromise of build tools or development pipelines
- ode signing procedures, server resources, or developer accounts.
- Attackers may also send malicious code as automated updates to hardware or firmware components or pre-install it on physical devices.

4. “Man in the Middle” (Mitm) Attacks

MitM attack involves intercepting the communication between two endpoints, such as a user and an application. The attacker can eavesdrop on the communication, steal sensitive data, and impersonate each party participating in the communication.

Examples of MitM attacks include:

- **Wi-Fi eavesdropping:** Wi-Fi eavesdropping is a cyberattack where an attacker creates a fake Wi-Fi connection to intercept users' data, such as login credentials and payment card details.
- **Email hijacking:** This cybersecurity attack occurs when an attacker impersonates a legitimate organization's email address to deceive users into revealing sensitive information or transferring money to the attacker.

- **DNS spoofing:** a domain is spoofed, directing a user to a malicious website posing as a legitimate site. The attacker may divert traffic from the legitimate site or steal the unsuspecting user's login credentials elsewhere.
- **IP spoofing:** An attacker can spoof an IP address to pose as a website and deceive users into thinking they are interacting with that website.
- **HTTPS spoofing:** HTTPS is generally considered the more secure version of HTTP, but can also be used to make malicious links to trick users of the browser into thinking that a malicious website is safe. The attacker uses "HTTPS" in the URL to conceal the malicious nature of the website from unsuspecting user.

5. Denial-of-Service Attack (DDoS attack)

Denial of service attacks is when an attacker takes over many (perhaps thousands) of devices and uses them to invoke the functions of a target system, e.g. a website, causing it to crash from an overload distributed denial of service on demand.

6. Attacks on IoT Devices

IoT devices like industrial sensors are vulnerable to multiple types of cyber threats. These include hackers taking over the device to make it part of a DDoS attack and gain unauthorized access to data being collected by the device.

Given their numbers, geographic distribution, and frequently out-of-date operating systems, IoT devices are a prime target for malicious cyber threat actors.

7. Zero-Day Exploits and Attacks

Zero-day exploits refer to security vulnerabilities that exist in a software operating system or network that the manufacturer is not aware of. For instance, a technology vendor could launch a new version of an OS/app that inadvertently includes a loophole that allows hackers to gain access to your data.

8. Injection Attacks

Injection attacks exploit vulnerabilities to insert malicious input into web application code. Such attacks may expose sensitive information, execute DoS attacks, or compromise the entire system.
Main vectors for injection attacks:

- **SQL Injection** is a type of cyber-attack where the attacker inserts an SQL query into an input channel, such as a comment field or web form. If the application is vulnerable, the data will be sent to the database, and the injected SQL commands will be executed. Since most web applications use SQL databases, they are susceptible to SQL injection attacks. A new variation of this attack is NoSQL injection, which targets databases that don't employ a relational data model
- **Code injection:** Attackers can exploit vulnerabilities in applications by injecting malicious code. When executed by the web server, the code behaves as part of the application.

- **Cross-Site Scripting (XSS)** is a cybersecurity attack where an attacker inputs a string of text containing malicious JavaScript. When the target's browser executes this code, it can redirect users to a malicious website or steal session cookies to hijack a user's session. An application is vulnerable to XSS if it doesn't sanitize user inputs to remove JavaScript code.

9. Password Attacks

Password attacks refer to any cyber-attacks in which hackers try to guess, brute force, or deceive you into revealing your passwords. There are several types of password-based cyber-attacks that you need to be mindful of:

- Password spraying is when hackers attempt to use the same password across multiple accounts. For instance, more than 3.5 million Americans use the password "123456". Brute force attacks occur when hackers develop software that tries various combinations of usernames and passwords until they find the correct one. They often use logins leaked to the Dark Web because many individuals reuse passwords across multiple accounts.

How To Avoid Being a Victim of Cyber-Threats

Numerous strategies exist to counteract cybersecurity threats, despite the challenges they present. Companies combat cybersecurity threats by implementing robust security measures

Here are some essential steps to consider:

1. **Patch Regularly:** Keep your operating system, software, and apps up-to-date with the latest security patches. Hackers often exploit outdated systems, so it's essential to check for and install updates regularly.
2. **Strong Authentication:** Utilize strong, unique passwords for each of your online accounts. Consider using a password manager to help you keep track of complex passwords. Enable two-factor authentication (2FA) whenever possible for an extra layer of security.
3. **Secure Networks:** Avoid using public Wi-Fi networks for sensitive transactions. If you must use public Wi-Fi, use a virtual private network (VPN) to encrypt your data and keep it private.
4. **Firewall and Antivirus Software:** Use a firewall to block unauthorized access to your computer. Install reputable antivirus and anti-malware software to detect and remove malicious software.
5. **Be Wary of Phishing Attempts:** Be skeptical of unexpected emails, especially those that ask for personal information or prompt you to click on a link. Verify the sender's identity before responding or clicking on any links.
6. **Limit Access:** Restrict access to your sensitive data. Only share it with those who absolutely need it, and be sure to revoke access when it's no longer necessary.

7. **Data Encryption:** Encrypt sensitive data, particularly when it's stored on portable devices like laptops or USB drives, which can easily be lost or stolen.
8. **Regular Backups:** Regularly back up your important data to an external drive or cloud storage. This ensures that you have a copy of your data in case of a cyber-attack like ransomware.
9. **Employee Training:** If you run a business, train your employees about the risks of cyber threats and how to prevent them. This includes safe internet usage, recognizing phishing attempts, and reporting any suspicious activities.
10. **Incident Response Plan:** Develop an incident response plan so you know what steps to take in case of a data breach. This should include who to contact, how to contain the breach, and how to notify affected individuals.
11. **Stay Informed:** Stay informed about the latest cyber threats and prevention strategies. Join forums, subscribe to newsletters, or follow cybersecurity experts on social media.

Cyber Threats and The Dark Web: The Role of The Dark Web

The dark web is an unregulated and anonymous part of the internet. It allows hackers to:

- Sell their creations multiple times, making it a multiplier for threats
- Provide a haven for cyber criminals to share knowledge
- Trade malicious software
- Launch attacks

The dark web serves as a multiplier for threats, with one hacker being able to sell his or her creation over and over, making the dark web an important facet of the cyber threat landscape.

It's a shadowy corner of the internet that often escapes the watchful eyes of law enforcement, creating challenges for cybersecurity professionals who strive to track and combat the increasingly sophisticated digital threats emanating from this hidden realm.

Data Privacy in the Digital Age: How to Protect Your Information

In the digital age, our lives are increasingly intertwined with technology, and our personal information is constantly being collected and stored by various entities. This data, ranging from our online activities and shopping preferences to our health records and financial transactions, is incredibly valuable and vulnerable. Data privacy has become a critical concern as we navigate the digital landscape, and it is essential to understand how to protect our information from privacy breaches and cyber threats. Let’s explore the importance of privacy in the digital age and Data privacy tips & tricks, the reasons behind data collection, and the distinction

between data protection and data privacy. We'll also offer practical tips on how to safeguard your data!

Who Collects Our Data and Why?

Data collection is a widespread practice conducted by various entities, which include:

- **Companies and organizations:** Many businesses collect data to better understand their customers, improve products and services, and personalize user experiences. This is especially true for app development companies. By analyzing user behavior and preferences, companies can tailor their offerings to meet individual needs and provide targeted marketing campaigns.
- **Advertisers and marketers:** They collect data to understand customer demographics, interests, and behaviors, allowing them to create personalized advertisements and increase the effectiveness of their campaigns.
- **Social media platforms:** They collect personal information, posts, and interactions. This data helps them enhance user engagement, deliver targeted content, and generate advertising revenue.
- **Governments and public institutions:** They collect data for the purposes of public safety, national security, and policymaking. Data collected by public institutions can inform decision-making processes, resource allocation, and the development of public services.
- **Service providers:** Internet service providers, telecommunications companies, and other service providers often collect data for billing purposes, network optimization, and service improvement.

What Is Data Privacy and Why Is It Important?

Data privacy refers to the protection and control of personal information, ensuring that individuals have the authority to determine how their data is collected, used, and shared. It encompasses the right of individuals to keep their personal information confidential and secure from unauthorized access, misuse, or abuse. Data privacy is of paramount importance for several reasons:

- **Protection of personal information** — The concept of data privacy safeguards sensitive personal information, such as financial details, health records, and identification data, from falling into the wrong hands. It ensures that individuals have control over who can access their data and for what purposes.
- **Preserving individual autonomy** — Privacy empowers individuals by giving them the autonomy to make decisions about their personal information. It allows individuals to maintain a sense of control over their digital identities and personal lives.

- **Mitigating identity theft and fraud** — Personal data can be exploited by cybercriminals for identity theft, financial fraud, and other malicious activities. Strong data privacy measures help reduce the risk of such incidents and protect individuals from potential harm.
- **Safeguarding reputation and professionalism** — Privacy breaches can lead to reputational damage and compromise professional relationships. Maintaining data privacy is crucial for individuals and businesses to preserve their reputations and maintain trust with clients, customers, and partners.
- **Ensuring compliance with regulations** — In many jurisdictions, data privacy is protected by laws and regulations. Organizations that collect and process personal data must comply with these regulations, such as the General Data Protection Regulation (GDPR) in the EU or the California Consumer Privacy Act (CCPA) in the USA.
- **Ethical considerations** — Respecting data privacy is an ethical obligation. Individuals have the right to privacy, and organizations should uphold ethical principles by ensuring data protection, transparency, and consent in their data handling practices.

Data Protection vs Data Privacy: Similarities and Differences

While data protection and data privacy are closely related concepts, they have distinct meanings and implications. Understanding the similarities and differences between these two terms is essential in comprehending the broader context of safeguarding personal information.

Similarities:

- **Focus on personal information** — Both data protection and data privacy revolve around the safeguarding of personal information. They aim to ensure that individuals' data is handled securely, confidentially, and in compliance with applicable regulations.
- **Security measures** — Both concepts involve implementing security measures to protect personal data from unauthorized access, use, disclosure, alteration, or destruction. This includes practices like encryption, access control, and regular security audits.

Differences:

- **Scope and perspective** — Data protection focuses on the technical and organizational measures implemented to secure personal data, such as firewalls, encryption, and data backup. It primarily concerns safeguarding data from breaches, cyberattacks, and accidental loss.

Data privacy, on the other hand, takes a broader perspective and encompasses the legal and ethical aspects of how personal data is collected, used, and shared. It involves ensuring individuals' rights to control their data and make informed choices about its handling.

- **Legal and regulatory framework** — Data protection is often governed by specific laws and regulations that dictate how personal data should be handled, stored, and processed. For instance, the GDPR in the EU and the CCPA in California set guidelines and requirements for data protection.

Data privacy, while influenced by legal frameworks, focuses on the broader principles and ethical considerations related to respecting individuals' privacy rights. It emphasizes the importance of obtaining consent, providing transparency, and giving individuals control over their personal information.

- **Data lifecycle perspective —** Data protection primarily focuses on securing personal data during its storage, transmission, and processing stages, It encompasses measures like data encryption, access controls, and secure data handling practices.

Data privacy takes a comprehensive approach, considering the entire lifecycle of personal data. It involves aspects such as the collection, purpose specification, retention, sharing, and disposal of personal data. It emphasizes the need for privacy-by-design principles and obtaining explicit consent from individuals for data processing activities.

6 Best Practices to Ensure the Privacy of Your Data

To ensure the security and privacy of your data in the digital age, it's essential to implement robust practices that mitigate the risk of data breaches and unauthorized access. By following these practices, you can significantly enhance your data protection measures. Let's explore six best practices in more detail:

1. Data Discovery

Data discovery is a crucial first step in protecting your personal data. By gaining a clear understanding of the information you generate and share, you can take proactive steps to enhance your data privacy. Here are some tips to help you effectively conduct data discovery as an individual:

- **Review Your Digital Footprint** — Take stock of the personal information you share online, such as on social media platforms, websites, or online forums. Understand which data is publicly available and consider adjusting your privacy settings to limit access to sensitive details.
- **Audit Your Devices** — Access the personal data stored on your devices, including smartphones, laptops, and tablets. Identify the types of information stored and organize it in a way that makes it easier to manage and protect.

- **Scrutinize App Permissions** — **Review** the permissions granted to the apps on your devices. Be cautious about granting excessive access to your personal information and only provide necessary permissions for apps to function properly.
- **Secure Your Email** — Regularly review your email accounts for any unnecessary personal information stored in emails or attachments. Delete any sensitive information that is no longer needed, and consider encrypting important emails to add an extra layer of protection.

2. Use Automated Discovery Tools

Automated discovery tools help you identify and classify sensitive information, allowing you to implement appropriate security measures based on the level of security.

- **Research and Select Reliable Tools** — Search for tools that offer robust scanning capabilities and comprehensive data classification features. Consider factors such as user reviews, customer support, and compatibility with your existing systems. Some of the best ones are the Varonis Data Classification Engine, Spirion, and Microsoft Azure Information Protection.
- **Scan Your Devices and Systems** — Once you've chosen an automated discovery tool, initiate scans across your devices and systems to identify personal data. These tools can search through files, folders, databases, and even cloud storage to locate sensitive information such as personally identifiable information (PII) or financial data.
- **Classify and Categorize Data** — After the scanning process, the automated discovery tool will help you categorize the identified data based on sensitivity levels. This classification allows you to prioritize your data protection efforts and apply appropriate security controls accordingly.

3. Focus on Monitoring and Review Alerts

By establishing robust monitoring practices and implementing alert mechanisms, you can proactively protect your data and minimize the impact of security breaches.

- **Implement Network Monitoring Solutions** — Deploy network monitoring tools that continuously monitor network traffic and system activities. These tools help detect abnormal behaviors, such as unauthorized access attempts or suspicious network traffic patterns. Some of the best tools for this purpose are Wireshark, SolarWinds Network Performance Monitor, and Zabbix.
- **Use Intrusion Detection Systems (IDS)** — Implement intrusion detection systems to identify and respond to potential security incidents. IDS solutions monitor network traffic and systems for signs of malicious activities, such as unauthorized access or malware infections. Configure the system to generate alerts when suspicious activities are detected and investigate them promptly. Some of the best tools for this purpose are Snort and Suricata.
- **Set Up Log Monitoring** — Enable logging features on your devices and systems and regularly monitor log files for any unusual activities. Log monitoring can provide

valuable insights into potential security breaches, system vulnerabilities, or unauthorized access attempts. Consider using log analysis tools to automate log monitoring and detect security events efficiently.

4. Use Antivirus and Anti-Malware Software

If the above-mentioned practices and tools are too complex for you as an individual user, the least you could do is rely on antivirus and anti-malware software. But you have to use it properly with the intention of protecting your data!

- **Install Reputable Security Software** — The solutions you choose must prevent unauthorized access to your sensitive information, ensuring data privacy.
- **Keep Your Software Up to Date** — Regularly updating your antivirus and anti-malware software ensures that you have the latest virus definitions and security patches. This helps protect your data from evolving malware threats, preserving data privacy. If you still have downloaded malware that endangers your data, click here to see how to remove it.
- **Enable Real-Time Scanning** — Real-time scanning actively monitors files and programs for malware in real-time. By enabling this feature in your security software, you can detect and block potential threats before they compromise your data privacy.
- **Schedule Regular System Scans** — Regular system scans help identify and remove malware that may have infiltrated your system. By conducting routine scans, you can detect and eliminate any malicious software that poses a risk to your data privacy.

5. Use Storage with Built-In Data Protection

Storage solutions with built-in data protection features provide additional layers of security, encryption, and safeguards to keep your data safe from unauthorized access or data breaches.

- **Choose Reliable Storage Solutions** — Opt for storage devices or services that offer built-in data protection features. Look for reputable brands or cloud storage providers known for their robust security measures and data encryptions. Some of the best options include Dropbox, Google Drive, Box, and iCloud.
- **Use Strong Passwords or Encryption Keys** — Set strong and unique passwords or encryption keys for your storage devices or cloud storage accounts. Avoid using easily guessable passwords and consider using a password manager to securely manage your credentials.
- **Enable Data Encryption** — If your storage solution offers encryption capabilities, enable them. Encryption converts your data into an unreadable format, protecting it from unauthorized access.

6. Implement Authentication and Authorization Practices

Authentication and authorization are fundamental aspects of data privacy and security. These mechanisms ensure that only authorized individuals can access and interact with sensitive data.

- **Enable Multi-Factor Authentication** — Enable MFA whenever possible. It requires additional verification, such as a temporary code sent to your mobile device, along with your password, making it significantly harder for unauthorized individuals to access your accounts.
- **Be Mindful of Sharing and Access Controls** — Be cautious about sharing your data or granting access permissions. Only provide access to trusted individuals or applications and review and revoke access permissions as needed to maintain data privacy. This is especially important when using eCommerce platforms. Only provide the information that's necessary to complete the transaction and avoid sharing excessive personal details with the support. If something is not relevant to the purchase or compromises your privacy, don't share it!
- **Implement Role-Based Access Control (RBAC)** — RBAC manages access permissions within your systems or applications. Assign roles with specific privileges to users based on their responsibilities and limit access to sensitive data to only those who require it.

How To Protect Online Business Transactions

The introduction of the World Wide Web about three decades ago ushered in the Information Age. It's 2022 and information has become a commodity that we can quickly and widely disseminate through our internet-enabled smartphones and computers. Over the years, we've come to rely on the internet for a range of day-to-day tasks such as shopping, learning, entertainment, and work. In the process, we generate a lot of sensitive information.

All the information we generate has value and with the world now firmly in the digital age, data has become a currency. As a result, there are a lot of scary individuals coming after the valuable information that we inadvertently put out there. It's not only individuals that are at risk of a cyberattack but also companies that store, process, or transmit personal data. Symantec's 2019 Internet Security Threat Report shows that attacks on businesses are up 12 percent.

Start-ups are Particularly Vulnerable

Every company, whether it's a tech giant or a small business, has vulnerabilities that hackers could exploit. However, young start-ups are more vulnerable to cyberattacks and data breaches compared to larger, established businesses. And given their propensity for embracing new technologies and the lack of robust cybersecurity infrastructure, this is no coincidence. Reports show that 43% of all cyberattacks target small and medium-sized businesses, including start-ups.

Do not make the mistake of assuming that your start-up, due to its size, is not a target for cybercriminals. Start-ups have customer records that may contain information such as credit card numbers, medical records, social security numbers, and personal information (including

birthdate, address, identification number, etc.). Hackers will try to steal this information from your organization, often with potentially devastating consequences.

Types of Cyber Risks Facing Start-Ups In 2020

Cyberattacks targeting young start-ups and other small businesses will continue to increase unless business owners do something about it. As a start-up entrepreneur, the most important thing you need to do is acknowledge the threat.

By truly understanding that you are a target of cybercrime, you have made an important first step toward better cybersecurity protection. Speaking of which, here are the types of cyber risks facing start-ups in 2022.

Phishing

Constituting more than 90 percent of all attacks, phishing is the most common form of cyberattack targeting businesses and individuals. Phishing is a form of social engineering tactic designed to trick the victim into sharing sensitive information such as usernames, passwords, and other sensitive information. A phishing attack often takes the form of an email directing you to a fraudulent website that is designed to look like a legitimate site.

Ransomware

Ransomware is a form of malware that is designed to encrypt or lock down a company's mission-critical data until the victim pays some cash. Hackers prefer payments in the form of cryptocurrencies such as Bitcoin because they are harder to trace compared to cash or online transactions.

Data Leakage

A data leak is an unauthorized transfer of confidential information from within a business to an external environment. In such an incident, sensitive or classified information seeps to the outside world. A data leak can be intentional or unintentional. The information exposed through a data leak may include contact info, user credentials, payment card information, etc. Data leakage is common in start-ups.

Ways to Secure Start-Ups

Any business that relies on technology to store and manage customer information is a potential target for a cyberattack. Despite their small size, start-ups are not exempt from cyberattacks and data breaches. In fact, start-ups and other small businesses are a favorite target for cybercriminals due to their lack of strong cybersecurity systems. So, what can start-up entrepreneurs do to ensure that their businesses do not fall victim to these attacks?

Assess Risks and Vulnerabilities: A risk assessment helps you identify where your company is most vulnerable and take the necessary steps to fix these weaknesses. By

assessing risks and vulnerabilities in your system, you can effectively defend against the wide range of cybersecurity threats out there. Start by auditing the most valuable data and information in your possession including customer information, intellectual property, employee data, and other records.

Install Antivirus Software: Install reliable anti-virus and anti-malware software on every laptop or desktop in your company to protect your system from malicious code. Antivirus refers to a set of programs designed to protect your system from malicious programs. Malware may include viruses, trojans, spyware, etc. Antivirus software can also remove glitches and unwanted programs to improve performance.

Secure your Wi-Fi network: A virtual private network (VPN) provides online privacy and anonymity by creating a private network in a public internet connection. Many large organizations use this service to protect sensitive data, and you can do the same for your start-up. A VPN on your router at the office will ensure that your office Wi-Fi network is encrypted, secure, and hidden. It provides a private connection and protects your office network against online threats.

Hire IT Services: Hackers often target start-ups due to their lack of a robust cybersecurity infrastructure as well as experience in IT security. As a start-up entrepreneur lacking the resources to invest in a strong cybersecurity system for your company, hiring IT services from a company that specializes in IT security gives you a chance to focus on expanding your business.

Backup Your Data: Backing up your data in the cloud and locally is important for a variety of reasons. Having a reliable backup is essential to start-up cybersecurity. A good backup strategy ensures that mission-critical data is available for restoration in the event of a cyberattack, say a ransomware attack that encrypts your files. You can use your backup to restore your data from the cloud and start afresh in the event of such an attack.

Limit Access to Company Information: The amount of information your vendors have access to can pose a cybersecurity risk, and most start-up entrepreneurs are not aware of this. Check your vendors' cybersecurity controls as part of the vetting and onboarding process. Items to look at include how they store data, compliance with data protection regulations, access control, etc.

Employee Training: Employees are often said to be a weak link in a company's cybersecurity system. Most data breaches can be traced back to negligent employees, contractors, and third parties. Employee actions such as continued use of default passwords, loss of company smartphones, etc. can easily result in a data breach. Employee training on cybersecurity best practices can greatly reduce the risk of an attack.

Build a Security Centric Culture: Developing a security-centric culture within your start-up, it's one of the most effective ways to secure your start-up's data. Encrypting mobile devices used for work purposes, blocking access to websites that pose cybersecurity risks, and making your employees use strong passwords reduce the risk of attacks and make it clear that you are serious about cybersecurity.

Cybersecurity is increasingly becoming a concern for businesses all over the world. Due to their propensity for embracing new technologies and the lack of robust cybersecurity infrastructure, start-ups are a favorite target for cybercriminals.

For a start-up entrepreneur, defending against cyberattacks and data breaches is always going to be a constant struggle. However, using IT security tools such as antivirus and VPN, backing up your data, and applying common sense policies can help you mitigate these risks.

RESPONSIBILITIES OF CYBERSECURITY PROFESSIONALS

Cybersecurity professionals play a crucial role in protecting digital systems, networks, and data from cyber threats. Their responsibilities vary depending on their specific roles, but they generally focus on preventing, detecting, responding to, and mitigating security risks.

Threat Protection: Before cyber criminals can access your information, cyber professionals create a defense system that blocks and protects any potential threats. As technology advances, so do the methods that hackers employ to reach your private data; consequently, cyber professionals have to always improve and fortify their defenses although they may not be impenetrable.

Threat Detention: When the defenses have been breached, your prized information is no longer safe, but how will anyone know that it happened? In addition to protecting sensitive information, another important aspect of cybersecurity is knowing when an attack has taken place. Cyber professionals stay vigilant, watching out for any breaches and analyzing the security system for any malicious activity.

Incident Response Plan and Threat Eradication: In the case of a data breach, cyber professionals also are responsible for what to do after they've detected a threat. An **incident response plan** is put in place for when a breach ever does occur, allowing professionals to not waste any time in eradicating the troublemaker.

HOW DO WEB DEVELOPERS AND CYBERSECURITY PROFESSIONALS WORK TOGETHER?

Web developers and cybersecurity professionals don't often work together; however, they are two disciplines that do intersect occasionally.

Coding

Because all websites and apps are vulnerable to cyber-attacks, cybersecurity professionals need to know *some* coding. By knowing HTML and CSS, cybersecurity professionals can get into the minds of hackers that will try to look for the flaws in the programming. Thinking like a

hacker is one of the most important skills that a cybersecurity professional can have in their arsenal. The most useful languages for cyber professionals to learn include:

- Java
- C and C++
- Python
- SQL
- PHP

Threats and Cyberattacks

On the other hand, web developers write code for websites and apps and should have an idea of security issues that could arise. By prioritizing cybersecurity when creating the website, web development teams are helping the cybersecurity team in doing their job and preventing the theft of user and company information.

Front end development codes for user-based protections and back-end development protects the server and information, thanks to encryption. Both depend greatly on vigilance, maintenance, and high-quality coding. The following tips come in handy for coding security measures:

- Include strict user input
- Avoid hidden files and typical XSS mistakes
- Add Captcha
- Use a strong Content Security Policy
- Disable iframe embedding

- Compartmentalize your app

Despite not directly working alongside one another, **cybersecurity and web developers have quite a bit of overlap**; web developers understanding and keeping cybersecurity in mind and vice versa enhances the creation process, final product, and maintenance.

In the end, users benefit greatly from a professional with skills in both cybersecurity and web development and these disciplines are essential to the freedom and safety of the internet.

STEPS TO ENHANCE CYBERSECURITY IN WEB DEVELOPMENT

1. **Conduct Regular Security Audits:** Identify vulnerabilities and immediately patch any found issues.
2. **Implement Strong Encryption:** Use SSL certificates and encrypt sensitive data.
3. **Educate Your Team:** Regular training on the latest security threats and prevention techniques.
4. **Utilize Secure Coding Practices:** Follow guidelines and advisories for secure coding.
5. **Deploy Web Application Firewalls (WAF):** Protect against common vulnerabilities like SQL injection.

TAKEAWAYS

Cyber security threats pose a significant risk to our digital life, impacting the key infrastructure of our society. It is crucial to understand these threats, adapt to new cyber threat actors and landscapes, and invest in our cyber skills, talent, and innovation.

Individuals and companies should take precautions to safeguard themselves, such as adopting strong passwords, updating software, and watching for unusual activities. Furthermore, having a response strategy in place is critical in a cyberattack.

Prey can help you keep connected devices and your important information safe and secure by providing device monitoring, anti-theft protection to prevent identity theft and to prevent data breaches, and remote wiping.

Staying up-to-date on the newest risks and taking precautions to safeguard oneself and others can aid in the prevention of cyber threats and guarantee a safer digital environment.

CHAPTER

3

WEB DEVELOPMENT

Web development refers to the creating, building, and maintaining of websites. It includes aspects such as web design, web publishing, web programming, and database management. It is the creation of an application that works over the internet i.e. websites. Web development is typically broken down into **3 core areas: Frontend**, **Backend**, **Full Stack** Development.

FRONT-END DEVELOPMENT

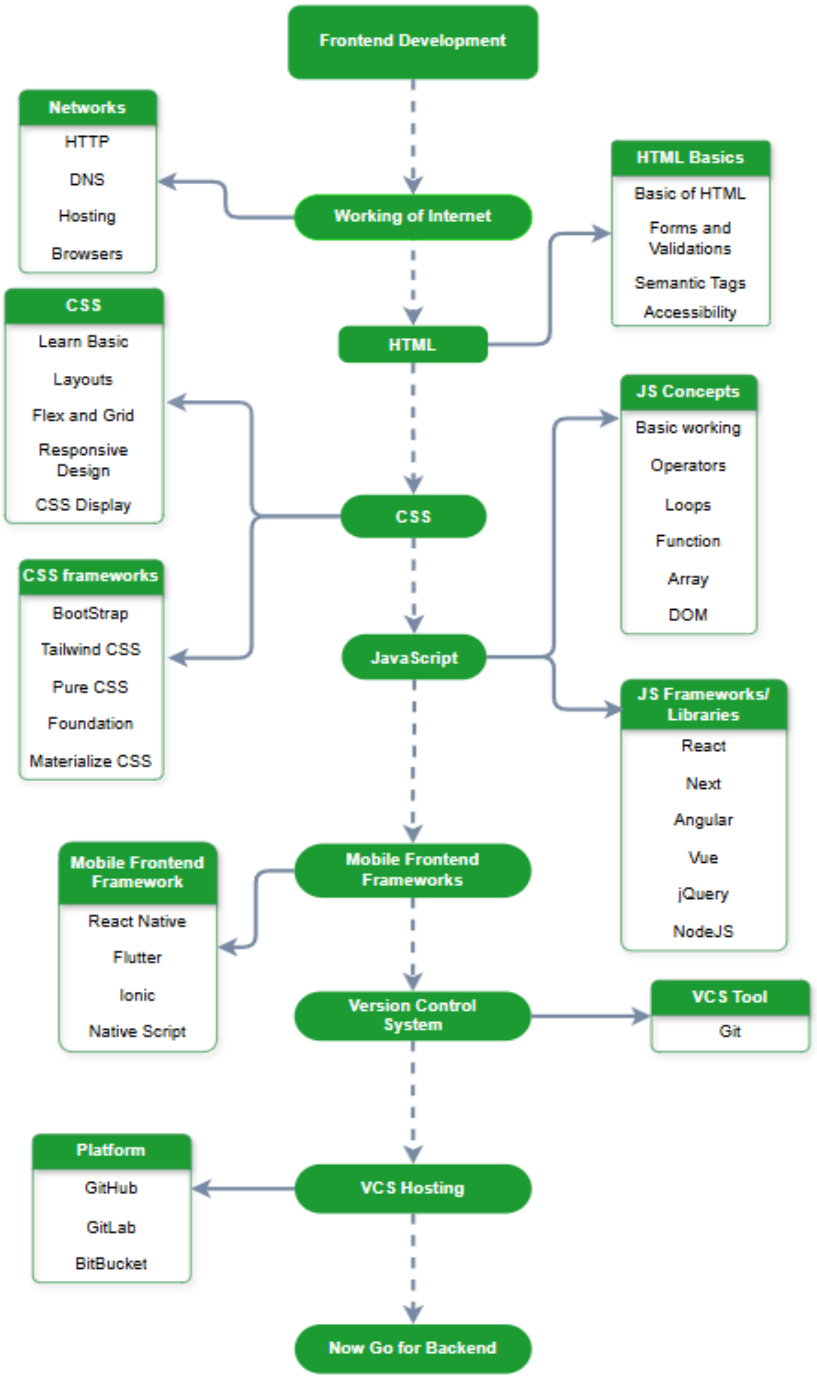
The Face of the Web. The part of a website where the user interacts directly is termed as front end. This involves designing the structure, layout, and behavior of the website It is also referred to as the 'client side' of the application.

Frontend Technologies

- **HTML:** HTML stands for Hypertext Markup Language. It is used to design the front-end portion of web pages using markup language. It acts as a skeleton for a website since it is used to make the structure of a website.
- **CSS:** Cascading Style Sheets fondly referred to as CSS is a simply designed language intended to simplify the process of making web pages presentable. It is used to style our website.
- **JavaScript:** JavaScript is a scripting language used to provide a dynamic behavior to our website.
- **Bootstrap:** Bootstrap is a free and open-source tool collection for creating responsive websites and web applications. It is the most popular CSS framework for developing responsive, mobile-first websites. Nowadays, the websites are perfect for all browsers (IE, Firefox, and Chrome) and for all sizes of screens (Desktop, Tablets, Phablets, and Phones).
    - Bootstrap 4
    - Bootstrap 5

Frontend Frameworks/Libraries

- **React.js**: A popular JavaScript library for building dynamic, component-based user interfaces.
- **Angular**: A full-fledged framework for building single-page applications (SPAs), with features like two-way data binding and dependency injection.
- **Vue.js**: A progressive JavaScript framework that is flexible and can be used for building both simple and complex user interfaces.



The Front-End Developer

A front-end developer creates websites and applications using web languages such as HTML, CSS, and JavaScript that allow users to access and interact with the site or app. When you visit a website, the design elements you see were created by a front-end developer.

Front-end developers create user interfaces (UI). UI is the graphical layout of an application that determines what each part of a site or application does and how it will look. "I've always found crafting polished user interactions that surprise and delight users to be the most rewarding and engaging task," says Mari Batilando, a software engineer at Meta. "In order to do this, you need to both have an eye for detail and a rock-solid understanding of the platform."

Front-End Developer Job Description

If someone wanted to build a website, they might hire a front-end developer to create the site's layout. The front-end developer determines where to place images, what the navigation should look like, and how to present the site. Much of their work involves ensuring the appearance and layout of the site or application is easy to navigate and intuitive for the user.

Average Salary for Front-End Developers

Glassdoor reports an average yearly pay of $90,345 for front-end developers in the US. This figure includes an average annual salary of $83,119 and a reported additional pay of $7,226. Additional pay insights may include profit-sharing, commissions, or bonuses. Factors like education level, experience, and certifications may affect salary range.

How To Become a Front-End Developer

A career as a front-end web developer can flex your creativity and problem-solving skills. As a field that is constantly evolving to incorporate new technology, front-end development can reward those who like to learn new things and face challenges. The next few sections outline some of the most prominent skills for front-end developers.

HTML, CSS, and JavaScript

These three languages are essential to anyone who wants to work in front-end development.

- **HTML (hypertext markup language):** The basic building block needed to develop websites; a language that allows you to make notes in digital documents that are different from regular text.
- **CSS (cascading style sheets):** The language used to create the layout, color, and overall style of the pages you create with HTML.
- **JavaScript:** The programming language that determines what the page will do.

  HTML, CSS, and JavaScript work together to determine the look and functionality of the page.

BACKEND DEVELOPMENT

Back-end development means working on server-side software, which focuses on everything you can't see on a website. Back-end developers ensure the website performs correctly, focusing on databases, back-end logic, application programming interface (APIs), architecture, and servers. They use code that helps browsers communicate with databases, store, understand, and delete data.

On a team, back-end developers collaborate with front-end developers, product managers, principal architects, and website testers to build the structure of a website or mobile app. Back-end developers must be familiar with many kinds of tools and frameworks, including languages such as Python, Java, and Ruby. They make sure the back-end performs quickly and responsively to front-end user requests.

The Back-End Developer

A back-end developer is a type of programmer who specializes in creating and maintaining the server-side logic, databases, and other components of a web application or software. While front-end developers focus on what users interact with directly, such as the visual elements and user interface, back-end developers work behind the scenes to ensure that the application runs smoothly and efficiently.

Back-end developers typically work with programming languages such as Python, Java, Ruby, PHP, or Node.js, as well as frameworks and tools like Django, Spring Boot, Ruby on Rails, Laravel, and Express.js. They often collaborate closely with front-end developers, designers, and other members of the development team to ensure that the entire application functions seamlessly and meets the needs of users.

Back-End Developer Tasks and Responsibilities

Back-end developers are required to have technical expertise, analytical thinking, and excellent collaboration skills. As a back-end web developer, you should be able to work independently to design the web infrastructure.

Here's what many back-end developers do on a day-to-day basis:

- **Build and maintain websites:** A back-end developer's main responsibility is to use various tools, frameworks, and languages to determine how best to develop intuitive, user-friendly prototypes and turn them into websites. This requires an understanding of cross-platform functionality and compatibility.
- **Write high-quality code:** To produce sustainable web applications, developers must write clean and easily maintainable code.
- **Perform quality assurance (QA) testing:** Create and oversee testing schedules to optimize user interface and experience, ensuring optimal display on various browsers and devices.

- **Assess efficiency and speed:** Once a website is up and running, and during updates and edits, developers need to assess its performance and scalability, adjusting code as necessary.
- **Troubleshoot and debug:** Be able to troubleshoot issues and resolve them, while communicating them to project managers, stakeholders, and QA teams.
- **Train and support:** Maintain workflows with client teams to ensure ongoing support, along with leading training and mentorship for junior developers.

What Tools Do Back-End Developers Use?

Web developers use a variety of tools to develop, test, and maintain web applications. Some common tools for back-end developers include:

- **Programming languages:** Python, PHP**,** JavaScript**,** Ruby**,** Java**,** C#
- **Frameworks:** Laravel**,** Django**,** Spring**,** Ruby on Rails**,** Meteor**,** Node.js
- **Databases:** MongoDB**,** MySQL**,** Oracle
- **Servers:** Apache**,** NGINX**,** Lighttpd**,** Microsoft IIS

How To Become a Back-End Developer?

There are many paths you can take to become a web developer. Whether you are a recent graduate or hoping to switch careers, it is important to assess what transferable skills you already have and consider building the new skills needed to pursue a back-end developer role.

Back-End Developer Technical Skills

As a back-end developer, there are certain technical skills you will need to learn to navigate developing the back-end of the web or mobile application.

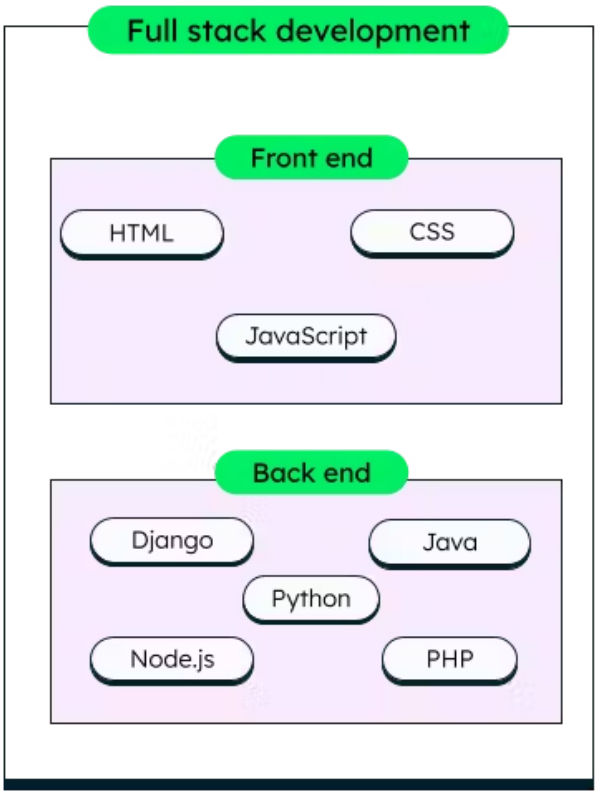
- **Programming languages:** Any back-end developer needs to be well-versed in back-end programming languages such as Python, Java, and PHP. These make the website function when used alongside databases, frameworks, and servers. Python is one of the most popular programming languages because it is compatible with artificial intelligence (AI) and machine learning, and works well for writing clear and logical code. Basic knowledge of front-end languages HTML, CSS, and JavaScript is a bonus.
- **Frameworks:** Frameworks are the libraries of back-end programming languages that help to build the server configuration. They tend to be linked with programming languages, so if you are familiar with Python, you'll also know Flask, Django, or another Python-based framework, and so on.

- **Databases and servers:** You'll need to understand how to stack and recover data from databases, as back-end programming controls access to this information, including storage and recovery. MongoDB and MySQL are popular database programs. The database stores and organizes the client's data so that it can be easily arranged and recovered, just like you might use cloud storage for your photos. This database then runs on a server that provides data upon request.
- **Application Program Interface (API):** An API is a series of definitions and rules for developing application software. In addition to internet browser websites, companies often want a mobile app for iOS or Android. Knowledge of application-building languages like JavaScript will expand your job opportunities.
- **Accessibility and security clearance:** You should develop knowledge of network protocols and web security. Knowing how to secure databases and servers will be critical to your success as a back-end developer.

FULL STACK DEVELOPMENT

Full stack development encompasses the complete process of application software development, including both the front-end and back-end development. The front end consists of the user interface (or UI), and the back end handles the business logic and application workflows that run behind the scenes.

Full stack developers possess the skills and knowledge to work across the entire technology stack, enabling seamless user experiences and developing robust backends.



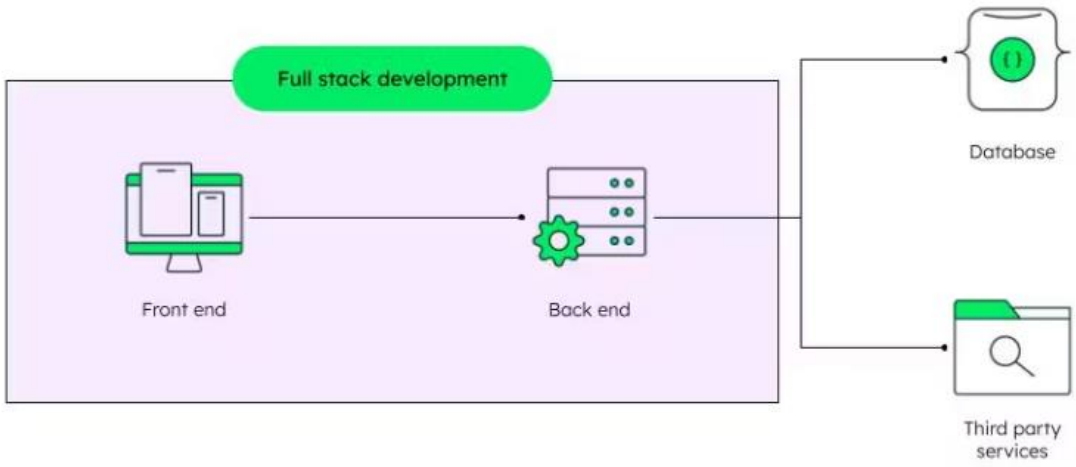
Full stack developers also stay current with the ever-evolving front-end landscape, constantly exploring and mastering the latest technologies and frameworks. Whether it's harnessing the power of React to build dynamic and interactive user interfaces, leveraging Angular's robust features for complex web applications, or embracing the simplicity and flexibility of Vue.js, full stack web developers are well-versed in a wide range of tools.

Full stack developers understand the importance of creating responsive designs that seamlessly adapt to various devices and screen sizes. They employ CSS frameworks like Bootstrap or Foundation to streamline the development process and ensure a consistent and visually appealing user interface across different platforms.

With their expertise in front-end optimization techniques, full stack web developers strive to deliver fast-loading and high-performing web experiences.

But front-end development isn't just about making things look pretty. Full stack developers also focus on usability and accessibility, ensuring the UI is intuitive, easy to navigate, and inclusive to all users, including those with disabilities.

Moreover, most full stack developers are proficient in integrating the front end with back-end APIs and services. They utilize modern JavaScript features that are built into modern browsers to seamlessly communicate with server-side components, retrieve and manipulate data, and update the user interface in real time. Additionally, they leverage popular JavaScript libraries and frameworks like React, Angular, or Vue.js to build dynamic, interactive user interfaces. This enables them to create data-driven web applications that provide a smooth and interactive user experience.



Full Stack Technologies:

- **MERN Stack:** MongoDB, Express.js, React, Node.js
- **MEAN Stack:** MongoDB, Express.js, Angular, Node.js
- **JAMstack:** JavaScript, APIs, Markup
- **Django Stack:** Django, MySQL/PostgreSQL, HTML/CSS/JavaScript
- **Spring Boot Stack:** Spring Boot, MySQL/PostgreSQL, Java

- **LAMP Stack:** Linux, Apache, MySQL, PHP
- **LEMP Stack:** Linux, Engine-X, MySQL, PHP

Full Stack Development in an End-To-End Workflow

To illustrate the power and versatility of full stack development, let's walk through a typical end-to-end workflow for building a web application.

1. Planning and Design

The full stack web developer starts by gathering requirements and understanding the project scope. They collaborate with stakeholders to define the desired features and technical specifications.

2. Front end development

With the design in place, the full stack developer shifts their focus to the front end. They begin by setting up the project structure and choosing the appropriate technologies, such as HTML, CSS, and JavaScript frameworks like React or Angular. They implement the user interface, ensuring responsiveness, accessibility, and adherence to design guidelines.

The developer creates reusable components, handles user interactions, and integrates with backend APIs to fetch and display data dynamically.

3. Back-end development

As the front end takes shape, the full stack developer is simultaneously building and integrating the back end. They select a server-side language like Node.js, Python, or Ruby and set up the necessary development environment. They design the database schema, choosing a suitable database management system like MySQL, PostgreSQL, or MongoDB.

The full stack web developer implements the server-side logic, creates RESTful APIs, and handles data retrieval, manipulation, and storage. They ensure the security and scalability of the back-end system, implementing authentication, authorization, and data validation mechanisms.

4. Testing and integration

Before launching into implementation and production, the full stack developer creates a comprehensive test plan. They define unit tests, integration tests, and end-to-end tests that address the desired functionality and application edge cases. By writing tests ahead of time, they establish clear goals and requirements for the required features.

Once the front-end and back-end components are in place, the full stack developer focuses on integrating them seamlessly. They establish communication between the client side and server side, ensuring smooth data flow and error handling. And the developer conducts thorough testing across a range of relevant devices, browsers, and scenarios to identify and fix any bugs or performance issues.

By following best practices like writing tests at the beginning of the project, the full stack developer ensures the application is reliable, maintainable, and meets the required specifications.

5. Deployment and maintenance

Once the web application is fully developed and tested, the full stack developer prepares it for deployment. They configure the necessary server environments, set up continuous integration and continuous deployment (CI/CD) pipelines, and ensure the application is optimized for production. They handle tasks such as database migrations, server provisioning, and scaling to accommodate user traffic.

After deployment, the developer monitors the application's performance, addresses any issues that arise, and provides ongoing maintenance and updates.

Throughout this end-to-end workflow, the full stack developer demonstrates their versatility and expertise in multiple technologies and domains. They seamlessly switch between front-end and back-end tasks, ensuring a cohesive and efficient development process. Their ability to handle the entire stack enables them to make informed decisions, optimize performance, and deliver a high-quality web application. What is a full stack developer and what do they do?

The Full-Stack Developer

Building the front end and the back end of a website require different skill sets. A full-stack developer is a developer or engineer who has expertise in both because they're involved with all aspects of the development process.

What Does a Full-Stack Developer Do?

Full-stack developers design and create websites and applications for various platforms. A full-stack developer’s job description might include the following:

- Develop and maintain web services and interfaces
- Contribute to front-end and back-end development processes
- Build new product features or APIs
- Perform tests, troubleshoot software, and fix bugs
- Collaborate with other departments on projects and sprints

The world of full-stack development is large, and many new and evolving technologies continually push the limits of what a full-stack developer can create. Staying on top of cutting-edge technology and techniques in the full-stack development field is one of the many exciting aspects of working in this role.

Full-Stack Developer Skills

Full-stack developers may be creative, graphically inclined, internet- and tech-savvy, and have excellent attention to detail. You'll want to learn the following skills to have a career in full-stack development:

- **Front-end development** is the process of creating the interface of a website. It entails coding details like drop-down menus, fonts, colors, and page layouts. Full-stack developers should also know how to work with front-end technologies like HTML, CSS, and scripting languages such as JavaScript to make websites and applications visually viable and appealing.
- **Back-end development** skills entail using back-end programming languages like Python, PHP, Ruby on Rails, and CakePHP and understanding how algorithms and business logic work.
- **Web design** includes using software such as Photoshop to create and design graphics and themes. It'll be helpful to familiarize yourself with basic UI (user interface) design principles to help you create navigational elements, backgrounds, and audio and video elements.
- **Database management** skills are required for full-stack developers, though they are also part of the skills needed for back-end web development. A full-stack developer should be able to design, understand, and manipulate database queries and web storage.

What Languages Do Full Stack Developers Use?

Full stack developers are free to use any set of languages that are compatible with each other and the overall application framework. JavaScript is a popular language often used by full stack developers as it's one of the very few languages that can be used both on the front end and back end. Companies will most likely hire a full stack developer for smaller or medium-size projects. Some popular languages are:

- **Front end:** HTML, CSS, JavaScript.
- **Back end:** Python, Java, R, Ruby, Node.js, PHP.

A popular and convenient practice is to use technology stacks like MEAN stack, MERN stack, Ruby on Rails, and LAMP for faster, more efficient development. These full-stack technologies also offer an easier learning curve.

The Benefits of Full Stack Developers

There are many advantages to hiring full stack developers for web application development, including:

1. Complete ownership and project understanding

Full stack developers have a comprehensive understanding of the entire web application, from the front end to the back end. This holistic knowledge allows them to take complete ownership of the project, making informed decisions and ensuring a cohesive development process. They can anticipate potential challenges, optimize performance, and create seamless integrations between different components of the application.

2. Time and cost savings/enhanced productivity

Hiring full stack developers can significantly reduce project time and costs while boosting productivity. Instead of engaging separate teams for front-end and back-end development, a full stack developer can handle both aspects simultaneously. This eliminates the need for extensive communication and coordination between different teams, streamlining the development process.

Full stack developers can work efficiently, leveraging their broad skill set to tackle multiple tasks concurrently.

3. Faster bug fixing and troubleshooting

When issues or bugs arise in a web application, a full stack developer can be invaluable. Their knowledge of the complete system allows them to quickly identify the root cause of the problem, whether it lies in the front end, back end, or the integration between the two. They can efficiently navigate through the codebase, pinpoint the issue, and implement a fix promptly.

4. Optimal division of work and resource allocation

With full stack developers on the team, project managers can achieve a better division of work and optimal resource allocation. Full stack developers can flexibly switch between front-end and back-end tasks based on project requirements and priorities, which allows for efficient resource utilization, prevents bottlenecks, and ensures a balanced workload distribution. Project managers can assign tasks strategically, leveraging the full stack developers' versatility to address critical areas of the project and maintain a steady pace of development.

ROLE OF AI IN WEB DEVELOPMENT

Artificial intelligence has revolutionized web development due to which the digital landscape is going under remarkable transformation. AI in Web Development isn’t just a buzzword. It’s a powerful tool which has the power to reshape, build and interact with websites. The given points will help you explain the role of AI in web development.

ChatBots

- AI-driven chatbots revolutionize customer support on websites.
- They provide instant, round-the-clock assistance to users.
- Chatbots handle multiple interactions simultaneously, ensuring efficiency.
- Improved user engagement and satisfaction result from their immediacy.
- Chatbots enhance website effectiveness through efficient customer service.

Personalized User Experiences

- AI's proficiency in analyzing vast data sets is invaluable in web development.
- AI in web development enables customization of content, product recommendations, and design based on user behaviors.
- Personalized experiences enhance user satisfaction and loyalty.
- AI-driven personalization boosts conversion rates significantly.

Enhanced Automation

- Artificial Intelligence and Machine Learning advancements have transformed web development, offering powerful tools and frameworks.
- Automation of repetitive tasks by AI and ML algorithms boosts developers' efficiency and productivity.
- Developers can allocate more time to critical aspects like user experience and creativity.
- AI and ML contribute to user experience enhancement on websites through personalized content and features.

Dynamic Content Generation

- AI algorithms streamline content creation, generating fresh and relevant material for websites.
- Web developers benefit from saved time and effort, as content creation becomes automated.
- The focus can shift to other critical tasks while ensuring a consistent flow of engaging content.
- AI-powered content generation enhances user experiences by providing timely and interesting material

SEO Strategies

- AI is revolutionizing SEO by providing deeper insights into search patterns and user behavior.

- Tools like Google's RankBrain leverage AI to interpret search queries and deliver relevant results.
- Web developers prioritize optimization of websites with AI, focusing on user intent and experience.
- AI-driven SEO enhances website visibility and relevance, ultimately improving user engagement and satisfaction.

Natural Language Processing

- NLP is a branch of AI focused on enabling computers to understand, interpret, and generate human language.
- NLP in web development includes chatbots, sentiment analysis, content recommendation systems, and language translation.
- Helps create personalized experiences, improve search functionality

Benefits Of AI In Web Development

Moreover, to give you a short overview, following are some of the benefits of AI in web development:

- Enhanced User Experience
- Improved Efficiency
- Enhanced SEO
- Personalization
- Predictive Analytics
- Security Enhancement
- Continuous Improvement
- Scalability
- Cost-Effectiveness
- Competitive Advantage

THE ETHICAL IMPLICATIONS OF AI ON WEB DEVELOPMENT

Artificial Intelligence (AI) is transforming web development by automating processes, improving user experience, and enabling intelligent decision-making. However, these advancements come with significant ethical concerns. Below are some key ethical implications of AI in web development:

1. Privacy and Data Security

Ethical Concern:

AI-powered websites and applications collect vast amounts of user data, raising concerns about privacy violations and unauthorized data usage.

Examples:

- AI-driven analytics tools track user behavior without explicit consent.
- AI chatbots may store sensitive conversations, posing risks of data breaches.
- AI personalization algorithms create detailed user profiles, sometimes without clear transparency.

Solutions & Best Practices:

- Implement strong encryption and security protocols.
- Follow regulations like **GDPR** (General Data Protection Regulation) and **CCPA** (California Consumer Privacy Act).
- Provide clear **privacy policies** and allow users to opt out of data collection.

2. Algorithmic Bias and Discrimination

Ethical Concern:

AI models may inherit biases from the data they are trained on, leading to **unfair discrimination** in web services.

Examples:

- AI-powered **resume screeners** may favor male applicants if trained on biased hiring data.
- **Facial recognition tools** on websites may perform poorly for people of certain ethnicities.
- **E-commerce AI recommendations** may promote certain brands over others unfairly.

Solutions & Best Practices:

- Use **diverse training datasets** to minimize bias.
- Conduct **regular audits** of AI models for fairness and transparency.
- Implement **ethical AI frameworks** like Google's AI Principles or IBM's AI Fairness 360.

3. Transparency and Explainability

Ethical Concern:

Many AI-driven web applications operate as "black boxes," making it difficult for users to understand how decisions are made.

Examples:

- AI chatbots make decisions without explaining why certain responses are chosen.
- Search engines rank results using **opaque AI algorithms**, potentially influencing information access.
- AI-driven financial approval systems may **reject loan applications** without clear reasoning.

Solutions & Best Practices:

- Implement **explainable AI (XAI)** to help users understand how AI makes decisions.
- Use **open-source AI models** or provide documentation on AI decision-making processes.
- Ensure **user control** by offering options to override AI-based recommendations.

4. Job Displacement and Automation

Ethical Concern:

AI-driven tools may replace human web developers, designers, and content creators, leading to **job losses** and economic disparities.

Examples:

- AI-powered **website builders** (e.g., Wix ADI, Framer AI) reduce the need for human developers.
- AI-generated **content writing tools** (e.g., ChatGPT, Jasper) replace human copywriters.
- AI-powered **UX design tools** automate layout and design processes, reducing demand for designers.

Solutions & Best Practices:

- Promote **AI as an assistive tool** rather than a replacement for human workers.
- Invest in **AI upskilling programs** for web developers and designers.
- Encourage **human-AI collaboration**, where AI handles repetitive tasks while humans focus on creativity and strategy.

5. Misinformation and Deepfakes

Ethical Concern:

AI can generate **fake content**, misleading users and contributing to misinformation.

Examples:

- AI-generated **fake reviews** influence consumer choices.
- AI deepfake technology is used to create **fake news and deceptive videos**.

- AI-generated **spam content** floods the web, reducing trust in online information.

Solutions & Best Practices:

- Implement **fact-checking mechanisms** for AI-generated content.
- Use AI **watermarking** or detection tools to identify manipulated media.
- Encourage **ethical AI content generation**, avoiding deceptive practices.

6. Ethical Use of AI in UX and Personalization

Ethical Concern:

AI-driven personalization can **manipulate user behavior**, leading to ethical concerns in web design.

Examples:

- AI-powered **"infinite scroll" designs** keep users addicted to social media.
- AI-driven **pricing algorithms** may offer different prices based on user profiles, leading to unfair treatment.
- AI-generated **dark patterns** trick users into subscriptions or purchases.

Solutions & Best Practices:

- Design **ethical UX** that prioritizes user well-being over engagement.
- Use **transparent pricing** algorithms that treat users fairly.
- Follow guidelines like the **Ethical Design Manifesto** to ensure AI-driven UX is user-friendly and non-manipulative.

Conclusion: Towards Ethical AI in Web Development

AI is a powerful tool in web development, but ethical considerations must be at the forefront of its implementation. Developers and businesses must ensure that AI is **transparent, fair, secure, and respectful of user rights**. By adopting responsible AI practices, the web can remain a **safe, inclusive, and ethical space** for all users.

CHAPTER

4

THE WEB DATABASE CONNECTIVITY

IMPORTANCE OF WEB AND DATABASE INTEGRATION

Web and database integration is crucial for modern applications as it allows websites to dynamically display and interact with data stored in a database, providing a seamless user experience, enabling personalized content, and facilitating efficient data management through centralized storage and retrieval, which is vital for features like user logins, product catalogs, and real-time updates across a website.

Key Benefits of Web and Database Integration:

Dynamic Content: Websites can display relevant information based on user input, location, or other variables by accessing data from the database in real-time, creating a more personalized experience.

Data Consistency: Centralized data storage in a database ensures all users are accessing the same information, eliminating inconsistencies across the website.

Scalability: Databases can handle large volumes of data efficiently, allowing websites to accommodate growing user bases and complex data requirements.

User Authentication: User logins and account management can be effectively implemented by storing user credentials securely in a database.

Data Analysis and Reporting: By integrating with a database, websites can access and analyze data to generate valuable insights for business decision-making.

Improved Efficiency: Developers can focus on building the website's functionality without worrying about managing data storage and retrieval, leading to faster development cycles.

Collaboration: Different teams within an organization can access and update data in a centralized location, fostering better collaboration.

Examples of Web and Database Integration:

E-commerce websites: Storing product details, user shopping carts, and order information in a database to enable seamless online shopping experiences.

Social media platforms: Managing user profiles, posts, and interactions through a database to display dynamic content on the website.

Content Management Systems (CMS): Storing website content like articles, images, and page layouts in a database, allowing easy updates and management.

BASIC CONCEPTS

Before we start to discuss Web database applications, we need to clarify a number of related terms and concepts.

Internet: It is a worldwide collection of interconnected computer networks, which belong to various organizations (e.g. educational, business and governments). It is not synonymous to the WWW. The services that are normally available on the Internet include email, real-time communication (e.g. conferencing and chat), news services, and facilities for accessing remote computers to send and receive documents.

The WWW or simply the Web: The WWW comprises software (e.g. Web servers and browsers) and data (e.g. Web sites). It simply represents a (huge) set of information resources and services that live on the Internet. Each Web site consists of a set of Web pages, which typically contain multimedia data (e.g. text, images, sound and video). In addition, a Web page can include hyperlinks to other Web pages which allow users (also called net surfers) to navigate through the Web of information pages.

Intranet: A Web site or group of sites which belongs to an organization and can only be accessed by members of that organization. Between the Internet and an intranet, there is an extra layer of software or hardware called a firewall. Its main function is to prevent unauthorized access to a private network (e.g. an intranet) from the Internet.

Extranet: An intranet which allows partial access by authorized users from outside the organization via the Internet.

HTTP (Hypertext Transfer Protocol): The standard protocol for transferring Web pages through the Internet. HTTP defines how clients (i.e. users) and servers (i.e. providers) should communicate.

HTML (Hypertext Markup Language): A simple yet powerful language that is commonly used to format documents which are to be published on the Web.

URL (Uniform Resource Locator): A string of alphanumeric characters that represents the location of a resource (e.g. a Web page) on the Internet and how that resource should be accessed.

There are two types of Web pages: static and dynamic.

Static: An HTML document stored in a file is a typical example of a static Web page. Its contents do not change unless the file itself is changed.

Dynamic: For a dynamic Web page, its contents are generated each time it is accessed. As a result, a dynamic Web page can respond to user input from the browser by, for example, returning data requested by the completion of a form or returning the result of a database query. A dynamic page can also be customized by and for each user. Once a user has specified some preferences when accessing a particular site or page, the information can be recorded and appropriate responses can be generated according to those preferences.

From the above, it can be seen that dynamic Web pages are much more powerful and versatile than static Web pages, and will be a focus for developing Web database applications. When the documents to be published are dynamic, such as those resulting from queries to databases, the appropriate hypertext needs to be generated by the servers. To achieve this, we must write scripts that perform conversions from different data formats into HTML 'on-the-fly'. These scripts also need to recognize and understand the queries performed by clients through HTML forms and the results generated by the DBMS.

In short, a Web database application normally interacts with an existing database, using the Web as a means of connection and having a Web browser or client program on the front end. Typically, such applications use HTML forms for collecting user input (from the client); CGI (Common Gateway Interface, to be discussed later) to check and transfer the data from the server; and a script or program which is or calls a database client to submit or retrieve data from the database. The diagram below gives a graphical illustration of such a scenario. More will be discussed in later parts of this chapter.

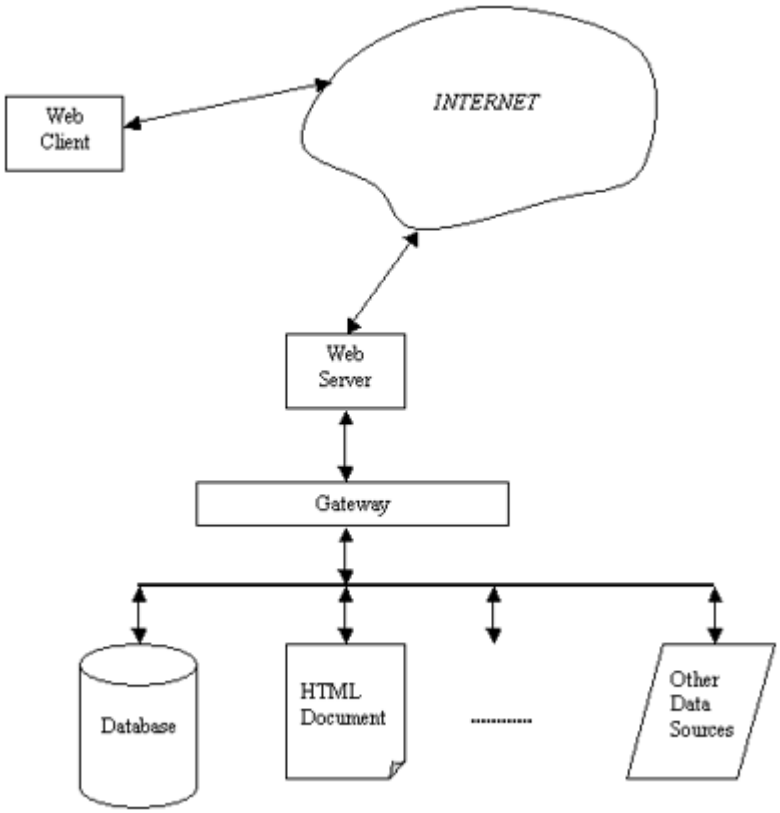
Web-Based Client-Server Applications

As mentioned earlier in the Introduction, Web-based database applications are a new type of client-server application. Some of the traditional client-server database techniques may still be adapted. However, because of the incorporation of the Web technology, there are important differences as set out in the following table.

Platform independence: Web clients are platform-independent and do not require modification to be run on different operating systems. Traditional database clients, on the other hand, require extensive porting efforts to support multiple platforms. This is arguably one of the most compelling reasons for building a Web-based client-server database application.

Interpreted applications: Web applications are written in interpreted languages (e.g. HTML and Java). This has an adverse effect on performance. In many applications, however, this is a price worth paying to gain the advantage of being platform independent. Time-critical applications may not be good candidates to be implemented on the Web.

No need for installation: Another benefit of Web database applications is that the need for installing special software is eliminated on the clients' side. It is pretty safe to assume that the clients have already had a Web browser installed, which is the only piece of software needed for the clients to run the applications.



Simple client: As a client needs just a browser to run a Web-based database application, the potential complications are minimized.

Common interface across applications: Again, because there is no need for specialized software, users have the benefit of using a browser for possibly different applications.

Limited GUI (Graphical User Interface): This is one area in which Web-based database applications may fall short. Highly customized application inter- faces and highly interactive clients may not translate well as Web applications. This is because of the HTML limitations. At the moment, HTML forms do not offer an extensive feature set. Although JavaScript language can extend the functionality of HTML-based applications, it is too complex, adds to downloading time, and degrades performance.

Integrate with other applications: Because of the benefit of being platform independent, different applications that adhere to the HTML standard can be integrated without many difficulties.

Non-persistent connection to database: Persistent database connections are highly efficient data channels between a client and the DBMS, and therefore, are ideal for database applications. However, Web-based applications do not have this benefit. A Web-based client maintains its connection to a database only as long as is necessary to retrieve the required data, and then releases it. Thus, Web application developers must address the added overhead for creating new database connections each time a client requires database access.

Apart from the above differences, there are some other important concerns for Web-based applications:

- **Reliability of the Internet:** At the moment, there are reliability problems with the Internet. It may break down; data may be lost on the net; large amounts of data traffic may slow down or even overwhelm the network system.
- **Security:** Security on the Internet is of great concern for any organization which has developed Web-based database applications. For example, the database may be broken into, or confidential data may be intercepted during transmission by unauthorized parties or even criminals.

At the present, a lot of research and development work is being carried out to address these concerns. There is no doubt that the potential problems can be overcome and over time, the Internet will be more reliable and more secure for connecting the world.

WEB DATABASE ARCHITECTURES

Components of a Database Application

Web database applications may be created using various approaches. However, there are a number of components that will form essential building blocks for such applications. In other words, a Web database application should comprise the following four layers (i.e. components):

Browser Layer

The browser is the client of a Web database application, and it has two major functions. First, it handles the layout and display of HTML documents. Second, it executes the client-side extension functionality such as Java, JavaScript, and ActiveX (a method to extend a browser's capabilities).

The three most popular browsers at the present are Mozilla Firefox (Firefox for short), Google Chrome and Microsoft Internet Explorer (IE).

All three browsers are graphical browsers. During the early days of the Web, a text-based browser, called Lynx, was popular. As loading graphics over the Internet can be a slow and time-consuming process, database performance may be affected. If an application requires a speedy client and does not need to display graphics, then the use of Lynx may be considered.

All browsers implement the HTML standard. The discussion of HTML is beyond this chapter, but you need to know that it is a language used to format data/documents to be displayed on the Web.

Browsers are also responsible for providing forms for the collection of user input, packaging the input, and sending it to the appropriate server for processing. For example, input can include registration for site access, guest books and requests for information. HTML, Java, JavaScript or ActiveX (for IE) may be used to implement forms.

Application Logic Layer

The application logic layer is the part of a Web database application with which a developer will spend the most time. It is responsible for:

- Collecting data for a query (e.g. a SQL statement).
- Preparing and sending the query to the database via the database connection layer.
- Retrieving the results from the connection layer.
- Formatting the data for display.

Most of the application's business rules and functionality will reside in this layer. Whereas the browser client displays data as well as forms for user input, the application logic component compiles the data to be displayed and processes user input as required. In other words, the application logic generates HTML that the browser renders. Also, it receives, processes and stores user input that the browser sends.

Depending on the implementation methods used for the database application, the application logic layer may have different security responsibilities. If the application uses HTML for the front end, the browser and server can handle data encryption (i.e. a security measure to ensure that data will not be able to be intercepted by unauthorized parties). If the application is a Java applet and uses Java for the front end, then it itself must be responsible for adopting transmission encryption.

Database Connection Layer

This is the component which actually links a database to the Web server. Be- cause manual Web database programming can be a daunting task, many current Web database building tools offer database connectivity solutions, and they are used to simplify the connection process.

The database connection layer provides a link between the application logic layer and the DBMS. Connection solutions come in many forms, such as DBMS net protocols, API (Application Programming Interface [see note below]) or class libraries, and programs that are themselves database clients. Some of these solutions resulted in tools being specifically designed for developing Web database applications. In Oracle, for example, there are native API libraries for connection and a number of tools, such as Web Publishing Assistant, for developing Oracle applications on the Web.

The connection layer within a Web database application must accomplish a number of goals. It has to provide access to the underlying database, and also needs to be easy to use, efficient, flexible, robust, reliable and secure. Different tools and methods fulfil these goals to different extents.

Note

An API consists of a set of interrelated subroutines that provide the functionality required to develop programs for a target operating environment. For example, Microsoft provides different APIs targeted at the construction of 16- and 32-bit Windows applications. An API would provide functions for all aspects of system activity, such as memory, file and process management. Specialized APIs are also supplied by software vendors to support the use of their products, such as database and network management systems.

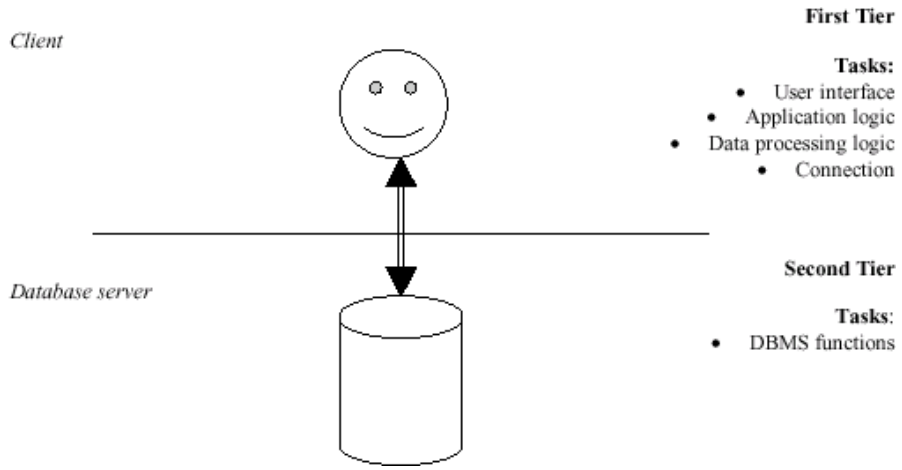
Database Layer

This is the place where the underlying database resides within the Web database application. As we have already learned, the database is responsible for storing, retrieving and updating data based on user requirements, and the DBMS can provide efficiency and security measures.

In many cases, when developing a Web database application, the underlying database has already been in existence. A major task, therefore, is to link the database to the Web (the connection layer) and to develop the application logic layer.

2-Tier Client-Server Architecture

Traditional client-server applications typically have a 2-tier architecture as il- lustrated in the figure below. The client (tier 1) is primarily responsible for the presentation of data to the user, and the server (tier 2) is responsible for supplying data services to the client. The client will handle user interfaces and main application logic, and the server will mainly provide access services to the underlying database.

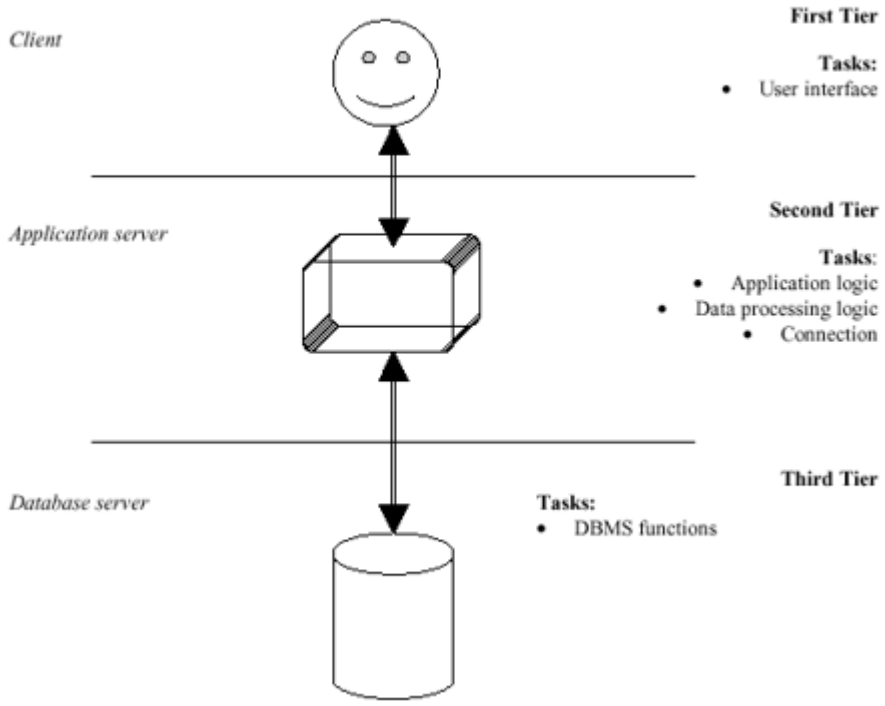


If such a 2-tier architecture is used to implement a Web database application, tier 1 will contain the browser layer, the application logic layer and the connection layer. Tier 2 accommodates the DBMS. This will inevitably result in a fat client.

3-Tier Client-Server Architecture

In order to satisfy requirements of increasingly complex distributed database applications, a 3-tier architecture was proposed to replace the 2-tier one. There are three tiers in this new architecture, each of which can potentially run on a different platform.

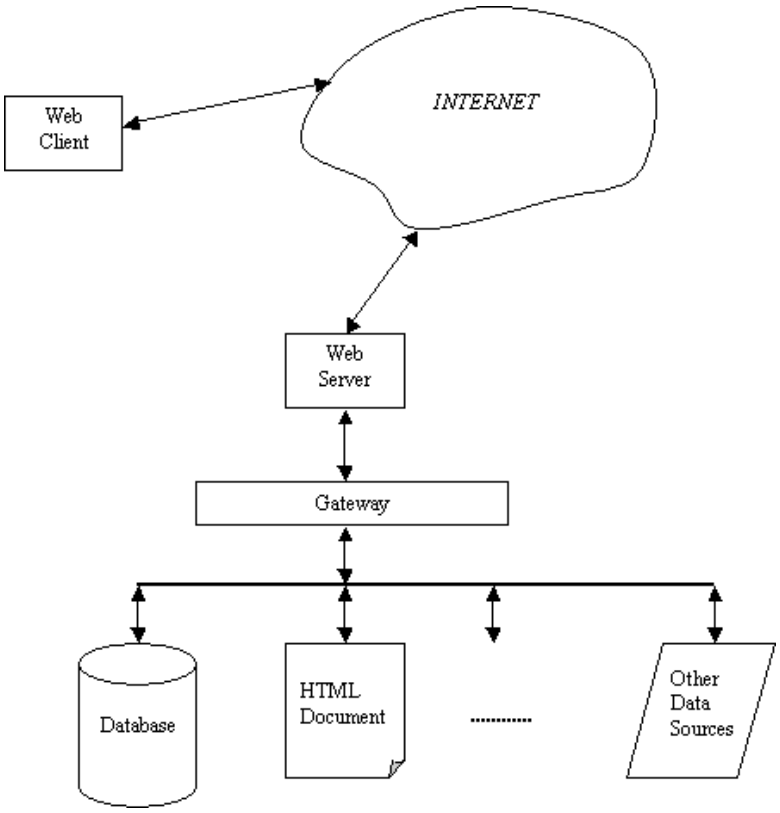
The first tier is the client, which contains user interfaces. The middle tier accommodates the application server, which provides application logic and data processing functions. The third tier contains the actual DBMS, which may run on a separate server called a database server.



The 3-tier architecture is more suitable for implementing a Web database ap- plication. The browser layer can reside in tier 1, together with a small part of the application logic layer. The middle tier implements the majority of the application logic as well as the connection layer. Tier 3 is for the DBMS.

Referring to the figure below, for example, it can be seen that the Web Client is in the first tier. The Web Server and Gateway are in the middle tier and they form the application server. The DBMS and possibly other data sources are in the third tier.

Having studied the Web database architectures, we should understand that the most important task in developing a Web database application is to build the database connection layer. In other words, we must know how to bridge the gap between the application logic layer and the database layer.

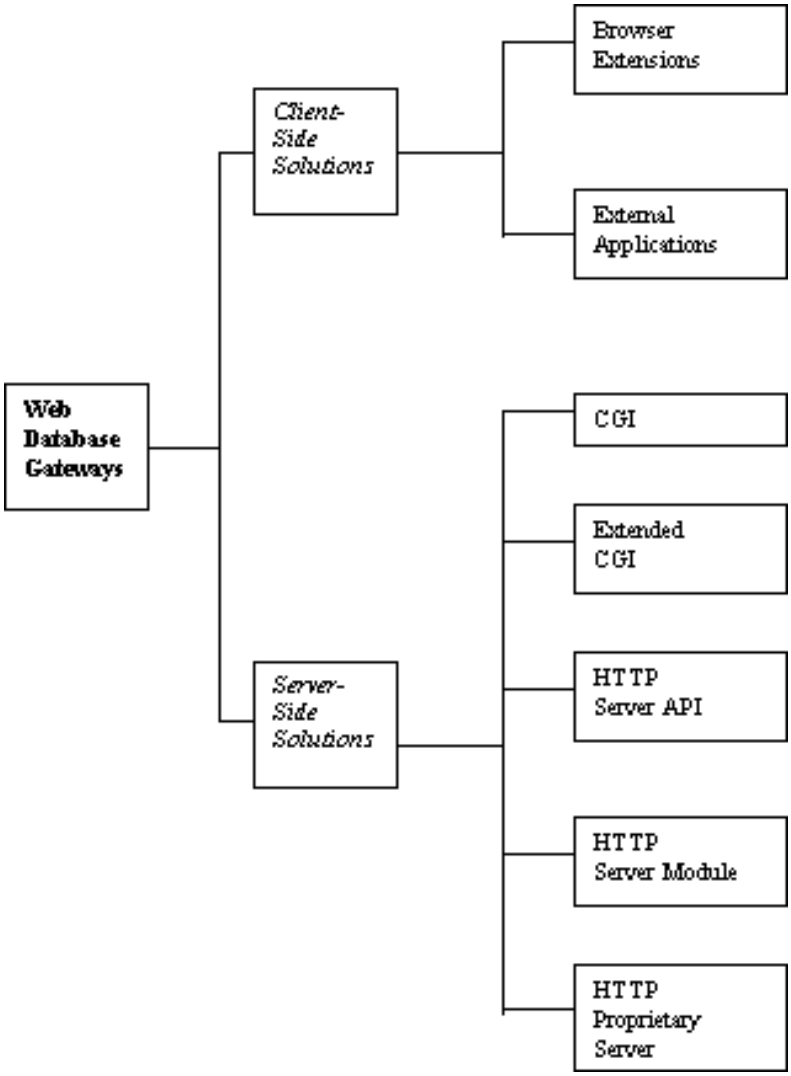


DATABASE GATEWAYS

A Web database gateway is a bridge between the Web and a DBMS, and its objective is to provide a Web-based application the ability to manipulate data stored in the database. Web database gateways link stateful systems (i.e. databases) with a stateless, connectionless protocol (i.e. HTTP). HTTP is a stateless protocol in the sense that each connection is closed once the server provides a response. Thus, a Web server will not normally keep any record about previous requests. This results in an important difference between a Web-based client-server application and a traditional client-server application:

- In a Web-based application, only one transaction can occur on a connection. In other words, the connection is created for a specific request from the client. Once the request has been satisfied, the connection is closed. Thus, every request involving access to the database will have to incur the overhead of making the connection.
- In a traditional application, multiple transactions can occur on the same connection. The overhead of making the connection will only occur once at the beginning of each database session.

There are a number of different ways to create Web database gateways. Gener- ally, they can be grouped into two categories: client-side solutions and server- side solutions, as illustrated below:



Client-Side Solutions

The client-side solutions include two types of approaches for connections: browser extensions and external applications.

Browser extensions are add-ons to the core Web browser that enhance and augment the browser's original functionality. Specific methods include plug-ins for Firefox, Chrome and IE, and ActiveX controls for IE. Also, all the three types of browsers (Firefox, Chrome and IE) support Java and JavaScript languages (i.e. Java applets and JavaScript can be used to extend browsers' capabilities).

External applications are helper applications or viewers. They are typically existing database clients that reside on the client machine and are launched by the Web browser in a particular Web application. Using external applications is a quick and easy way to bring legacy database applications online, but the resulting system is neither open nor portable. Legacy database clients do not take advantages of the platform independence and language independence available through many Web solutions. Legacy clients are

resistant to change, meaning that any modification to the client program must be propagated via costly manual installations throughout the user base.

Server-Side Solutions

Server-side solutions are more widely adopted than the client-side solutions. A main reason for this is that the Web database architecture requires the client to be as thin as possible. The Web server should not only host all the documents, but should also be responsible for dealing with all the requests from the client.

In general, the Web server should be responsible for the following:

- Listening for HTTP requests.
- Checking the validity of the request.
- Finding the requested resource.
- Requesting authentication if necessary.
- Delivering requested resource.
- Spawning programs if required.
- Passing variables to programs.
- Delivering output of programs to the requester.
- Displaying error message if necessary.

The client (browser) should be responsible for some of the following:

- Rendering HTML documents.
- Allowing users to navigate HTML links.
- Displaying image.
- Sending HTML form data to a URL.
- Interpreting Java applets.
- Executing plug-ins.
- Executing external helper applications.
- Interpreting JavaScript and other scripting language programs.
- Executing ActiveX controls in the case of IE.

In the following sections, we are going to discuss both client-side and server-side solutions in some detail.

CLIENT-SIDE WEB DATABASE PROGRAMMING

Major tasks of client-side Web database application programming include the creation of browser extensions and the incorporation of external applications. These types of gateways take advantage of the resources of the client machine, to aid server-side database access. Remember, however, it is advantageous to have a thin client. Thus, the scope of such programming on the client-side should be limited. A very large part of the database application should be on the server side.

Browser Extensions

Browser extensions can be created by incorporating script language interpreters to support script languages (e.g. JavaScript), bytecode interpreters to support Java, and dynamic object linkers to support various plug-ins.

1. JavaScript

JavaScript is a scripting language that allows programmers to create and customize applications on the Internet and intranets. On the client side, it can be used to perform simple data manipulation such as mathematical calculations and form validation. JavaScript code is normally sent as a part of an HTML document and is executed by the browser upon receipt (the browser must have the script language interpreter).

Note that JavaScript has little to do with Java language. JavaScript was originally called LiveScript, but it was changed to benefit from the excitement sur- rounding Java. The only relationship between JavaScript and Java is a gateway between the former and Java applets (Web applications written in Java).

JavaScript provides developers with a simple way to access certain properties and methods of Java applets on the same page, without having to understand or modify the Java source code of the applet.

Connection To Databases

As a database gateway, JavaScript on the client side does not offer much without the aid of a complementary approach such as Java, plug-ins and CGI (Common Gateway Interface, to be discussed later). For example:

- If a Java applet on a page of HTML has access to a database, a programmer can write JavaScript code using LiveConnect to manipulate the applet.
- If there is a form on the HTML document and if an action parameter for that form refers to a CGI program that has access to a database, a programmer can write JavaScript code to manipulate the data elements within the form and then submit it (i.e. submit a kind of request to a DBMS).

Performance

JavaScript can improve the performance of a Web database application if it is used for client-side state management. It can eliminate the need to transfer state data repeatedly between the browser and the Web server. Instead of sending an

HTTP request each time it updates an application state, it sends the state only once as the final action. However, there are some side effects resulting from this gain in performance. For example, it may result in the application becoming less robust if state management is completely on the client side. If the client accidentally or deliberately exits, the session state is lost.

2. Java

As mentioned earlier, Java applets can be manipulated by JavaScript functions to access databases. In general, Java applets can be downloaded into a browser and executed on the client side (the browser should have the bytecode interpreter). The connection to the database is made through appropriate APIs (Application Programming Interface, such as JDBC and ODBC). We will dis- cuss the details in the next section: Server-Side Web Database Programming.

3. ActiveX

ActiveX is a way to extend Microsoft IE's (Internet Explorer) capabilities. An ActiveX control is a component on the browser that adds functionality which cannot be obtained in HTML, such as access to a file on the client side, other applications, complex user interfaces, and additional hardware devices. ActiveX is similar to Microsoft OLE (Object Linking and Embedding), and ActiveX controls can be developed by any organization and individual. At the present, more than one thousand ActiveX controls, including controls for database access, are available for developers to incorporate into Web applications.

Connection To Databases

A number of commercial ActiveX controls offer database connectivity. Because ActiveX has abilities similar to OLE, it supports most or all the functionality available to any Windows program.

Performance

Like JavaScript, ActiveX can aid in minimizing network traffic. In many cases, this technique results in improved performance. ActiveX can also offer rich GUIs. The more flexible interface, executed entirely on the client side, makes operations more efficient for users.

4. Plug-ins

Plug-ins are Dynamic Link Libraries (DLL) that give browsers additional functionality. Plug-ins can be installed to run seamlessly inside the browser window, transparent to the user. They have full access to the client's resources, because they are simply programs that run in an intimate symbiosis with the Web browser.

To create a plug-in, the developer writes an application using the plug-in API and native calls. The code is then compiled as a DLL. Installing a plug-in is just a matter of copying the DLL into the directory where the browser looks for plug-ins. The next time that the browser is run, the MIME type(s) that the new plug-in supports will be opened with the plug-in. One plug-in may support multiple MIME types.

There are a number of important issues concerning plug-ins:

- Plug-ins incur installation requirements. Because they are native code, not packaged with the browser itself, plug-ins must be installed on the client machine.
- Plug-ins are platform dependent. Whenever a change is made, it must be made on all supported platforms.

Connection To Databases

Plug-ins can operate like any stand-alone applications on the client side. They can be used to create direct socket connections to databases via the DBMS net protocols (such as SQL *Net for Oracle). Plug-ins can also use JDBC, ODBC, OLE and any other methods to connect to databases.

Performance

Plug-ins are loaded on demand. When a user starts up a browser, the installed plug-ins are registered with the browser along with their supported MIME types, but the plug-ins themselves are not loaded. When a plug-in for a particular MIME type is requested, the code is then loaded into memory. Because plug- ins use native code, their executions are fast.

External Applications

External helper applications can be new or legacy database clients, or a terminal emulator. If there are existing traditional client-server database applications which reside on the same machine as the browser, then they can be launched by the browser and execute as usual.

This approach may be an appropriate interim solution for migrating from an existing client-server application to a purely Web-based one. It is straightforward to configure the browser to launch existing applications. It just involves the registration of a new MIME type and the associated application name. For organizations that cannot yet afford the time and funds needed to transfer existing database applications to the Web, launching legacy applications from the browser provides a first step that requires little work.

Maintenance Issues

Using the external applications approach, the existing database applications need not be changed. However, it means that all the maintenance burdens associated with traditional client-server applications will remain. Any change to the external application will require a very costly reinstallation on all client machines. Because this is not a pure Web-based solution, many advantages offered by Web-based applications cannot be realized.

Performance

Traditional client-server database applications usually offer good performance. They do not incur the overhead of requiring repeated connections to the database. External database clients can make one connection to the remote database and use that

connection for as many transactions as necessary for the session, closing it only when finished.

SERVER-SIDE WEB DATABASE PROGRAMMING

CGI (Common Gateway Interface)

CGI is a protocol for allowing Web browsers to communicate with Web servers, such as sending data to the servers. Upon receiving the data, the Web server can then pass them to a specified external program (residing on the server host machine) via environment variables or standard input stream (STDIN). The external program is called a CGI program or CGI script. Because CGI is a protocol, not a library of functions written specifically for any particular Web server, CGI programs/scripts are language independent. As long as the program/script conforms to the specification of the CGI protocol, it can be written in any language such as C, C++ or Java. In short, CGI is the protocol governing communications among browsers, servers and CGI programs.

In general, a Web server is only able to send documents and to tell a browser what kinds of documents it is sending. By using CGI, the server can also launch external programs (i.e. CGI programs). When the server recognizes that a URL points to a file, it returns the contents of that file. When the URL points to a CGI program, the server will execute it and then send back the output of the program's execution to the browser as if it were a file.

Before the server launches a CGI program, it prepares a number of environment variables representing the current state of the server which is requesting the action. The program collects this information and reads STDIN. It then carries out the necessary processing and writes its output to STDOUT (the standard output stream). In particular, the program must send the MIME header information prior to the main body of the output. This header information specifies the type of the output.

Refer to the figure under the Basic Concepts section. The CGI approach enables access to databases from the browser. The Web client can invoke a CGI program/script via a browser, and then the program performs the required action and accesses the database via the gateway. The outcome of accessing the database is then returned to the client via the Web server. Invoking and executing CGI programs from a Web browser is mostly transparent to the user. The following steps need to be taken in order for a CGI program to execute successfully:

- The user (Web client) calls the CGI program by clicking on a link or by pressing a button. The program can also be invoked when the browser loads an HTML document (hence being able to create a dynamic Web page).
- The browser contacts the Web server, asking for permission to run the CGI program.
- The server checks the configuration and access files to ensure that the program exists and the client has access authorization to the program.
- The server prepares the environment variables and launches the program.

- The program executes and reads the environment variables and STDIN.
- The program sends the appropriate MIME headers to STDOUT, followed by the remainder of the output, and terminates.
- The server sends the data in STDOUT (i.e. the output from the program's execution) to the browser and closes the connection.
- The browser displays the information received from the server.

As mentioned earlier, when preparing data for the browser to display, the CGI program has to include a header as the first line of output. It specifies how the browser should display the output. This header may be one of the following types:

Header	Type of output (document)
Content-type: text/html	an HTML document
Content-type: text/plain	ordinary text
Content-type: image/gif	a GIF file (Graphics Interchange Format)
Content-type: image/jpeg	a JPEG file (Joint Photographic Experts Group)
Content-type: image/png	a Portable Network Graph
Content-type: application/postscript	postscript document
Content-type: video/avi Microsoft Audio	Visual Interleave file
Content-type: video/mov	Apple QuickTime Movie file
Content-type: video/mpeg	Moving Picture Experts Group file

Primarily, there are four methods available for passing information from the browser to a CGI program. In this way, clients' input (representing users' specific requirements) can be transmitted to the program for actions.

1. Passing parameters on the command line.

2. Passing environment variables to CGI programs.

3. Passing data to CGI programs via STDIN.

4. Using extra path information.

Detailed discussions on these methods are beyond the scope of this chapter. Please refer to any book dealing specifically with the CGI topic.

Advantages and disadvantages of CGI Advantages

CGI is the de facto standard for interfacing Web clients and servers with external applications, and is arguably the most commonly adopted approach for interfacing Web applications to data sources (such as databases). The main advantages of CGI are its simplicity, language independence, Web server independence and its wide acceptance.

Disadvantages

The first notable drawback of CGI is that the communication between a client (browser) and the database server must always go through the Web server in the middle, which may cause a bottleneck if there is a large number of users accessing the Web server simultaneously. For every request submitted by a Web client or every response delivered by the database server, the Web server has to convert data from or to an HTML document. This incurs a significant overhead to query processing.

The second disadvantage of CGI is the lack of efficiency and transaction support in a CGI program. For every query submitted through CGI, the database server has to perform the same logon and logout procedure, even for subsequent queries submitted by the same user. The CGI program could handle queries in batch mode, but then support for online database transactions that contain multiple interactive queries would be difficult.

The third major shortcoming of CGI is due to the fact that the server has to generate a new process or thread for each CGI program. For a popular site (like Yahoo), there can easily be hundreds or even thousands of processes competing for memory, disk and processor time. This situation can incur significant overhead.

Last but not least, extra measures have to be taken to ensure server security. CGI itself does not provide any security measures, and therefore developers of CGI programs must be security conscious. Any request for unauthorized action must be spotted and stopped.

Extended CGI

As discussed in the previous section, one of the major concerns with CGI is its performance. With CGI, a process is spawned on the server each time a request is made for a CGI program. There is no method for keeping a spawned process alive between successive requests, even if they are made by the same user. Furthermore, CGI does not inherently support distributed processing, nor does it provide any mechanism for sharing commonly used data or functionality among active and future CGI requests. Any data that exists in one instance of a CGI program cannot be accessed by another instance of the same program.

In order to overcome these problems, an improved version of CGI, called FastCGI, has been developed with the following features:

- **Language independence:** As with CGI, FastCGI is a protocol and not dependent on any specific language.
- **Open standard:** Like CGI, FastCGI is positioned as an open standard. It can be implemented by anyone. The specifications, documentation and source code (in different languages) can be obtained at the Web site https://soramimi.jp/fastcgi/fastcgispec.html.
- **Independence from the Web server architecture:** A FastCGI application need not be modified when an existing Web server architecture changes. As long as the new architecture supports the FastCGI protocol, the appli- cation will continue to work.

- **Distributed computing:** FastCGI allows the Web application to be run on a different machine from the Web server. In this way, the hardware can be tuned optimally for the software.
- **Multiple, extensible roles:** In addition to the functionality offered by CGI (i.e. receiving data and returning responses), FastCGI can fill multiple roles such as a filter role and an authorizer role. A FastCGI application can filter a requested file before sending it to the client; the authorizer program can make an access control decision for a request, such as looking up a username and password pair in a database. If more roles are needed, more definitions and FastCGI programs can be written to fulfil them.
- **Memory sharing:** In some cases, a Web application might need to refer to a file on disk. Under CGI, the file would have to be read into the memory space of that particular instance of the CGI program; if the CGI program were accessed by multiple users simultaneously, the file would be loaded and duplicated into different memory locations. With FastCGI, different instances of the same application can access the same file from the same section of memory without duplication. This approach improves performance.
- **Allocating processes:** FastCGI applications do not require the Web server to start a new process for each application instance. Instead, a certain number of processes are allotted to the FastCGI application. The number of processes dedicated for an application is user-definable. These processes can be initiated when the Web server is started or on demand.

FastCGI seems to be a complete solution for Web database programming, as it includes the best features of CGI and server APIs. In the following sections, a number of CGI-alternative approaches are discussed.

HTTP Server Apis and Server Modules

HTTP server (Web server) APIs and modules are the server equivalent of browser extensions. The central theme of Web database sites created with HTTP server APIs or modules is that the database access programs coexist with the server. They share the address space and run-time process of the server. This approach is in direct contrast to the architecture of CGI, in which CGI programs run as separate processes and in separate memory spaces from the HTTP server.

Instead of creating a separate process for each CGI program, the API offers a way to create an interface between the server and the external programs using dynamic linking or shared objects. Programs are loaded as part of the server, giving them full access to all the I/O functions of the server. In addition, only one copy of the program is loaded and shared among multiple requests to the server.

Server Vendor Modules

Server modules are just prefabricated applications written in some server APIs. Developers can often purchase commercial modules to aid or replace the development of an application feature. Sometimes, the functionality required in a Web database application can be found as an existing server module.

Vendors of Web servers usually provide proprietary server modules to support their products. There are a very large number of server modules that are commercially available, and the number is still rising. For example, Oracle provides the Oracle PL/SQL module, which contains procedures to drive database- backed Web sites. The Oracle module supports both NSAPI and ISAPI.

Advantages of server APIs and modules

Having database access programs coexist with the HTTP server improves Web database access due to improved speed, resource sharing, and the range of functionality.

- **Server speed**

  API programs run as dynamically loaded libraries or modules. A server API program is usually loaded the first time the resource is requested, and therefore, only the first user who requests that program will incur the overhead of loading the dynamic libraries. Alternatively, the server can force this first instantiation so that no user will incur the loading overhead. This technique is called preloading. Either way, the API approach is more efficient than CGI.

- **Resource sharing**

  Unlike a CGI program, a server API program shares address space with other instances of itself and with the HTTP server. This means that any common data required by the different threads and instances need exist only in one place. This common storage area can be accessed by concurrent and separate instances of the server API program.

  The same principle applies to common functions and code. The same set of functions and code are loaded just once and can be shared by multiple server API programs. The above techniques save space and improve performance.

- **Range of functionality**

  A CGI program has access to a Web transaction only at certain limited points. It has no control over the HTTP authentication scheme. It has no contact with the inner workings of the HTTP server, because a CGI program is considered external to the server.

  In contrast, server API programs are closely linked to the server; they exist in conjunction with or as part of the server. They can customize the authentication method as well as transmission encryption methods. Server API programs can also customize the way access logging is performed, providing more detailed transaction logs than are available by default.

  Overall, server APIs provide a very flexible and powerful solution to extending the capabilities of Web servers. However, this approach is much more complex than CGI, requiring specialized programmers with a deep understanding of the Web server and sophisticated programming skills.

Important Issues

- **Server architecture dependence**

  Server APIs are closely tied to the server they work with. The only way to provide efficient cross-server support is for vendors to adhere to the same API standard. If a common API standard is used, programs written for one server will work just as well with another server. However, setting up standards involves compromises among competitors. In many cases, they are hard to come by.

- **Platform Dependence**

  Server APIs and modules are also dependent on computing platforms. Some servers are supported on multiple platforms. Nevertheless, each supporting version is dependent on that platform. Similarly, the Microsoft server is only available for various versions of Windows.

- **Programming Language**

  Most Web servers can be extended using a variety of programming languages and facilities. In addition, Microsoft provides an application environment called Active Server Pages. Active Server Pages is an open, compile-free application environment in which developers can combine HTML, scripts and reusable ActiveX server components to create dynamic and powerful Web-based business solutions.

Comparison of CGI, server APIs and modules, and FastCGI

The following table provides a straightforward comparison among approaches of CGI, server APIs and modules, and FastCGI:

Features	CGI	APIs and Modules	APIs and Modules
Language-independent	Yes		Yes
Runs in different process from core Web server	Yes		Yes
Open standard	Yes		Yes
Web server architecture-independent	Yes		Yes
Supports distributed computing		Yes	Yes
Multiple, extensible roles		Yes	Yes
Memory sharing with other processes		Yes	Yes
Does NOT create new process for each instance		Yes	Yes
Easy to use	Yes		Yes

Proprietary HTTP servers

A proprietary HTTP server is defined as a server application that handles HTTP requests and provides additional functionality that is not standard or common among available HTTP servers. The functionality includes access to a particular database or data source, and translation from a legacy application environment to the Web.

Examples of proprietary servers include IBM Domino, Oracle Application Express Listener and Hyper-G. These products were created for specific needs. For Domino, the need is tight integration with legacy Lotus Notes applications, al- lowing them to be served over the Web. Oracle Application Express Listener was designed to provide highly efficient and integrated access to back-end Oracle databases. For Hyper-G, the need is to have easily maintainable Web sites with automatic link update capabilities.

The main objectives of creating proprietary servers are to meet specialized and customized needs, and to optimize performance. However, the benefits of proprietary servers must be carefully weighed against their exclusive ties to a Web database product (which may bring many shortcomings). It requires a thorough understanding of the business requirements in order to determine whether or not a proprietary Web server is appropriate in a project.

CONNECTING TO THE DATABASE

In previous sections, we have studied various approaches that enable browsers (Web clients) to communicate with Web servers, and in turn allow Web clients to have access to databases. For example, CGI, FastCGI or API programs can be invoked by the Web client to access the underlying database. In this section, we are going to discuss how database connections can actually be made via those CGI/FastCGI/API programs. We will learn what specific techniques, tools and languages are available for making the connections. In short, we will see how the database connection layer is built for the underlying database.

In general, database connectivity solutions include the use of:

- Native database APIs
- Database-independent APIs
- Template-driven database access packages
- Third-party class libraries

Do not be confused with the concepts of Web server APIs and database APIs. Web server APIs are used to write server applications, in which database APIs are used specifically for connecting to and accessing the database. Also, database APIs can be used to write a CGI program which allows developers to create a Web application with a database back end. Similarly, a template- driven database access package, along with a program written in a Web server's API (e.g. NSAPI, ISAPI), is another way to link a Web front end to a database back end.

Database API Libraries

Before we look at specific API database connectivity solutions, let's give back- ground to database API libraries.

Database API libraries are at the core of every Web database application and gateway. Regardless how a Web database application is built (whether by manually coding CGI programs or by using a visual application builder), database API libraries are the foundation of database access.

The Approach

Database API libraries are collections of functions or object classes that provide source code access to databases. They offer a method of connecting to the database engine (under a username and password if user authentication is supported by the DBMS), sending queries across the connection, and retrieving the results and/or error messages in a desired format.

Traditional client-server database applications have already employed database connectivity libraries supplied by vendors and third-party software companies (i.e., third party class libraries). Because of this fact of wider user base, database APIs have the advantage over other gateway solutions for Web database connectivity.

The Web database applications that require developers to use database API libraries are mainly CGI, FastCGI or server API programs. Web database application building tools, including template-driven database access packages and visual GUI builders, use database APIs as well as the supporting gateways (such as CGI and server API), but all these interactivities are hidden from the developers.

Native Database APIs

Native database APIs are platform-dependent as well as programming language dependent. However, most popular databases (such as Oracle) support multiple platforms in the first place, and therefore, the porting between different platforms should not require excessive effort.

In general, programs that use native database APIs are faster than those using other methods, because the libraries provide direct and low-level access. Other database access methods tend to be slower, because they add another layer of programming to provide the developer a different, easier, or more customized programming interface. These additional layers slow the overall transaction down.

Native database API programming is not inherently dependent on a Web server. For example, a CGI program using native API calls to Oracle that works with the Netscape server should also work with other types of servers. However, if the CGI program also incorporates Web server-specific functions or modules, it will be dependent on that Web server.

Database-Independent APIs: ODBC

The most popular standard database-independent API was pioneered by Microsoft. It is called ODBC (Open Database Connectivity) and is supported by all of the most popular databases such as Microsoft Access and Oracle.

ODBC requires a database-specific driver or client to be loaded on the database client machine. In a Java application that accesses Oracle, for example, the server that hosts the Java application would need to have an Oracle ODBC client installed. This client would allow the Java application to connect to the ODBC data source (the actual database) without knowing anything about the Oracle database.

In addition to the database-specific ODBC driver being installed on the client machine, Java requires that a JDBC-ODBC bridge (i.e. another driver. JDBC stands for Java Database Connectivity) be present on the client machine. This JDBC-ODBC driver translates JDBC to ODBC and vice versa, so that Java programs can access ODBC-compliant data sources but still use their own JDBC class library structure.

Having the database-specific ODBC driver on the client machine dictates that Web database Java applications or applets using ODBC be 3-tiered. The database client of the Web application must reside on a server: either the same server as the Web server or a remote server. Otherwise, the database-specific ODBC driver would have to exist on every user's computer, which is a very undesirable situation. The diagram below provides a graphical illustration of such an architecture.

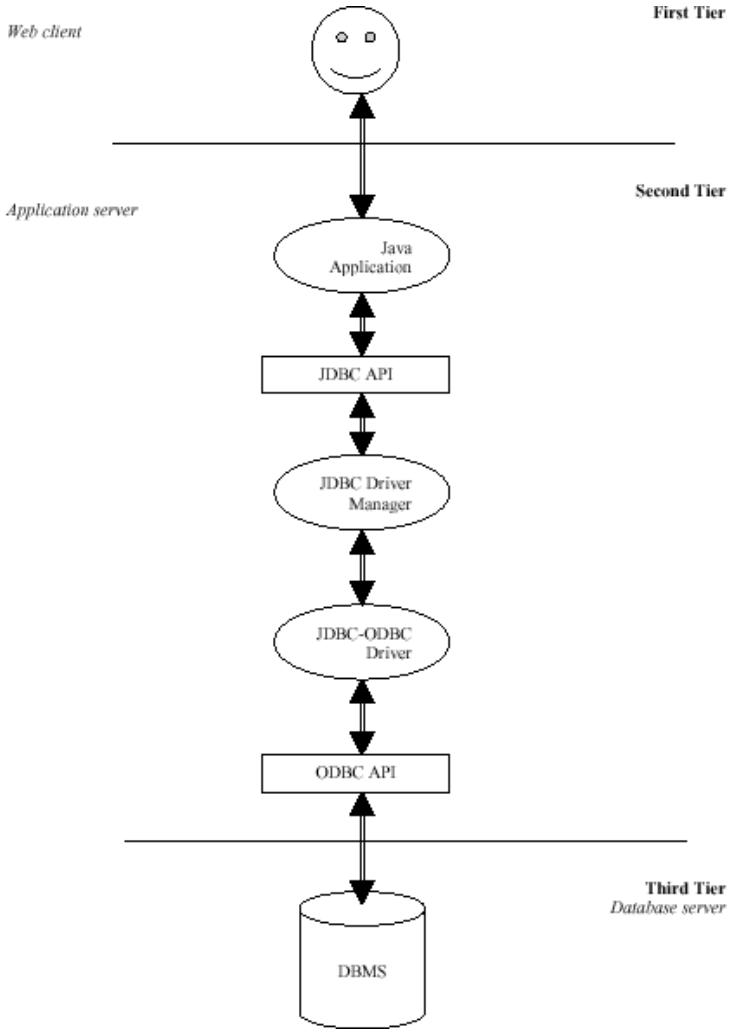
Benefits of Database APIs

Database APIs (native or independent) arguably offer the most flexible way in which Web database applications are created. Applications created with native database APIs are more efficient than those with database-independent APIs. This database connectivity solution is the fastest way to access database functionality and has been tested rigorously in the database software industry. It is worth noting that database APIs have been used successfully for years even before the invention of the Web.

Shortcomings of Database APIs

The most notable disadvantage of programming in database API is complexity. For rapid application development and prototyping, it is better to use a high- level tool, such as template-driven database access software or visual application builders.

Another disadvantage is with ODBC. Because ODBC standardizes access to databases from multiple vendors, applications using ODBC do not have access to native SQL database calls that are not supported by the ODBC standard. In some cases, this can be inconvenient and may even affect application performance.



Template-Driven Packages

The Approach

Template-driven database connectivity packages are offered by database vendors and third-party developers to simplify Web database application programming. Such a package usually consists of the following components:

- Template consisting of HTML and non-standard tags or directives
- Template parser
- Database client daemons

Template-driven packages are very product dependent. Different DBMSs re- quire database access templates in different formats. An application developed for one product will be strongly tied to it. Migrating from one product to an- other is very difficult and requires a rewrite of all the database access, flow control and output-formatting commands.

An example of a template-driven package is PHP.

Benefits Of Template-Driven Packages

The most important benefit from using a template-driven package is speed of development. Assuming an available package has been installed and configured properly, it takes as little time as a few hours to create a Web site that displays information directly from the database.

Shortcomings Of Template-Driven Packages

The structures of templates are normally predetermined by vendors or third- party developers. As a result, they only offer a limited range of flexibility and customizability. Package vendors provide what they feel is important functionality, but, as with most off-the-shelf tools, such software packages may not let you create applications requiring complex operations.

Although templates offer a rapid path to prototyping and developing simple Web database applications, the ease of development is obtained for the cost of speed and efficiency. Because the templates must be processed on demand and require heavy string manipulation (templates are of a large text type or string type; they must be parsed by the parser), using them is slow compared with using direct access such as native database APIs.

The actual performance of an application should be tested and evaluated before the usefulness of such a package is ruled out. The overhead of parsing templates may be negligible if using high-performance machines. Other factors, such as development time or development expertise, may be more important than a higher operational speed.

GUI Application Builders

Visual Web database building tools offer an interesting development environment for creating Web database applications. For developers accustomed to point-and-click application programming, these tools help speed the development process. For instance, Visual Basic and/or Microsoft Access developers should find such a tool intuitive and easy to use.

The Approach

The architectures of visual building tools vary. In general, they include a user- friendly GUI (Graphical User Interface), allowing developers to build a Web database application with a series of mouse clicks and some textual input. These tools also offer application management so that a developer no longer needs to juggle multiple HTML documents and CGI, NSAPI or ISAPI programs manually.

At the end of a building session, the tool package can generate applications using various techniques. Some applications are coded using ODBC; some use native database APIs for the databases they support; and others may use database net protocols.

Some of these tools create their own API, which can be used by other developers. Some generate code that works but can still be modified and customized by developers using various traditional IDEs, compilers and debuggers.

A building tool may generate a CGI program or a Web server API program (such as NSAPI and ISAPI). Some sophisticated tools even offer all the options. The developer can choose what he/she wants.

Unlike native database APIs or template-driven database connectivity packages, visual Web database development tools tend to be as open as possible. Many offer development support for the popular databases.

Benefits of Visual Tools

Visual development tools can be of great assistance to developers who are familiar and comfortable with visual application development techniques. They offer rapid application development and prototyping, and an organized way to manage the application components. Visual tools also shield the developer from low-level details of Web database application development. As a result, a developer can create a useful Web application without the need to know what is happening in the code levels.

Shortcomings of Visual Tools

Depending on the sophistication of the package used, the resulting programs may be slower to execute than similar programs coded by an experienced programmer. Visual application building tools, particularly Object-oriented ones, tend to generate fat programs with a lot of unnecessary sub-classing.

Another potential drawback is cost. A good visual tool may be too expensive for a small one-off development budget.

MANAGING STATE AND PERSISTENCE IN WEB APPLICATIONS

State is an abstract concept of being, which can be explained by a set of rules, facts or truisms. A state in a database application includes a set of variables and/or other means to record who the user/client is, what tasks he/she has been doing, at what position he/she is at a particular instance in time, and many other useful pieces of information about a database session. Persistence is the capability of remembering a state and tracking state changes across different applications or different periods of time within an instance of an application or multiple instances.

The requirement of state maintenance in Web database applications results in the increased complexity. As mentioned before in the Context section, HTTP is connectionless, which means that once an HTTP request is sent and a response is received, the connection to the server is closed. If a connection were to be kept open between client and server, the server could at any time query the client for state information and vice versa. The server would be able to know the identity of the user throughout the session once the user logged in. However, the reality is that there is no constant connection throughout the session. Thus, the server cannot

have memory of the user's identity even after user login. In this situation, programmers must find a way to make session state persist.

Technical Options

There are several options available to programmers to maintain state. They range from open systems options defined in HTTP and CGI standards, to proprietary mechanisms written from scratch.

The most important task in maintaining persistence is to keep track of the identity of the user. If the identity can persist, any other data/information can usually be made to persist in exactly the same manner.

The URL Approach

It works as follows:

- A registration or login page is delivered to the user.
- The user types in a username and password, and then submits the page.
- The username and password pair are sent to a server-side CGI program, which extracts the values from the QUERY_STRING environment variable.
- The values are checked by the server to determine whether or not the user is authenticated.
- If he/she is authenticated, the authenticated state is reflected in a randomly generated session ID (SID), which is stored in a database along with other necessary data to describe the state of the user session.
- The SID can then be stored in all URLs within HTML documents returned by the server to the client, therefore tracking the identity of the user throughout the session.

Benefits of the URL Approach

The URL approach is easy to use to maintain state. To retrieve a state, the receiving CGI program need only collect the data from environment variables in the GET method and act on it as necessary. To pass on, set or change the state, the program simply creates new URLs with the appropriate data.

Shortcomings of the URL Approach

If the state information has to be kept in the URL, the URL becomes very long and can be very messy. Also, such a URL displays part of the application code and low-level details. This causes security concerns, and may be used by hackers.

If an application manages state on the client side using the URL method, the state will be lost when the user quits the browser session unless the user book- marks the URL. A bookmark saves the URL in the browser for future retrieval. If state is maintained solely in the URL without any server-side state data management, bookmarking is sufficient to recreate the state in a new browser session. However, having the user perform this maintenance task is obviously undesirable.

URL QUERY_STRING

This is another popular method of maintaining state. A registered user in a site has a hidden form appended to each page visited within the site. This form contains the username and the name of the current page. When the user moves from one page to another, the hidden form moves as well and is appended to the end of the succeeding HTML page.

Benefits of the Hidden Fields Approach

Like the URL approach, it is easy to use to maintain state. In addition, because the fields are hidden, the user has a seamless experience and sees a clean URL.

Another advantage of using this approach is that, unlike using URLs, there is no limit on the size of data that can be stored.

Shortcomings of the Hidden Fields Approach

As with the URL approach, users can fake states by editing their own version of the HTML hidden fields. They can bring up the document source in an editor, change the data stored, and then submit the tampered form to the server. This raises serious security concerns.

Data is also lost between sessions. If the entire session state is stored in hidden fields, that state will not be accessible after the user exits the browser unless the user specifically saves the HTML document to disk or with a bookmark. Again, it is undesirable to involve users in this kind of maintenance task.

HTTP Cookies

An HTTP cookie is a technique that helps maintain state in Web applications. A cookie is in fact a small text file containing:

- Name of the cookie
- Domains for which the cookie is valid
- Expiration time in GMT
- Application-specific data such as user information

Cookies are sent by the server to the browser, and saved to the client's disk. Whenever necessary, the server can request a desired cookie from the client. The client browser will check whether it has it. If it does, the browser will send the information stored in the cookie to the server.

Benefits of Cookies

Cookies can be completely transparent. As long as a user does not choose the browser option to be alerted before accepting cookies, his/her browser will handle incoming cookies and place them on the client disk without user intervention.

Cookies are stored in a separate file, whose location is handled by the browser and difficult for the user to find. Also, cookies are difficult to tamper with. This increases security.

Because cookies are stored on the client disk, the cookie data is accessible even in a new browser session. It does not require the user to do anything.

If a programmer chooses to set an expiration date or time for a cookie, the browser will invalidate the cookie at the appropriate time.

Shortcomings of Cookies

The amount of data that can be stored with a cookie is usually limited to 4 kilobytes. If an application has very large state data, other techniques must be considered.

Because cookies are physically stored on the client disk, they cannot move with the user. This side effect is important for applications whose users often change machines.

Although cookies are difficult to tamper with, it is still possible for someone to break into them. Remember a cookie is just a text file. If a user can find it and edit it, it can still cause security problems.

Important Considerations

1. Managing State on the Client

An application can maintain all of its state on the client-side with any of the methods discussed in the previous section.

- **Benefits of the client-side maintenance**

  On attraction of maintaining state on the client is simplicity. It is easier to keep all the data in one place, and by doing it on the client, it eliminates the need for server database programming and maintenance.

  If an application uses client-side extensions to maintain state, it can also provide a faster response to the user because the need to network access is eliminated.

- **Shortcomings of the client-side maintenance**

If all the state data is on the client-side, there is a danger that users can somehow forge state information by editing URLs, hidden fields, and cookies. This leads to security risks in server programs.

With the exception of the cookie approach to maintaining state, there is no guarantee that the necessary data will be saved when the client exits unexpectedly. Thus, the robustness of the application is compromised.

2. Managing State on the Server

This approach for maintaining state actually involves using both the client and the server. Usually a small piece of information, either a user ID or a session key is stored on the client-side. The server program uses this ID or key to look up the state data in a database.

- **Benefits of the server-side maintenance**

  Maintaining state on the server is more reliable and robust than the client- side maintenance. As long as the client can provide an ID or a key, the user's session state can be restored, even between different browsing sessions.

  Server-side maintenance can result in thin clients. The less dependent a Web database application is on the client; the less code needs to exist on or be transmitted to the client.

  Server-side maintenance also leads to better network efficiency, because only small amounts of data need to be transmitted between the client and the server.

- **Shortcomings of the server-side maintenance**

  The main reason an application would not be developed using server-side state maintenance is its complexity, because it requires the developer to write extensive code. However, the benefits of implementing server-side state management outweigh the additional work required.

SECURITY ISSUES IN WEB DATABASE APPLICATIONS

Security risks exist in many areas of a Web database application. This is because the very foundations of the Internet and Web – TCP/IP and HTTP – are very weak with respect to securities. Without special software, all Internet traffic travels in the open and anyone with a little bit skill can intercept data transmission on the Internet. If no measures are taken, there will be many security loopholes that can be explored by malicious users on the Internet.

In general, security issues in Web database applications include the following:

- Data transmission (communication) between the client and the server is not accessible to anyone else except the sender and intended receiver (privacy).
- Data cannot be changed during transmission (integrity).
- The receiver can be sure that the data is from the authenticated sender (authenticity).

- The sender can be sure the receiver is the genuinely intended one (non- fabrication).
- The sender cannot deny he/she sent it (non-repudiation).
- The request from the client should not ask the server to perform illegal or unauthorized actions.
- The data transmitted to the client machine from the server must not be allowed to contain executables that will perform malicious actions.

At the present, there are a number of measures that can be taken to address some of the above issues. These measures are not perfect in the sense that they cannot cover every eventuality, but they should help get rid of some of the loopholes. It must be stressed that security is the most important but least understood aspect of Web database programming. More work still needs to be done to enhance security.

Proxy Servers: A proxy server is a system that resides between a Web browser and a Web server. It intercepts all requests to the Web server to determine if it can fulfil the requests itself. If not, it forwards the requests to the Web server.

Due to the fact that the proxy server is between browsers and the Web server, it can be utilized to be a defense for the server.

Firewalls: Because a Web server is open for access by anyone on the Internet, it is normally advised that the server should not be connected to the intranet (i.e., an organization's internal network). This way, no one can have access to the intranet via the Web server.

However, if a Web application has to use a database on the intranet, then the firewall approach can be used to prevent unauthorized access.

A firewall is a system designed to prevent unauthorized access to or from a private network (intranet). It can be implemented in either hardware, software, or both. All data entering or leaving the intranet (connected to the Internet) must pass through the firewall. They are checked by the firewall system and anything that does not meet the specified security criteria is blocked.

A proxy server can act as a firewall because it intercepts all data in and out, and can also hide the address of the server and intranet.

Digital Signatures: A digital signature consists of two pieces of information: a string of bits that is computed from the data (message) that is being signed along with the private key of the requester for the signature.

The signature can be used to verify that the data is from a particular individual or organization. It has the following properties:

- Its authenticity is verifiable using a computation based on a corresponding public key.

- If the private key is kept secret, the signature cannot be forged.
- It is unique for the data signed. The computation will not produce the same result for two different messages.
- The signed data cannot be changed, otherwise the signature will no longer verify the data as being authentic.

The digital signature technique is very useful for verifying authenticity and maintaining integrity.

Digital Certificates: A digital certificate is an attachment to a message used for verifying the sender's authenticity. Such a certificate is obtained from a Certificate Authority (CA), which must be a trust-worthy organization.

When a user wants to send a message, he/she can apply for a digital certificate from the CA. The CA issues an encrypted certificate containing the applicant's public key and other identification information. The CA makes its own key publicly available.

When the message is received, the recipient uses the CA's public key to decode the digital certificate attached to the message, verifies it as issued by the CA, and then obtains the sender's public key and identification information held within the certificate. With this information, the recipient can send an encrypted reply.

Kerberos: Kerberos is a server of secured usernames and passwords. It provides one centralized security server for all data and resources on the network. Database access, login, authorization control, and other security measures are centralized on trusted Kerberos servers. The main function is to identify and validate a user.

Secure Sockets layer (SSL) and Secure HTTP (S-HTTP): SSL is an encryption protocol developed by Netscape for transmitting private documents over the Internet. It works by using a private key to encrypt data that is to be transferred over the SSL connection. Netscape, Firefox, Chrome and Microsoft IE support SSL.

Another protocol for transmitting data securely over the Internet is called Secure HTTP, a modified version of the standard HTTP protocol. Whereas SSL creates a secure connection between a client and a server, over which any amount of data can be sent securely, S-HTTP is designed to transmit individual messages securely.

In general, the SSL and S-HTTP protocols allow a browser and a server to establish a secure link to transmit information. However, the authenticity of the client (the browser) and the server must be verified. Thus, a key component in the establishment of secure Web sessions using SSL or S-HTTP protocols is the digital certificate. Without authentic and trustworthy certificates, the protocols offer no security at all.

Java Security: If Java is used to write the Web database application, then many security measures can be implemented within Java. Three Java components can be utilized for security purposes:

- **The class loader:** It not only loads each required class and checks it is in the correct format, but also ensures that the application/applet does not violate system security by allocating a namespace. This technique can effectively define security levels for each class and ensure that a class with a lower security clearance can never be in place of a class with a higher clearance.
- **The bytecode verifier:** Before the Java Virtual Machine (JVM) will allow an application/applet to execute, its code must be verified to ensure: compiled code is correctly formatted; internal stacks will not overflow or underflow; no illegal data conversions will occur; bytecode instructions are correctly typed; and all class member accesses are valid.
- The security manager: An application-specific security manager can be defined within a browser, and any applets downloaded by this browser are subject to its (security manager's) security policies. This can prevent a client from being attacked by dangerous methods.

ActiveX Security: For Java, security for the client machine is one of the most important design factors. Java applet programming provides as many features as possible without compromising the security of the client. In contrast, ActiveX's security model places the responsibility for the computer's safety on the user (client). Before a browser downloads an ActiveX control that has not been digitally signed or has been certified by an unknown CA, it displays a dialog box warning the user that this action may not be safe. It is up to the user to decide whether to abort the downloading, or continue and accept a potential damaging consequence.

Performance issues in Web database applications: Web database applications are very complex, more so than stand-alone or traditional client-server applications. They are a hybrid of technology, vendors, programming languages, and development techniques.

Many factors work together in a Web database application and any one of them can hamper the application's performance. It is crucial to understand the potential bottlenecks in the application as well as to know effective, well-tested solutions to address the problems.

The following is a list of issues concerning performance:

- **Network consistency:** The availability and speed of network connections can significantly affect performance.
- **Client and server resources:** This is the same consideration as in the traditional client-server applications. Memory and CPU are the scarce resources.
- **Database performance:** It is concerned with the overhead for establishing connections, database tuning, and SQL query optimization.
- **Content delivery:** This is concerned with the content's download time and load time. The size of any content should be minimized to reduce download time; and appropriate format

should be chosen for a certain document (mainly images and graphics) so that load time can be minimized.

- **State maintenance:** It should always minimize the amount of data transferred between client and server and minimize the amount of processing necessary to rebuild the application state.
- **Client-side processing:** If some processing can be carried out on the client-side, it should be done so. Transmitting data to the server for processing that can be done on the client-side will degrade performance.
- **Programming language:** A thorough understanding of the tasks at hand and techniques available can help choose the most suitable language for implementing the application.

CHAPTER

5

TECHNOLOGIES AND TOOLS

Database connectivity interfaces allow software applications to communicate with database management systems (DBMSs). They enable applications to access, create, update, and delete data from a database.

DATABASE CONNECTIVITY INTERFACES

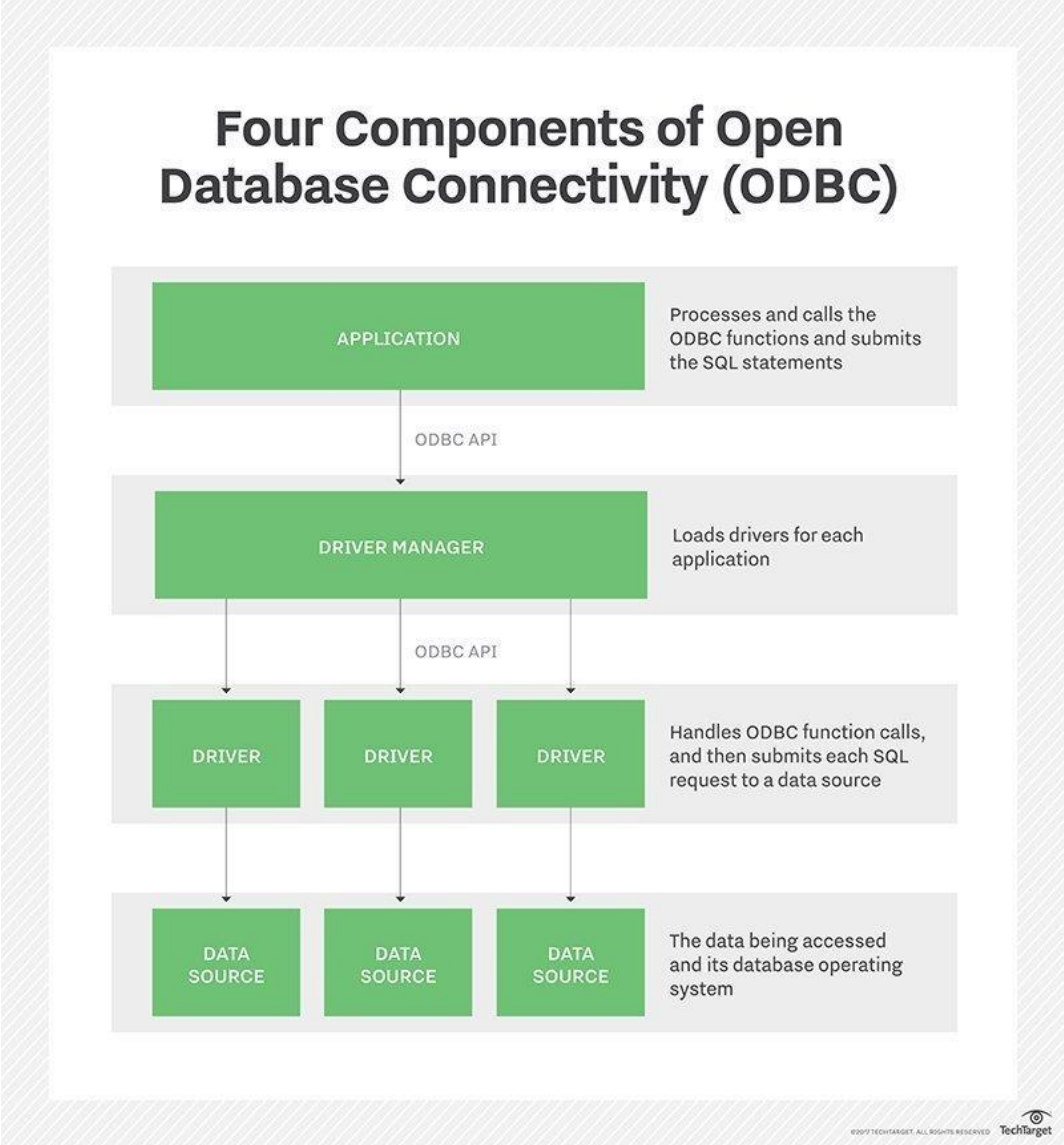
Open Database Connectivity (ODBC): is an open standard application programming interface (API) that allows application programmers to easily access data stored in a database.

The main proponent and supplier of ODBC programming support is Microsoft, but ODBC is based on and closely aligned with the Structured Query Language (SQL) Call Level Interface (CLI) standard from The Open Group. Sponsored by many major vendors, including Oracle, IBM and Hewlett Packard Enterprise, The Open Group is a consortium that develops and manufactures The Open Group Architecture Framework, or TOGAF. In addition to CLI specifications from The Open Group, ODBC also aligns with the International Organization for Standardization/International Electrotechnical Commission for database APIs.

How does ODBC work?

ODBC acts as a translator between an application and a data store. The application makes a simple standard call to the ODBC engine, which changes that to a form the database can handle. ODBC can save a lot of time and effort because the program doesn't need to know the details of each database, and the database doesn't need to be configured for every application that wants to access it.

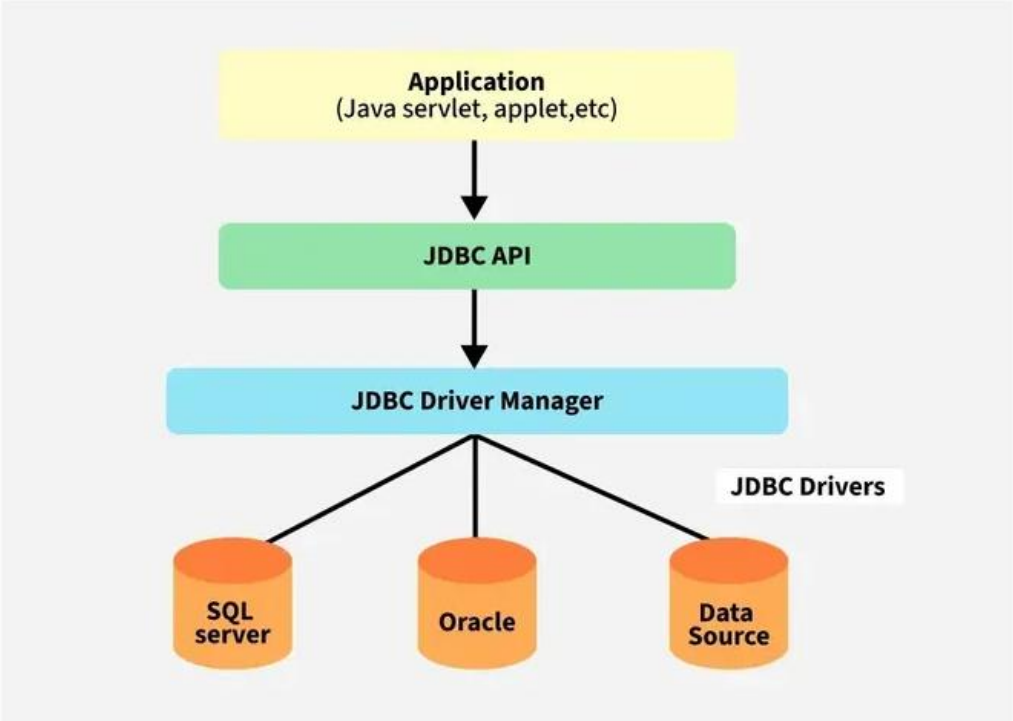
ODBC allows programs to use SQL requests that access databases without knowing the proprietary interfaces to the databases. ODBC handles the SQL request and converts it into a request each database system understands.



ODBC consists of four components, working together to enable functions:

- **Application.** The end-user program that calls the ODBC functions and submits the SQL statements.
- **Driver manager.** Loads the correct driver for each application and database. It can also handle some other tasks that the application might not be fully programmed for, such as authentication and encryption.
- **Driver.** Handles ODBC function calls, and then submits each SQL request to the data source.
- **Data source.** The database being accessed and its database management system. Simple data sources include flat text and comma-separated value files. ODBC can also work with MySQL when its driver is called MyODBC. Sometimes, this is referred to as the MySQL Connector/ODBC.

Java Database Connectivity (JDBC) is an API in Java that enables applications to interact with databases. It allows a Java program to connect to a database, execute queries, and retrieve and manipulate data. By providing a standard interface, JDBC ensures that Java applications can work with different relational databases like MySQL, Oracle, PostgreSQL, and more.



JDBC Architecture

- **Application:** It is a Java applet or a servlet that communicates with a data source.
- **The JDBC API:** It allows Java programs to execute SQL queries and retrieve results. Key interfaces include Driver, ResultSet, RowSet, PreparedStatement, and Connection. Important classes include DriverManager, Types, Blob, and Clob.
- **DriverManager:** It plays an important role in the JDBC architecture. It uses some database-specific drivers to effectively connect enterprise applications to databases.
- **JDBC drivers:** These drivers handle interactions between the application and the database.

JDBC Components

There are generally 4 main components of JDBC through which it can interact with a database. They are as mentioned below:

1. JDBC API

It provides various methods and interfaces for easy communication with the database. It includes two key packages

java.sql: This package, is the part of Java Standard Edition (Java SE) , which contains the core interfaces and classes for accessing and processing data in relational databases. It also provides essential functionalities like establishing connections, executing queries, and handling result sets

javax.sql: This package is the part of Java Enterprise Edition (Java EE) , which extends the capabilities of java.sql by offering additional features like connection pooling, statement pooling, and data source management.

It also provides a standard to connect a database to a client application.

2. JDBC Driver Manager

Driver manager is responsible for loading the correct database-specific driver to establish a connection with the database. It manages the available drivers and ensures the right one is used to process user requests and interact with the database.

3. JDBC Test Suite

It is used to test the operation (such as insertion, deletion, updating) being performed by JDBC Drivers.

4. JDBC Drivers

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand.

JAVA DATABASE CONNECTIVITY VS. OPEN DATABASE CONNECTIVITY

The Java Database Connectivity (JDBC) API uses the Java programming language to access a database. When writing programs in Java using JDBC APIs, users can employ software that includes a JDBC-ODBC bridge to access ODBC-supported databases.

However, the JDBC-ODBC bridge -- or JDBC Type 1 driver -- should be viewed as a transitional approach, as it creates performance overhead; API calls must pass through the JDBC bridge to the ODBC driver, then to the native database connectivity interface. In addition, it was removed in Java Development Kit 8, and Oracle does not support the JDBC-ODBC bridge.

The recommended approach is the use of JDBC drivers provided by database vendors, rather than the JDBC-ODBC bridge.

OTHER ALTERNATIVES TO ODBC

In addition to JDBC, there are other alternatives to ODBC, including the following:

Object Linking and Embedding Database: OLE DB is a Microsoft technology that acts as an API between an application and a data source, like ODBC. OLE DB was designed to work with more types of data sources, such as file systems and spreadsheet tables, instead of primarily databases. Microsoft designed OLE DB to replace ODBC, but most databases continue to support ODBC. The OLE DB manager can usually translate to ODBC connections as necessary.

ADO.NET: ADO.NET is the data access technology provided as part of the .NET Framework. It can be used in .NET programs to work with databases and other data sources. Bridges exist to translate ADO.NET queries into ODBC requests.

Embedded SQL: Embedded SQL is simply putting the desired SQL statements in the application code without a middleman layer, such as ODBC, to translate them. While embedded SQL has largely fallen out of favor, it can still be used in cases where extremely high performance is needed.

DATABASE CONNECTIVITY INTERFACES USE

Connection strings: Is a string of text that provides the necessary information for an application to connect to a database or another data source. It typically includes the database type, location, credentials, and other connection details. Connection strings are critical for database-driven applications as they define how and where to connect to the data source.

Key Components of a Connection String

1. **Database Server Address:**
    - Specifies the server's hostname or IP address where the database is hosted.
    - Example: localhost, 192.168.1.1, or myserver.database.windows.net.
2. **Database Name:**
    - The name of the specific database to connect to.
    - Example: my_database.
3. **Authentication Credentials:**
    - Username and Password: For logging into the database.
    - Example: User ID=my_user; Password=my_password;.
    - Integrated Security: For Windows Authentication (no username/password needed).

4. **Port Number (Optional):**
    - Specifies the port on which the database is listening.
    - Example: 3306 for MySQL or 5432 for PostgreSQL.
5. **Driver or Provider:**
    - Identifies the database type or protocol to use.
    - Example: SQL Server Native Client, ODBC, OLE DB.
6. **Other Optional Parameters:**
    - Timeout: Time to wait before the connection attempt fails.
    - SSL Settings: Enforces secure connections.
    - Pooling Options: For managing connection efficiency.

Best Practices for Connection Strings

Security: Avoid hardcoding sensitive information like passwords in source code. Use environment variables or secret managers.

Encryption: Use encrypted connections (e.g., SSL/TLS) to protect data in transit.

Connection Pooling: Enable connection pooling for better performance in high-traffic applications.

Error Handling: Handle connection failures gracefully by using try-catch blocks.

Environment-Specific Configurations: Use different connection strings for development, testing, and production environments.

DATA PROVIDERS

A **data provider** is a component, service, or entity that supplies data to an application, system, or user. Data providers act as intermediaries, enabling data access from various sources, such as databases, APIs, or external files, in a structured and standardized manner.

Types of Data Providers

1. **Database Providers:**
   These allow applications to interact with databases. Examples include:
   - **ADO.NET Data Providers:** For .NET applications to interact with relational databases like SQL Server, Oracle, or MySQL.
   - **JDBC (Java Database Connectivity):** A Java API to connect and execute queries on databases.
2. **Web API Providers:**
   These provide data over HTTP/HTTPS using REST, SOAP, or GraphQL protocols. Examples include:
   - **OpenWeather API:** For weather data.
   - **Twitter API:** For social media data.
3. **File-Based Providers:**
   Used to access data stored in files like Excel, CSV, or JSON. Examples include:
   - **ODBC (Open Database Connectivity):** For accessing flat files and other non-database sources.
   - **Pandas (Python Library):** For reading and processing data files like Excel or CSV.
4. **Cloud-Based Data Providers:**
   Cloud platforms offering databases and storage solutions with APIs for access. Examples include:

- **AWS S3:** For file storage.
- **Google BigQuery:** For large-scale analytics.

5. **Data Aggregators:**
   These providers compile and offer datasets from multiple sources. Examples include:
   - **World Bank Data:** For global statistics.
   - **Statista:** For market and consumer data.
6. **Streaming Data Providers:**
   Enable access to real-time or live data streams. Examples include:
   - **Kafka Providers:** For real-time message processing.
   - **WebSocket Providers:** For live feeds like stock prices or sports scores.

Common Data Providers for Popular Platforms

1. .**NET Framework (ADO.NET Providers):**

- **SQL Server Provider:** For Microsoft SQL Server databases.
- **Oracle Data Provider for .NET (ODP.NET):** For Oracle databases.
- **MySQL Connector/NET:** For MySQL databases.
- **SQLite Provider:** For lightweight embedded databases.

2. **Java (JDBC Providers):**

- **MySQL JDBC Driver:** For MySQL database connectivity.
- **PostgreSQL JDBC Driver:** For PostgreSQL databases.
- **Oracle JDBC Driver:** For Oracle databases.

3. **Python Libraries:**

- **Psycopg2:** For PostgreSQL databases.
- **SQLAlchemy:** A database toolkit that supports multiple providers like SQLite, MySQL, and Oracle.
- PyMongo: For MongoDB.

4. **PHP Data Objects (PDO):**

- A universal interface for connecting to multiple databases such as MySQL, PostgreSQL, SQLite, and Oracle.

5. **JavaScript Providers:**

- Mongoose: For MongoDB in Node.js.
- Sequelize: For SQL databases like MySQL, PostgreSQL, and SQLite.

How Data Providers Work

- **Connection:** The data provider establishes a connection between the application and the data source.
- Example: A connection string is used to connect to a database.
- **Query Execution:** The provider translates user queries into the native query language (e.g., SQL) of the data source.
- **Data Retrieval:** The provider retrieves data from the source and converts it into a format the application can understand.
- **Data Manipulation:** Providers also support data manipulation tasks like inserting, updating, or deleting records.

Benefits of Using Data Providers

- **Abstraction:** Simplifies interaction with data sources by hiding low-level details.
- **Consistency:** Provides a standardized way to interact with various data sources.
- **Flexibility:** Supports multiple data sources and formats with minimal changes to the application code.
- **Efficiency:** Optimizes query execution and data retrieval processes.

Examples of Data Provider Use Cases

- **E-Commerce Applications:** Using data providers to access customer databases, product catalogs, and transaction records.
- **Business Intelligence Tools:** Connecting to relational databases and APIs to visualize and analyze data.
- **Web Applications:** Fetching data from REST APIs or cloud databases for real-time updates.
- **Data Science and Machine Learning:** Loading and processing large datasets using Python libraries like Pandas and SQLAlchemy.

APPLICATION PROGRAMMING INTERFACES

An API, or application programming interface, is a set of rules or protocols that enables software applications to communicate with each other to exchange data, features and functionality.

APIs simplify and accelerate application and software development by allowing developers to integrate data, services and capabilities from other applications, instead of developing them from scratch. APIs also give application owners a simple, secure way to make their application data and functions available to departments within their organization. Application owners can also share or market data and functions to business partners or third parties.

APIs allow for the sharing of only the information necessary, keeping other internal system details hidden, which helps with system security. Servers or devices do not have to fully expose data—APIs enable the sharing of small packets of data, relevant to the specific request.

API documentation is like a technical instruction manual that provides details about an API and information for developers on how to work with an API and its services. Well-designed documentation promotes a better API experience for users and generally makes for more successful APIs.

How do APIs Work?

It's useful to think about API communication in terms of a request and response between a client and server. The application submitting the request is the client, and the server provides the response. The API is the bridge establishing the connection between them.

A simple way to understand how APIs work is to look at a common example—third-party payment processing. When a user purchases a product on an e-commerce site, the site might prompt the user to "Pay with PayPal" or another type of third-party system. This function relies on APIs to make the connection.

- When the buyer clicks the payment button, an API call is sent to retrieve information. This is the request. This request is processed from an application to the web server through the API's Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes, a request body.
- After receiving a valid request from the product webpage, the API calls to the external program or web server, in this case, the third-party payment system.
- The server sends a response to the API with the requested information.
- The API transfers the data to the initial requesting application, in this case, the product website.

While the data transfer differs depending on the web service used, the requests and responses all happen through an API. There is no visibility on the user interface, meaning APIs exchange data within the computer or application, and appear to the user as a seamless connection.

Types of APIs

APIs can be categorized by use case, including data APIs, operating system APIs, remote APIs and web APIs.

Web APIs

Web APIs enable the transfer of data and functionality over the internet using HTTP protocol.

Today, most APIs are web APIs. Web APIs are a type of remote API (meaning that the API uses protocols to manipulate external resources) that expose an application's data and functionality over the internet.

The four main types of web APIs are:

1. Open APIs: Open APIs are open-source application programming interfaces you can access with the HTTP protocol. Also known as public APIs, they have defined API endpoints and request and response formats.

2. Partner APIs: Partner APIs connect strategic business partners. Typically, developers access these APIs in self-service mode through a public API developer portal. Still, they need to complete an onboarding process and get login credentials to access partner APIs.

3. Internal APIs: Internal, or private, APIs remain hidden from external users. These private APIs aren't available for users outside of the company. Instead, organizations use them to improve productivity and communication across different internal development teams.

4. Composite APIs: Composite APIs combine multiple data or service APIs. They allow programmers to access several endpoints in a single call. Composite APIs are useful in microservices architecture where running a single task might require information from several sources.

Other Types of APIs

Less common types of APIs include:

1. Data (or database) APIs, used to connect applications and database management systems

2. Operating system (or local) APIs, used to define how apps use operating system services and resources

3. Remote APIs, used to define how applications on different devices interact

API Examples

Because APIs allow organizations to open access to their resources while maintaining security and control, they have become a valuable aspect of modern business and personal applications.

Here are some API examples that users often encounter:

Universal logins: A popular API example is the function that enables people to log in to websites by using their Facebook, X, or Google profile login details. This convenient feature allows any website to use an API from one of the more popular services for quick authentication. This capability helps save users the time and hassle of setting up a new profile for every web application or new membership.

Internet of Things (IoT): These “smart devices” offer added functionality, such as internet-enabled touchscreens and data collection, through APIs. For example, a smart fridge can connect to recipe applications or take and send notes to mobile phones through text message.

Internal cameras connect to various applications so that users can see the contents of the refrigerator from anywhere.

Travel booking comparisons: Travel booking sites aggregate thousands of flights, showcasing the cheapest options for every date and destination. APIs enable this service by providing application users access to the latest information about availability from hotels and airlines.

This access is available either through a web browser or the travel booking company's own application. With an autonomous exchange of data and requests, APIs dramatically reduce the time and effort involved in checking for available flights or accommodation.

Navigation apps: Navigation apps use core APIs that display static or interactive maps. These apps also use other APIs and features to provide users with directions, speed limits, points of interest, traffic warnings and more. Users communicate with an API when plotting travel routes or tracking items on the move, such as a delivery vehicle.

Social media: Social media companies use APIs to allow other entities to share and embed content featured on social media apps to their own sites. For example, the Instagram API enables businesses to embed their Instagram grid on their website and for the grid to update automatically as users add new posts.

SaaS applications: APIs are an integral part of the growth in software as a service (SaaS) product. Platforms like CRMs (customer relationship management tools) often include several built-in APIs that let companies integrate with applications they already use, such as messaging, social media and email apps.

This integration drastically reduces time spent switching between applications for sales and marketing tasks. It also helps reduce or prevent data silos that might exist between departments that use different applications.

API Protocols, Architectural Styles and Languages

Traditionally, API referred to an interface connected to an application created with any of the low-level programming languages, such as JavaScript. However, modern APIs vary in their architectures and use of data formats. They are typically built for HTTP, resulting in developer-friendly interfaces that are easily accessible and widely understood by applications written in Java, Ruby, Python and many other languages.

As the use of web APIs has increased, it has led to the development and use of certain protocols, styles, standards and languages. These structures provide users with a set of defined rules, or API specifications, that create accepted data types, commands and syntax. In effect, these API protocols facilitate standardized information exchange.

SOAP (simple object access protocol): is a lightweight XML-based messaging protocol specification that enables endpoints to send and receive data through a range of communication protocols including SMTP (simple mail transfer protocol) and HTTP (hypertext transfer protocol.) SOAP is independent, which allows SOAP APIs to share

information between apps or software components running in different environments or written in different languages.

Remote procedure call (RPC): is a protocol that provides the high-level communications paradigm used in the operating system. RPC presumes the existence of a low-level transport protocol, such as transmission control protocol/internet protocol (TCP/IP) or user datagram protocol (UDP), for carrying the message data between communicating programs.
RPC implements a logical client-to-server communications system designed specifically for the support of network applications. The RPC protocol enables users to work with remote procedures as if the procedures were local.[1]

XML-RPC (XML- remote procedure call): protocol relies on a specific XML format to transfer data. XML-RPC is older than SOAP, but simpler, and relatively lightweight in that it uses minimum bandwidth.

JSON-RPC: Like XML-RPC, JSON-RPC is a remote procedure call, that uses JSON (JavaScript Object Notation) instead of XML. JSON is a lightweight format for data exchange that is simple to parse and uses name/value pairs and ordered lists of values. Because JSON uses universal data structures, it can be used with any programming language.

gRPC is a high-performance, open-source RPC framework initially developed by Google. gRPC uses the network protocol HTTP/2 and Protocol Buffers data format and is commonly used to connect services in a microservices architecture.

WebSocket APIs enable bidirectional communication between client and server. This type of API does not require a new connection to be established for each communication—once the connection is established it allows for continuous exchange. This makes Web Socket APIs ideal for real-time communication.

REST (representational state transfer)

REST is a set of web API architecture principles. REST APIs—also known as RESTful APIs—are APIs that adhere to certain REST architectural constraints. REST APIs use HTTP requests such as GET, PUT, HEAD and DELETE to interact with resources. REST makes data available as resources, with each resource represented by a unique URI. Clients request a resource by providing its URI.

REST APIs are stateless—they do not save client data between requests. It's possible to build RESTful APIs with SOAP protocols, but practitioners usually view the two standards as competing specifications.

GraphQL is an open-source query language and server-side runtime that specifies how clients should interact with APIs. GraphQL allows users to make API requests with just a few lines, rather than having to access complex endpoints with many parameters. This capability can make it easier to generate and respond to API queries, particularly more complex or specific requests that target multiple resources.

REST vs. SOAP

SOAP and REST represent different approaches to API design, describing rules and standards for how an API should interact with other applications. SOAP is a protocol while REST is a set of constraints that constitute an architectural style. Both use HTTP to exchange information.

REST is often considered a simpler alternative to SOAP because it is lightweight, flexible, transparent and relatively easy to use; SOAP requires users to write more code to complete each task than REST requires.

SOAP is more deterministic and robust (due to type checking), and proponents make the case that it is easier to use because of the SOAP support built into many development tools.[3] SOAP features built-in compliance, and developers often consider it a more secure protocol, better suited for situations with strict data integrity requirements.

RESTful systems support messaging in different formats, such as plain text, HTML, YAML, XML and JSON, while SOAP only allows XML. Each has their strengths, and the "right choice," might depend on use case. However, the ability to support multiple formats for storing and exchanging data is one of the reasons REST is a prevailing choice for building public APIs.

REST vs. GraphQL

GraphQL is a query language and API runtime that Facebook developed internally in 2012 before it became open source in 2015. GraphQL and REST are both stateless, use a client/server model and use HTTP. GraphQL solves for some limitations of REST, for example, providing the ability to more accurately target wanted resources with a single request.

REST APIs follows a fixed structure, and always return a whole data set for a specified object. If the request is more complex, spanning multiple resources, for example, the client must submit separate requests for each resource. These limitations can lead to under or over-fetching issues.

Neither REST nor GraphQL APIs are inherently superior. They're different tools that are suited to different tasks.

REST is generally easier to implement and can be a good choice when a straightforward, cacheable communication protocol with stringent access controls is preferred (for public-facing e-commerce sites like Shopify and GitHub, as one example).

GraphQL APIs enable more flexible, efficient data fetching, which can improve system performance and ease-of-use for developers. These features make GraphQL especially useful for building APIs in complex environments with rapidly changing front-end requirements.

APIs, Web Services and Microservices

APIs vs. Web Services

A web service is an internet software component that facilitates data transfers over a network. Because a web service exposes an application's data and functionality to other applications, in effect, every web service is an API. However, not every API is a web service.

APIs are any software component that serves as an intermediary between two disconnected applications. While web services also connect applications, they require a network to do so. Web services are typically private and only approved users can access them.

Apis, Microservices and Cloud-Native Development

Microservices is an architectural style that divides an application into smaller, independent components, often connected by using REST APIs. Building an application as a collection of separate services enables developers to work on one application component independent of the others, and makes applications easier to test, maintain and scale.

Microservices architecture has become more prevalent with the rise of cloud computing and, together with containers and Kubernetes, is foundational to cloud-native application development.

API Benefits

APIs simplify the design and development of new applications and services, and the integration and management of existing ones. They also offer significant benefits to developers and organizations at large.

Improved collaboration: The average enterprise uses almost 1,200 cloud applications, many of which are disconnected. APIs enable integration so that these platforms and apps can seamlessly communicate with one another. Through this integration, companies can automate workflows and improve workplace collaboration. Without APIs, many enterprises would lack connectivity, causing information silos that compromise productivity and performance.

Accelerated innovation: APIs offer flexibility, allowing companies to make connections with new business partners and offer new services to their existing market. This flexibility also enables companies to access new markets that can boost returns and drive digital transformation.

For example, the company Stripe began as an API with just seven lines of code. The company has since worked with many of the biggest enterprises in the world. Stripe has diversified to offer loans and corporate cards, and received a recent valuation of USD 65 billion.

Data monetization: Many companies choose to offer APIs for free, at least initially, so that they can build an audience of developers around their brand and forge relationships with potential partners. If the API grants access to valuable digital assets, a business monetizes it by selling access. This practice is referred to as the API economy.

When AccuWeather started its self-service developer portal to sell a wide range of API packages, it took just 10 months to attract 24,000 developers, selling 11,000 API keys. This move helped to build a thriving community in the process.

System security: APIs separate the requesting application from the infrastructure of the responding service and offer layers of security between the two as they communicate. For example, API calls typically require authentication credentials. HTTP headers, cookies or query strings can provide additional security during data exchange. An API gateway can control access to further minimize security threats.

User security and privacy: APIs provide added protection within a network. They can also provide another layer of protection for personal users. When a website requests a user's location (a location API provides this information), the user can decide whether to allow or deny this request.

Many web browsers and desktop and mobile operating systems have built-in permission structures. When an app must access files through an API, operating systems such as iOS, macOS, Windows and Linux use permissions for that access.

WEB-TO-DATABASE MIDDLEWARE

Web-to-database middleware is a type of software that allows web servers to communicate with database servers. It's a database server-side extension that simplifies data transfer and management.

Web-to-database middleware acts as a bridge between web applications and database servers. It provides a standardized interface for web applications to communicate with database servers. Web applications can use this interface to query, update, and manage data in databases

Examples of web-to-database middleware Java Database Connectivity (JDBC), Active Server Pages (ASP), and PHP Hypertext Preprocessor (PHP).

Middleware

Middleware is software that connects applications, tools, and databases to provide unified services to users. It's sometimes called plumbing because it connects applications together like a pipe

Benefits Of Web-To-Database Middleware:

- Allows users to submit forms on a web browser

- Allows web servers to return dynamic web pages based on a user's profile
- Simplifies data transfer and management
- Web-to-Database Middleware: Facilitates communication between web applications and databases.
- Client-Side Extensions: Enhancing web applications with database functionalities.
- XML and JSON: Structured data exchange formats over the web.

Challenges and Best Practices:

- Data Redundancy and Inconsistencies: Addressing issues arising from multiple databases.
- Security Concerns: Ensuring data protection during integration.
- Scalability: Managing growing data volumes and user demands.
- Performance Optimization: Enhancing the speed and efficiency of data operations.